TensorCIM: Digital Computing-in-Memory Tensor Processor With Multichip-Module-Based Architecture for Beyond-NN Acceleration

Yiqi Wang, *Graduate Student Member, IEEE*, Zihan Wu, Weiwei Wu, Leibo Liu, *Senior Member, IEEE*, Yang Hu, *Member, IEEE*, Shaojun Wei, *Fellow, IEEE*, Fengbin Tu, *Member, IEEE*, and Shouyi Yin, *Senior Member, IEEE*

***Abstract*— While neural networks (NNs) have achieved great results in various intelligent tasks like image classification and speech recognition, real-world scenarios have more applications beyond just NN processing like graph convolutional network (GCN) and deep-learning recommendation model (DLRM), which typically consist of sparse gathering (SpG) and sparse algebra (SpA). Their large application size leads to substantial data movement. Although the fusion of digital computing-in-memory (CIM) and multichip-module (MCM) can reduce data movement efficiently and scale out CIM's capacity in a high-yield solution, the MCM-CIM system raises new challenges for beyond-NN acceleration: SpG involves repeated off-chip DRAM access, interchiplet access, and redundant reduction operations; SpA suffers from inter-CIM workload imbalance and intra-CIM under-utilization. Thus, we design TensorCIM as the CIM processor chiplet with three corresponding features: 1) the redundancy-eliminated gathering manager (REGM) dynamically maintains frequently accessed features and reduction results in the CIM to eliminate redundant accesses and reductions; 2) the equal operation-based CIM initializer (EOCI) calculates effective multiply-accumulation (MAC) operations and initializes CIM macros with a balanced inter-CIM workload at the subarray level; and 3) the input-lookahead CIM (ILA-CIM) architecture looks ahead at future inputs to fully utilize CIM logic. The fabricated MCM-CIM system consumes only 3.7 nJ/Gather for the GCN model, achieving 8.3-TFLOPS/W algebra efficiency at FP32.**

***Index Terms*— Computing-in-memory (CIM), floating-point, graph neural network (NN), multichip-module (MCM), recommendation system, sparsity.**

Manuscript received 17 January 2024; revised 8 April 2024; accepted 24 May 2024. Date of publication 13 June 2024; date of current version 30 January 2025. This article was approved by Associate Editor Priyanka Raina. This work was supported in part by the National Key Research and Development Program under Grant 2018YFB2202600, in part by the NSFC under Grant U19B2041 and Grant 61774094, in part by the Beijing Science and Technology (S&T) Project under Grant Z191100007519016, and in part by the Beijing Innovation Center for Future Chip. *(Corresponding authors: Fengbin Tu; Shouyi Yin.)*

Yiqi Wang, Zihan Wu, Weiwei Wu, Leibo Liu, Yang Hu, Shaojun Wei, and Shouyi Yin are with the School of Integrated Circuits, Beijing Innovation Center for Future Chip, and Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: yinsy@tsinghua.edu.cn).

Fengbin Tu is with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, China (e-mail: fengbintu@ust.hk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSSC.2024.3406569.

Digital Object Identifier 10.1109/JSSC.2024.3406569

I. Introduction

While neural networks (NNs) have achieved great results in various intelligent tasks [10], real-world scenarios have more applications that have computational and data-movement requirements beyond those seen in typical NN processing, like graph convolutional network (GCN) [8] for graph-based learning and deep-learning recommendation model (DLRM) [14] for recommendation (see Fig. 1). GCN aggregates features on a large graph and then performs an NN-based combination. DLRM gathers and reduces items from an embedding table and generates recommendation results via fully connected layers. However, prior artificial intelligence (AI) chip research focuses on NN-only applications, and we demand a new architecture for beyond-NN acceleration.

To realize efficient beyond-NN acceleration, we first analyze the workload features of beyond-NN applications. Beyond-NN applications typically consist of sparse gathering (SpG) and sparse algebra (SpA). SpG gathers and reduces tensors from sparsely distributed addresses, like GCN's aggregation phase and DLRM's embedding layer. SpA refers to NN-based sparse tensor multiplication for the gathered tensors (GTs) like GCN's combination phase and DLRM's fully-connected (FC) layer. Besides, due to the large application size, data movement is the main bottleneck for beyond-NN acceleration, and FP32 precision is usually required for high accuracy.

Our solution for beyond-NN applications is the fusion of digital computing-in-memory (CIM) and multichip-module (MCM), which reduces data movement while maintaining high precision and accommodates large application sizes, respectively. In the past few years, digital CIM has proven to be an efficient and precise architecture for reducing data movement [4], [5], [13], [16], [19], [20]. The large-scale beyond-NN applications bring huge computational and storage requirements and motivate the demand for scaling out digital CIM processors. MCM provides a