Optimizing Heterogeneous Compute-in-Memory with Hybrid Dataflow and In-Network Reduction for Vision Transformer

Zexin Fu[†‡], Yihang Zuo[†], Yuzhe Ma[†], Jiayi Huang[†*]
[†]Thrust of Microelectronics, the Hong Kong University of Science and Technology (Guangzhou)
[‡]Guangdong-Macao Joint Laboratory for Modular Chip Design and Testing, the Hong Kong University of Science and Technology (Guangzhou)
{zexin.fu,yzuo099}@connect.hkust-gz.edu.cn, {yuzhema,hjy}@hkust-gz.edu.cn [*]Corresponding Author

***Abstract*—Transformer models have shown remarkable performance in AI tasks. However, their large model sizes and large-scale matrix computations heavily press the memory bandwidth. Compute-in-memory (CIM) solutions have emerged to address these issues by integrating computation into memory. Nevertheless, existing scalable and heterogeneous CIM designs further suffer from inefficiencies caused by isolated mapping strategies and communication redundancy. To address these issues, we propose Cross-Type Mapped Dataflow, an architecture-dataflow co-design that enables efficient and collaborative analog-digital CIM execution.**

Our approach features a heterogeneous analog-digital CIM architecture with flat network-on-chip (NoC) interconnects, enabling seamless cross-type communication. The core innovations include: a Hybrid Dataflow that schedules idle CIMs for cross-type collaboration on vector-matrix multiplication, and an In-Network Reduction (INR) mechanism that eliminates redundant data transfers by embedding Reduction operations within NoC communication. Experimental results show an average 3.57× speedup and 2.78× energy efficiency improvement over Homogeneous DCIM architecture. Optimized by the proposed Hybrid Dataflow and INR, our heterogeneous CIMs further achieve an average 4.63× speedup and 2.59× energy efficiency improvement over state-of-the-art X-Former-like design across various ViT workloads.

I. INTRODUCTION

In recent years, deep neural networks (DNNs) have demonstrated superior performance across various domains. As these models grow in complexity, their computational demands are increasing rapidly. This trend places significant pressure on the data access bandwidth, exacerbating the memory wall problem. Moreover, the rise of emerging models like Transformer [1] has further intensified this challenge.

Compute-in-Memory (CIM) has emerged as a promising computing paradigm for DNN workloads by integrating computational capabilities directly into memory. This approach offers high memory bandwidth and substantially reduces data movement between Processing Elements (PE) and memory, thereby alleviating memory access bottlenecks and enhancing computational efficiency. CIM is particularly well-suited for performing vector-matrix multiplication (VMM) operations, which are central to DNNs. Resistive RAM (RRAM) is considered one of the ideal memory cell candidates for CIM-based DNN accelerators due to their data persistence, high density, and low read energy [2]. These RRAM-based designs are typically paired with peripherals, such as ADCs, DACs, Wordline Driver, Switch Matrix and Shift Adders, to facilitate computation in the analog domain, referred to as analog CIM.

While RRAM-based Analog CIM designs offer advantages, they encounter significant challenges when applied to emerging Transformer models. Although VMMs remain predominant in Transformer, the attention mechanism in these models introduces dynamically generated intermediate weights, as opposed to the pre-trained weights in CNNs. This results in frequent write operations, imposing significant latency and energy overhead as well as accuracy damage.

To address these issues, some studies propose using SRAM-based analog CIM or digital CIM. The latter performs computations in the digital domain. Additionally, heterogeneous CIM designs, which combine Analog RRAM with Analog SRAM or Digital SRAM CIMs, are gaining attention because of their potential to overcome the limitations of homogeneous designs and achieve dual benefits.

979-8-3315-2710-5/25/$31.00 ©2025 IEEE

As systems scale to larger sizes, Network-on-Chip (NoC) has become a fundamental component of CIM accelerators. In typical designs, a single CIM module is limited in size and cannot accommodate entire models, necessitating the interconnection of multiple CIM modules via NoC. This allows data and instructions to be transferred through NoC in the form of messages. Compared to Bus-based CIM architecture, NoC-based designs offer better scalability and flexibility, particularly in model mapping and data flow scheduling, making them more adept at handling complex Transformer models.

Our work targets NoC-based heterogeneous CIM accelerators for Vision Transformer to address the limitations found in previous studies. Although previous work addresses the feasibility of using CIM for Transformers, certain inefficiencies still exist in Scalable and Heterogeneous CIMs. Firstly, prior research often maps different operations to distinct CIMs. Such isolated mapping strategies lead to underutilization, as one CIM type remains idle when the other is active. Secondly, Reduction operations mapped on global Reduction units introduce additional routing and communication redundancy, which limits system performance.

To solve the above problems, we propose Cross-Type Mapped Dataflow, a novel optimization approach combining two key innovations: (1) Cross-Type CIM Collaboration (i.e., Hybrid Dataflow), which schedules idle cross-type PEs for cooperative computing, and (2) In-Network Reduction, enabling on-the-fly data reduction during NoC transmission. This unified solution exploits previously unexplored opportunities by simultaneously leveraging spatial parallelism through cross-type PE utilization and temporal efficiency via in-network computation. The contributions of this work are as follows.

- We propose a flat and heterogeneous NoC-based architecture for Vision Transformer, to facilitate cross-type collaboration on VMM and Reduction operations compared with traditional hierarchical CIM architectures.
- We design a Hybrid Dataflow to enhance heterogeneous PE utilization and improve parallelism for VMM operations by effectively reusing cross-type CIMs.
- We design an In-Network Reduction mechanism to enable on-the-fly data reduction for both intra-type and cross-type CIMs.
- Experimental results show an average 3.57× speedup and 2.78× energy efficiency improvement over Homogeneous DCIM. Optimized by the proposed Hybrid Dataflow and INR, our heterogeneous CIMs further achieve an average 4.63× speedup and 2.59× energy efficiency improvement over state-of-the-art X-Former-like design across various ViT workloads.

II. Background and Related Work

A. Heterogeneous CIM for Vision Transformer

A typical Vision Transformer [3], [4] (ViT) first reshapes an image into 2D fixed-size patch sequences. Then transformer blocks, composed of Multi-head Self-attention and Feed Forward Network layers, process the sequence of patch tokens and finally make a prediction.

To speed up the inference of Vision Transformer, Compute-in-Memory (CIM), which modifies traditional memory arrays to enable computing within memory, has been considered a critical component in ViT hardware. The core Matrix Multiplication operators can be decomposed into Vector-Matrix Multiplication (VMM) operations, which can be naturally mapped onto CIM modules.

Due to the non-ideal effects of RRAM ACIM and the low-density of SRAM DCIM, a heterogeneous architecture has been adopted in prior research on CIM-based Transformer accelerators [5]–[16]. Firstly, the Dynamic Matrix Multiplication (DMM) in Transformer requires writing runtime-generated weights, which is unfriendly for RRAM-based CIM due to high energy cost and non-ideal effects. Previous studies have proposed using SRAM-based Analog CIM [7], [10], [13], [16] or Digital CIM [8], [9] to perform DMM operations. Secondly, Static Matrix Multiplication (SMM) remains prevalent. So, mapping SMM onto RRAM-based [5]–[9], [11], [14], [16] or SRAM-based Analog CIM [13], [15] still yields benefits.

Prior research on heterogeneous-CIM-based Transformer accelerators aims to reduce computation amounts of DMM through algorithm co-design [8], [14], [15]. Some studies have developed specific modules to leverage the varying sparsity, precision, and weight writing needs of operators [10], [12]. While diverse PEs enhance energy efficiency, they often result in lower PE utilization despite the potential for cross-type PE reuse. X-Former [7] proposed a Sequence Blocking Dataflow to improve PE utilization by overlapping computations between SRAM-based and RRAM-based PEs. However, its benefits are limited due to the operator ordering in Transformers. Previous approaches have overlooked opportunities for reusing cross-type but similar PEs during idle times, which we address with a hybrid dataflow in our work.

B. In-Network Reduction

Due to CIM's limited subarray size, mapping and partitioning VMM operations on CIMs generate additional Reduction operations (when partitioned in weight rows). For typical designs, Reduction operations are mapped to reduction units (e.g. adders or SIMD units). However, such designs introduce additional routing (sources-reduction unit, followed by reduction unit-destination) and communication redundancy. Inspired by dataflow principles, it is possible to keep the ideal sources-dst routing while conducting reduction operations during the NoC transmission process, referred to as In-Network Reduction.

Previous work has proposed basic designs to support reduction operations inside routers [17]–[19]. However, [17], [18] implemented In-Network Reduction based on simple routers, instead of pipelined routers adopted in the modern NoC, lacking practicality in real applications. INA [19] proposed an INR solution based on pipelined routers for CNN accelerators. However, it only considers fixed sequential INR routing in homogeneous architectures, hindering its application to heterogeneous architectures that require more complex routing to avoid deadlocks. Our proposed INR mechanism supports deadlock-aware routing for mix layouts, enabling both intra-type and cross-type source reductions (introduced by cross-type mapped VMMs) on heterogeneous architecture.

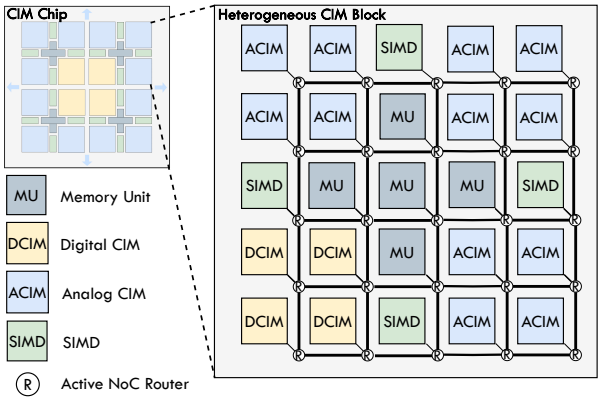


Fig. 1 Scale-up Architecture with Heterogeneous CIM Blocks

III. Architecture Design

The proposed architecture features a flat NoC-based design with heterogeneous PE components. The main design philosophy focuses on enabling seamless cross-type PE communication and flexible dataflows to enhance resource utilization.

A. Heterogeneous CIM Architecture

The heterogeneous CIM block is proposed to facilitate the accelerated computing of various operation types in Transformer models, as shown in Fig. 1. This architecture integrates four primary types of units, including Analog CIM PEs (ACIM), Digital CIM PEs (DCIM), SIMD PEs (SIMD), and Memory Units (MU). These units are interconnected via a mesh-based NoC. A mixed layout is carefully designed to minimize communication distances between cross-type PEs, such as the DCIM-ACIM pair and CIM-SIMD pair.

Analog CIM PEs are designed for Static VMM operations. Each complete PE consists of multiple crossbars and peripherals, similar to [20]. Additional dummy columns programmed to the off state are attached to the CIM crossbars to compensate for the finite on/off ratio effect. The SRAM-based Digital CIM PEs, following [21], are adopted for Dynamic VMM operations. The ratio of ACIM to DCIM PEs is set at 3:1, reflecting the predominance of Static VMMs among all VMM operators. Both DCIM and ACIM PEs maintain the same computation size, enabling seamless weight sharing between CIM PEs without data concatenation or splitting.

Memory Units are distributed SRAM-based on-chip buffers. We distribute MUs in the middle of each Heterogeneous CIM Block to facilitate memory access for shortcut staging and potential DRAM data buffering.

To handle the various and complex non-linear operations (e.g. Softmax, LayerNorm, Gelu, and Square Root), programmable SIMD PEs are introduced which support INT8 approximation following [22]. We also map Reduction operations to facilitate partial sum accumulation. All SIMDs are shared and distributed between cross-type CIM PEs.

B. Flat Interconnect Hierarchy

To facilitate more flexible cross-type PE communication, we adopt a flat NoC hierarchy. Compared with traditional CIM designs whose NoC interconnected and scheduling units are 3-level Tiles (Crossbar-PE-Tile), our design interconnects 2-level PEs (Crossbar-PE) via NoC to provide more communication resources and efficiency. In the Tile hierarchy, a PE can be scheduled for new tasks until the intra-Tile bus is available (may occupied by other PEs) for data transmission, which is inefficient for cross-type PE collaboration. In our architecture,

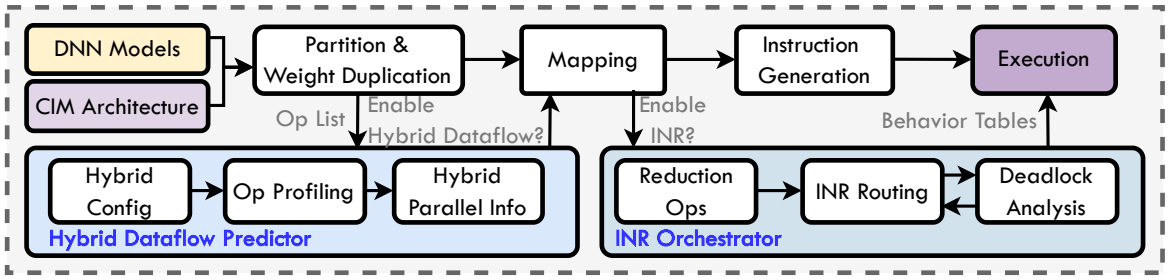


Fig. 2 Compilation Framework

Each PE is independently schedulable to initiate data transmission and computation at an earlier stage. This design especially benefits our proposed Hybrid Dataflow (VMMs on cross-type CIMs) and In-Network Reduction (Reductions from various CIMs).

C. Scale-up Design

The proposed architecture is designed for scalability to accommodate larger model sizes. We apply a spatial dataflow for ACIM PEs where static weights are all stored on-chip with temporary dataflow for DCIM PEs. To scale for larger networks, the number of ACIM units is increased to handle all Static VMM weights, ensuring sufficient DCIM capacity for the largest Dynamic VMM operator. The architecture enables the stitching of four Heterogeneous CIM blocks into a larger block, with DCIMs centrally positioned for weight duplication. Once sufficient DCIMs are in place, ACIM PEs are added in concentric rings from the edges of the large block, forming the final architecture, as depicted in Fig. 1.

D. Compilation Framework

An end-to-end compilation framework has been implemented for the proposed architecture. The general flow is shown in 2. The target DNN model and CIM architecture configurations are input into the framework. The compiler first partitions operators in the model and conducts weight duplication (the duplication factor is predefined in the input configuration). The partitioned operators are then mapped to PEs. We adopt a typical zigzag mapping strategy for NoC-interconnected PEs. After mapping, each PE generates its executable instructions according to its allocated operators. Finally, the simulator starts its execution.

The compilation framework also contains optimization processes for Hybrid Dataflow and In-Network Reduction, which will be explained in Section IV-C and Section V-B3.

IV. Hybrid Dataflow

A. Dataflow Principles

To unlock the potential of fine-grained designs, we incorporate dataflow architectures principles [23], [24]. In this framework, computation or communication tasks are executed immediately upon satisfying dependencies. We partition row-wise operators (e.g., VMM, Weight-Writing, LayerNorm, Softmax), enabling row-by-row execution instead of layer-by-layer execution.

B. Dataflow Design

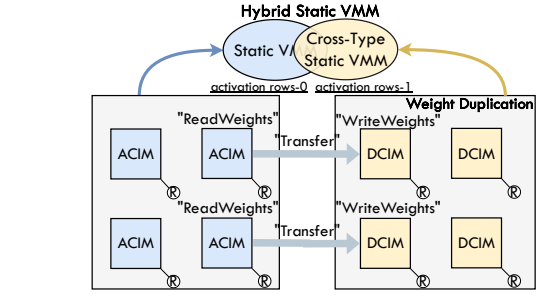
We proposed a novel dataflow for heterogeneous CIM architectures that efficiently utilizes idle DCIMs for weight duplication to support the Static VMM operations of ACIMs, shown in Fig. 3 (a). Weight duplication is a widely adopted technique that enhances computation parallelism by copying weights to multiple ACIMs with additional PEs.

In heterogeneous CIM architectures, it is common for one type of PE to be idle while the other type is active. This may be due to operator ordering in models and the distinct designs of operators, such as the separation of dynamic and static VMMs. Consequently, opportunities for cross-type PE reuse are often overlooked.

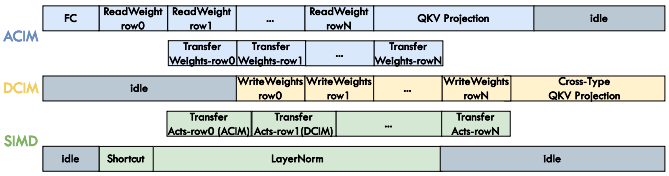
The proposed dataflow leverages the weight-stationary properties and mapping types of both ACIM and DCIM. When DCIMs are idle, weights are read from ACIMs, transferred to and written into DCIMs, thereby increasing parallelism for Static VMM operations. This strategy effectively enhances PE utilization in the heterogeneous architecture compared to previous proposals.

Owing to the reusable nature of DCIM, it is often utilized in a fully parallel manner to accelerate operations. When reuse occurs, weights must be read and written in advance. As a result, the hybrid dataflow is particularly well-suited for Static VMMs (e.g., QKV Projection and FFN) that have a certain interval since their last DCIM computation, effectively hiding the latency associated with weight movement. In cases where the subsequent operation is also mapped to DCIM, choosing the appropriate degree of parallelism can optimize the pipeline between operators, ultimately enhancing overall system performance. To minimize the overhead of weight duplication, we implement a weight duplication pipeline that follows a row-by-row pattern—compromising Read Weights, Transfer, and Write Weights—to hide the latency of weight movements. Figure 3 (b) presents the execution timeline of QKV projection operations with Hybrid Dataflow.

The Hybrid Dataflow can be further extended to more types of



(a) Mapping and Execution Process of Hybrid Static VMM



(b) Timeline for QKV Projection Operations with Hybrid Dataflow

Fig. 3 Hybrid Dataflow

heterogeneous PEs, which we leave as future work. For instance, idle SIMD units could be leveraged to collaboratively execute ACIM or DCIM operations. By leveraging similarities between cross-type PEs in heterogeneous architectures, spatial parallelism can be significantly enhanced even when model structures constrain pipeline or homogeneous operation parallelism.

C. Hybrid Dataflow Compilation Process

The Hybrid Dataflow compilation process has been integrated into our end-to-end compilation framework, as shown in Fig.2. After partitioning and weight duplication, the operation list is processed via Hybrid Dataflow Predictor. Based on the hybrid configuration (e.g. mapping Static VMM on cross-type DCIMs), the predictor automatically profiles the target operator to check IDLE resources at runtime. After profiling, a hybrid parallel indicating the cross-type operator parallelism is delivered to the mapping process.

V. In-Network Reduction

A. INR Overview

The proposed In-Network Reduction brings the following benefits. Both additional routing to reduction units and overall transmission amounts can be reduced, alleviating traffic congestion. Besides, the reduction latency can be overlapped with the router pipeline stages.

Three core issues need to be addressed for the INR mechanism. The first issue is how to route a group of source nodes through a shared reduction path. So we adopt software scheduled routing instead of traditional hardware routing (Section V-B1), ensuring that the source nodes can share the path by pre-compilation (Section V-B3). Secondly, due to the potential for NoC congestion, different source messages may arrive asynchronously. This requires that: 1) the destination node should recognize reduction messages by a unique ID representing the source group, rather than a specific message ID (Section V-B1); and 2) specially designed routers capable of capturing asynchronous source flits (Section V-C). The third issue is the deadlock risk in INR for heterogeneous architectures with cross-type reduction. We address this through specialized routing algorithm design (Section V-B2) and deadlock analysis during compilation (Section V-B3).

B. Software Scheduled Routing

1) INR Flow Mechanism: Our design utilizes software scheduling to precompute the routing for each INR operation. Compared to hardware routing, this approach not only reduces the complexity of routing in terms of runtime logic and hardware overheads but also leverages global awareness to determine how to route source nodes onto a shared reduction path.

We define a Reduction operation mapped to INR as a flow. The destination node identifies reduction messages by a Flow ID (FID), rather than a specific Message ID (MID). During compilation, each INR operation is assigned a unique Flow ID. For the routing of each flow, the locations of the source nodes and the destination node are taken into account, and the reduction orders and routing within each flow are precomputed during compilation. The routing information is then written into the flow behavior table of each router along the reduction path. During runtime, the routing information will be retrieved from the table.

2) Routing for Heterogeneous Architecture: In homogeneous architectures, INR deadlock can be avoided through sequential mapping and routing. However, for heterogeneous architectures, the types of source nodes may differ, and the source and destination nodes may also vary in type. Due to the mixed layout of different types of PEs,

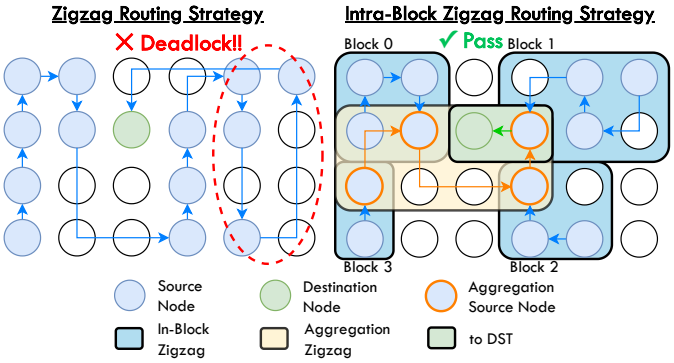


Fig. 4 Routing Strategies in Heterogeneous Architectures

routing may form circles (an example is depicted in Fig. 4), leading to potential deadlock . Therefore, a considerate routing design is essential.

Compared to point-to-point transmission, INR involves multiple sources, requiring the determination of both the reduction order and the routing. In a mesh topology, distance-based methods (e.g., determining the transmission order based on the distance from the destination) can lead to path intersections, resulting in circles.

Based on observations of the distribution of reduction sources and destinations, we find that:

- Most sources tend to appear in clusters. This is why previous works adopted sequential routing (when sources and the destination are distributed in the same row or column) or zigzag routing (when distributed across different rows or columns).
- When the destination is located in the middle of a cluster of sources (a characteristic of heterogeneous architectures), routing circles are difficult to avoid.

We propose an INR routing algorithm called Intra-Block Zigzag. It retains the basic zigzag routing to leverage the characteristic of sources appearing in clusters, but ensures that the destination of each zigzag is always at the corner of the source cluster to avoid deadlock.

Intra-Block Zigzag, as shown in Fig. 4, consists of three stages. For each flow, the source nodes are divided into a maximum of four blocks by the destination node. In the first stage, In-Block Zigzag, the sources within each block form INR routing in a zigzag manner. In the second stage, Aggregation Zigzag, the last source in each block performs INR in a zigzag manner. In the third stage, toDST, the last source from the aggregation stage is routed to the destination. During the entire process, the zigzag direction in each stage can be reversed to prevent forming circles between stages.

3) INR Compilation Process: As shown in Fig. 2, after mapping, all Reduction operators are sent to the INR Orchestrator. Corresponding flows and routing are generated based on the specified routing algorithm. A deadlock analysis module is implemented based on the model structure, which analyzes potential circles between specified flows and other NoC transmissions in the timespan of concurrent operators. If a deadlock risk is detected, the INR routing is iteratively regenerated by reversing each zigzag's direction. If the deadlock cannot be resolved by any reversing, we select the flow with the fewest sources involved in the formed circle and remap it to SIMD units instead. Once no deadlock risk remains, the flow routing information is written into the behavior table of routers.

C. Active Router

We integrate INR functionalities with the original pipelines of multi-stage pipelined routers and name them active routers. Figure 5

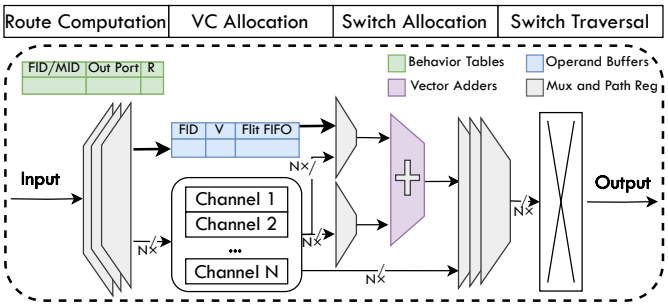


Fig. 5 Microarchitecture of Active Router

shows the microarchitecture of the active router, with colored elements indicating the new components added.

Behavior Tables: used to obtain pre-computed routing information. A Common Behavior Table is for non-INR messages while a Flow Behavior Table is for INR messages. Flow Behavior Table is separated from the common one due to the asynchronism of INR making it hard to predict the specific Message ID. During Route Computation, active routers do not compute routing and turn to read results from behavior tables.

Both Behavior Tables have multiple entries of output ports indexed by the Message ID or Flow ID. For the Flow Behavior Table, an additional bit (R) is added to each entry, indicating whether the flow message will be reduced in this router.

Operand Buffers: used to deal with the asynchronous arrival of source messages. For the first arrived source, its flits will be reserved in the Flit FIFO of Operand Buffers. During Route Computation, Operand Buffers are looked up for the head flit of each reduction message (when only one reduction message of a flow is input). If another source has been staged in the buffer, both sources will be directed to Shared Vector Adders. Otherwise, the reduction message will be staged to a buffer entry during VC Allocation.

Each Flit Buffer is indexed by its Flow ID. A valid bit (V) indicates if a source has been reserved in the Flit FIFO.

Shared Vector Adders: used for conducting reduction operations. It is shared by all channels. The resource contention for the adder will be avoided during compilation. The Reduction operation is performed during Switch Allocation.

MUX and Path Registers: used for path navigation. The reduction flits in active routers may be navigated from/to Operand Buffers, Shared Vector Adders, and channels. We add Multiplexers and Path Registers to support diverse path configurations. These register control signals will be generated during Route Computation.

VI. Evaluation

A. Experimental Setup

1) Hardware Modeling: We developed a cycle-level simulator for NoC-based heterogeneous CIM architecture. Digital CIM data is from [21] and scaled to 22nm via [25] while analog CIM data is extracted from NeuroSim v1.3 [26]. SIMD modules for non-linear operations are implemented in NeuroSim following [22]. To estimate NoC performance, we extend and integrate BookSim [27] and obtain power results from the DSENT power model [28]. PPA data of off-chip DRAM access is obtained from CACTI [29]. We modify NeuroSim v2.1 [30] to evaluate the non-ideal effects.

The system configuration is summarized in Table I. The number of ACIMs is configured to store all Static VMM weights, while the number of DCIMs is set to exceed the required resources by the largest Dynamic VMM layer.

TABLE I Experimental Setup

Module	Parameter	Value
Analog CIM	Cell Type	2-bit RRAM; Ron-6KΩ; On/Off-17 [31]
	Crossbar size	128x128
	ADC Config	8-bit Flash ADC; Shared by 8 cols
	Device Non-ideality	DriCoe-0.001˜0.005; LTP-1.75, LSP-1.46
	Read Voltage	0.3V
Digital CIM	Cell Type	1-bit SRAM
	Subarray size	128x256
SIMD	Vector Size	16
NoC	Message Size	128 Bytes; 128-bit Flit
System	Quantization	W8A8; I-ViT [22]
	Clock Frequency	200MHz
	Technology	22nm
Computation Granularity	1x1	ACIM: 1 Crossbar/PE; DCIM: 1 Subarray/PE
	2x2	ACIM: 4 Crossbars/PE; DCIM: 4 Subarrays/PE

2) Benchmarks: We evaluate our proposed designs using two representative Vision Transformers (ViTs): DeiT-Tiny and DeiT-Small [4], using the ImageNet 2012 dataset [32].

B. Performance and Energy Evaluation

Since patch merging [33] and pruning [34] have been emerging trends for ViT, we evaluate various patch token counts (i.e., SL) for DeiT-Tiny and DeiT-Small layers.

Considering the impracticability of RRAM-based HomoACIM (as depicted in Section VI-C), we only focus on heterogeneous CIMs and Homogeneous DCIM (HomoDCIM) for evaluation. We determine the DCIM PE resources for HomoDCIM under the same area budgets as heterogeneous CIMs. Due to the high density of RRAM, DCIM PEs are 2.95× to 5.42× larger than RRAM ACIM PEs in our experiments. As a result, HomoDCIM has a 3.64× (DeiT-Tiny) and 2.66× (DeiT-Small) smaller weight capacity. A temporal dataflow is employed for HomoDCIM, requiring loading weights from DRAM.

For Heterogeneous CIMs, we first adopt the proposed architecture with the isolated mapping for heterogeneous PEs (as in previous work) and the original Row-by-Row Dataflow as a baseline (IsolatedMap). We also implement an X-Former [7] like architecture but tailored for a fine-grained NoC-based design (XFormerLike). A block factor of 4 is applied to the Sequence Blocking Dataflow of X-Former, partitioning the sequence length into 4 blocks. The block factor = Sequence Length / Sequence Block (SB parameter used in the X-Former paper), and we adopt a fixed block factor value to balance the impact of different SB parameters. Then we evaluate the proposed Hybrid Dataflow (HD) and INR separately and together. Besides, INA [19] is implemented in our framework as a baseline of In-Network Reduction. Since it only supports simple sequential routing (which we extended to zigzag routing for cross-row/column traversal), it may fail to route complex INR flows. When routing fails, we fall back to mapping the INR operation to SIMD modules.

As shown in Fig. 6, the proposed heterogeneous CIMs (IsolatedMap) archive a maximum speedup of 16.67× and an average speedup of 3.57× compared to HomoDCIM, as well as an average energy efficiency improvement of 2.78×. This is because the expensive DRAM access, which is difficult to overlap with other operations in HomoDCIM, has dominated the total latency and energy.

The Row-by-Row dataflow achieves results comparable to Sequence Blocking Dataflow due to their shared fine-grained design, but the latter improves upon this by partitioning Q×K operators into smaller blocks. While dependencies in Softmax operations limit pipeline length, smaller blocks enable early transmission, efficient weight-writing, and help satisfy row-wise dependencies in Softmax.

Optimized by Hybrid Dataflow, an average speedup of 1.93× and energy efficiency improvement of 1.56× can be achieved over the baseline with isolated mapping and Row-by-Row dataflow. Compared to the XFormer-like design, Hybrid Dataflow also achieves an average speedup of 1.82× and an average energy efficiency improvement

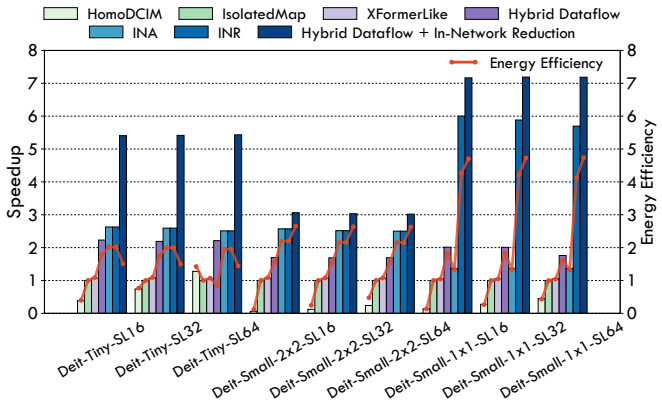


Fig. 6 Speedup and Normalized Dynamic Energy Over Heterogeneous CIMs (isolated mapping)

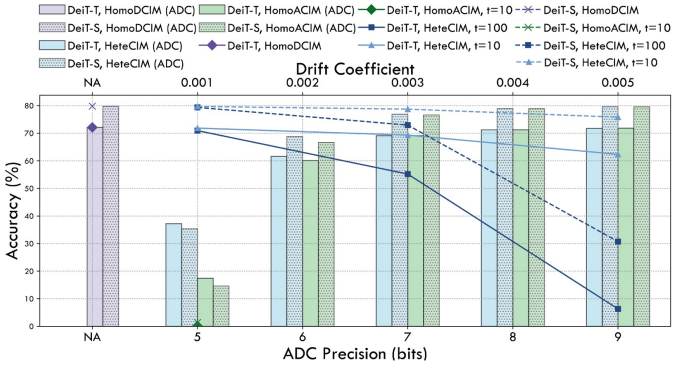


Fig. 7 Accuracy with ADC and Device Non-Ideal Effects. Bars represent ADC effects (bottom x-axis); point-line plots indicate device parts (top x-axis).

of 1.46×. Both Row-by-Row and Sequence Blocking Dataflows share a common approach of partitioning row-wise operations to enable finer-grained access and execution between different types of PEs. However, the benefits of this approach are primarily limited to the QKV Projection and Q×K operations due to the dependency constraints imposed by the Softmax operations. In contrast, the Hybrid Dataflow leverages the idle PE types to facilitate static VMMs, providing more opportunities for efficiency in Transformer models. In the case of DeiT-Small, when the NoC size is partitioned into finer-grained hardware, Hybrid Dataflow achieves greater performance improvements, demonstrating that additional communication resources benefit cross-type weight duplication. In contrast, X-Former shows relatively similar speedup ratios under different granularities.

The INR design demonstrates a speedup of 3.37× and a 2.63× energy efficiency improvement over the IsolatedMap baseline on average, as well as a speedup of 1.63× and a 1.47× energy efficiency improvement over INA, owing to the transmission reduction in advance and reduced routing. Compared to INA, the performance shows comparable performance on DeiT-Tiny layers with 1×1 granularity and DeiT-Small layers with 2×2 granularity. This is because the number of source nodes for reduction operations in these cases is limited and their distribution is less complex, allowing both INA and INR to successfully route all reduction operations. However, under fine-grained hardware configurations and larger models, INR demonstrates significantly superior performance over INA. For instance, on DeiT-Small layers with 1×1 granularity, INR achieves 4.28× speedup and 3.17× energy efficiency improvement over INA on average. This is because INA's simple sequential routing algorithm fails to handle the intricate distribution of source nodes in large-scale heterogeneous architectures, causing many cases to fall back to SIMD modules. As model size increases, the INA exhibits a higher routing failure rate (e.g., 2/21 in DeiT-Small and 12/51 in DeiT-Base), while the INR consistently achieves valid routing solutions. Consequently, INA can only deliver limited performance gains.

Combining with Hybrid Dataflow, the HD-INR architecture can further speed up by 4.91×, with a 2.64× energy efficiency improvement. Compared with the XFormer-like design, a 4.63× speedup, and a 2.47× energy efficiency improvement are achieved. Results show great potential for cross-type collaboration in heterogeneous CIMs.

C. Accuracy Validation

In this section, the accuracies of heterogeneous CIMs (HeteCIM), homogeneous RRAM-based ACIM (HomoACIM), and homogeneous DCIM (HomoDCIM) are compared. ADC quantization and device non-idealities are included for ACIM parts. For device non-idealities, we consider cell variation, static data retention, and the non-ideal properties of weight updates. The finite on/off ratio impact is not incorporated, as prior work has demonstrated that it can be effectively mitigated via the dummy column scheme [35].

With ADC quantization effects, as shown in Fig. 7, the accuracy of both Homogeneous ACIM and Heterogeneous CIMs drops sharply when the ADC resolution is six bits or lower, rendering them ineffective. When the ADC resolution reaches 8 bits or higher, the accuracy loss is negligible, which is less than 1%.

As shown in Fig. 7, the accuracy of RRAM-based HomoACIM drops to 0.12% (DeiT-Tiny) and 1.38% (Deit-Small) due to the substantial overhead associated with weight updates, making it unsuitable for Transformer models. In contrast, heterogeneous CIMs demonstrate significant potential, achieving 71.88% (DeiT-Tiny) and 79.74% (DeiT-Small) accuracy when $DriCoe = 0.001$ for a timespan of 10s. It is worth noting that periodic weight calibration is still necessary for heterogeneous CIM, especially when $DriCoe \geq 0.003$. Device engineering [36] and circuit-device co-design [37] can further alleviate effects, which is out of the scope of this work.

VII. Conclusion

Transformer models have revolutionized AI tasks with their exceptional performance. However, they face significant computational and memory efficiency challenges due to their scale and complexity. Our solution integrates a heterogeneous analog/digital CIM architecture for Vision Transformers, combining RRAM-based analog and SRAM-based digital CIMs with a hybrid dataflow and In-Network Reduction. The architecture alone achieves 3.57× speedup and 2.78× energy efficiency improvement over homogeneous DCIM, while the Hybrid Dataflow and INR further deliver 4.63× speedup and 2.59× energy efficiency improvement over X-Former-like designs. These results highlight the critical role of architecture-dataflow co-design in maximizing heterogeneous CIM potential for AI acceleration.

Acknowledgment

This work was supported in part by the Guangdong Basic and Applied Basic Research Foundation (No. 2023A1515110353), the Department of Education of Guangdong Province (No. 2024KTSCX037), the Guangdong Science and Technology Department (No. 2025B1212150003), the Guangdong Provincial Project (No. 2023QN10X252), and GDIC.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," *Advances in Neural Information Processing Systems*, 2017.

[2] H. Jiang, S. Huang, and S. Yu, "Compute-in-Memory Architecture," in *Handbook of Computer Architecture*, A. Chattopadhyay, Ed. Singapore: Springer Nature Singapore, 2023, pp. 1–40.

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale," *arXiv preprint arXiv:2010.11929*, 2020.

[4] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training Data-Efficient Image Transformers & Distillation Through Attention," in *International Conference on Machine Learning*, 2021.

[5] A. F. Laguna, M. M. Sharifi, A. Kazemi, X. Yin, M. Niemier, and X. S. Hu, "Hardware-Software Co-Design of an In-Memory Transformer Network Accelerator," *Frontiers in Electronics*, 2022.

[6] S. Jain, H. Tsai, C.-T. Chen, R. Muralidhar, I. Boybat, M. M. Frank, S. Woźniak, M. Stanisavljevic, P. Adusumilli, P. Narayanan, K. Hosokawa, M. Ishii, A. Kumar, V. Narayanan, and G. W. Burr, "A Heterogeneous and Programmable Compute-In-Memory Accelerator Architecture for Analog-AI Using Dense 2-D Mesh," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2023.

[7] S. Sridharan, J. R. Stevens, K. Roy, and A. Raghunathan, "X-Former: In-Memory Acceleration of Transformers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2023.

[8] S. Liu, C. Mu, H. Jiang, Y. Wang, J. Zhang, F. Lin, K. Zhou, Q. Liu, and C. Chen, "HARDSEA: Hybrid Analog-ReRAM Clustering and Digital-SRAM In-Memory Computing Accelerator for Dynamic Sparse Self-Attention in Transformer," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2023.

[9] W. Li, M. Manley, J. Read, A. Kaul, M. S. Bakir, and S. Yu, "H3DAtten: Heterogeneous 3-D Integrated Hybrid Analog and Digital Compute-in-Memory Accelerator for Vision Transformer Self-Attention," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2023.

[10] Y. Luo and S. Yu, "H3D-Transformer: A Heterogeneous 3D (H3D) Computing Platform for Transformer Model Acceleration on Edge Devices," *ACM Transactions on Design Automation of Electronic Systems*, 2024.

[11] P. Dhingra, J. Doppa, and P. P. Pande, "HeTraX: Energy Efficient 3D Heterogeneous Manycore Architecture for Transformer Acceleration," in *Proceedings of the 29th ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, 2024.

[12] Y. Qiu, Y. Ma, M. Wu, Y. Jia, X. Qu, Z. Zhou, J. Lou, T. Jia, L. Ye, and R. Huang, "Quartet: A 22nm 0.09mJ/Inference Digital Compute-in-Memory Versatile AI Accelerator with Heterogeneous Tensor Engines and Off-Chip-Less Dataflow," in *2024 IEEE Custom Integrated Circuits Conference (CICC)*, 2024.

[13] G. Yin, Y. Chen, M. Lee, X. Du, Y. Ke, W. Tang, Z. Chen, M. Zhou, J. Yue, H. Yang, H. Jia, Y. Liu, and X. Li, "A 28nm 8928Kb/mm2-Weight-Density Hybrid SRAM/ROM Compute-in-Memory Architecture Reducing >95% Weight Loading from DRAM," in *2024 IEEE Custom Integrated Circuits Conference (CICC)*, 2024.

[14] J. Cai, M. A. Kaleem, R. Genov, M. R. Azghadi, and A. Amirsoleimani, "In-Memory Transformer Self-Attention Mechanism Using Passive Memristor Crossbar," in *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2024.

[15] A. Moradifirouzabadi, D. S. Dodla, and M. Kang, "An Analog and Digital Hybrid Attention Accelerator for Transformers with Charge-based In-Memory Computing," in *2024 IEEE European Solid-State Electronics Research Conference (ESSERC)*, 2024.

[16] C. Wang, Z. Chen, and S. Huang, "MICSim: A Modular Simulator for Mixed-signal Compute-in-Memory based AI Accelerator," in *Proceedings of the 30th Asia and South Pacific Design Automation Conference*, 2025.

[17] B. Wang, J. Zhou, W.-F. Wong, and L.-S. Peh, "Shenjing: A Low Power Reconfigurable Neuromorphic Accelerator with Partial-Sum and Spike Networks-on-Chip," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020.

[18] K. Zhou, Y. He, R. Xiao, J. Liu, and K. Huang, "A Customized NoC Architecture to Enable Highly Localized Computing-on-the-Move DNN Dataflow," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2021.

[19] B. Tiwari, M. Yang, X. Wang, and Y. Jiang, "In-Network Accumulation: Extending the Role of NoC for DNN Acceleration," in *2022 IEEE 35th International System-on-Chip Conference (SOCC)*, 2022.

[20] J. Lee, A. Lu, W. Li, and S. Yu, "Neurosim v1.4: Extending Technology Support for Digital Compute-in-Memory Toward 1nm Node," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2024.

[21] J. Chen, F. Tu, K. Shao, F. Tian, X. Huo, C.-Y. Tsui, and K.-T. Cheng, "AutoDCIM: An Automated Digital CIM Compiler," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023.

[22] Z. Li and Q. Gu, "I-ViT: Integer-Only Quantization for Efficient Vision Transformer Inference," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[23] C. Bai, X. Wei, Y. Zhuo, Y. Cai, H. Zheng, B. Yu, and Y. Xie, "Klotski: DNN Model Orchestration Framework for Dataflow Architecture Accelerators," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2023.

[24] R. Pelke, J. Cubero-Cascante, N. Bosbach, F. Staudigl, R. Leupers, and J. M. Joseph, "CLSA-CIM: A Cross-Layer Scheduling Approach for Computing-in-Memory Architectures," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2024.

[25] S. Sarangi and B. Baas, "DeepScaleTool: A Tool for the Accurate Estimation of Technology Scaling in the Deep-Submicron Era," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021.

[26] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, "DNN+NeuroSim: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators with Versatile Device Technologies," in *2019 IEEE International Electron Devices Meeting (IEDM)*, 2019.

[27] N. Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. E. Shaw, J. Kim, and W. J. Dally, "A Detailed and Flexible Cycle-Accurate Network-on-Chip Simulator," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2013.

[28] C. Sun, C.-H. O. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, and V. Stojanovic, "DSENT - A Tool Connecting Emerging Photonics with Electronics for Opto-Electronic Networks-on-Chip Modeling," in *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip (NOCS)*, 2012.

[29] R. Balasubramonian, A. B. Kahng, N. Muralimanohar, A. Shafiee, and V. Srinivas, "CACTI 7: New Tools for Interconnect Exploration in Innovative Off-Chip Memories," *ACM Transactions on Architecture and Code Optimization*, 2017.

[30] X. Peng, S. Huang, H. Jiang, A. Lu, and S. Yu, "DNN+NeuroSim V2.0: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators for On-Chip Training," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[31] P. Jain, U. Arslan, M. Sekhar, B. C. Lin, L. Wei, T. Sahu, J. Alzate-Vinasco, A. Vangapaty, M. Meterelliyoz, N. Strutt *et al.*, "13.2 A 3.6Mb 10.1Mb/mm2 Embedded Non-Volatile ReRAM Macro in 22nm FinFET Technology with Adaptive Forming/Set/Reset Schemes Yielding Down to 0.5V with Sensing Time of 5ns at 0.7V," in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*, 2019.

[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, 2015.

[33] Y. Wang, R. Huang, S. Song, Z. Huang, and G. Huang, "Not All Images Are Worth 16x16 Words: Dynamic Transformers for Efficient Image Recognition," *Advances in Neural Information Processing Systems*, 2021.

[34] Y. Xu, Z. Zhang, M. Zhang, K. Sheng, K. Li, W. Dong, L. Zhang, C. Xu, and X. Sun, "Evo-ViT: Slow-Fast Token Evolution for Dynamic Vision Transformer," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[35] S. Huang, H. Jiang, and S. Yu, "Hardware-Aware Quantization/Mapping Strategies for Compute-in-Memory Accelerators," *ACM Transactions on Design Automation of Electronic Systems*, 2023.

[36] I. Giannopoulos, A. Sebastian, M. Le Gallo, V. Jonnalagadda, M. Sousa, M. Boon, and E. Eleftheriou, "8-bit Precision In-Memory Multiplication with Projected Phase-Change Memory," in *2018 IEEE International Electron Devices Meeting (IEDM)*, 2018.

[37] S. Ambrogio, M. Gallot, K. Spoon, H. Tsai, C. Mackin, M. Wesson, S. Kariyappa, P. Narayanan, C.-C. Liu, A. Kumar, A. Chen, and G. W. Burr, "Reducing the Impact of Phase-Change Memory Conductance Drift on the Inference of large-scale Hardware Neural Networks," in *2019 IEEE International Electron Devices Meeting (IEDM)*, 2019.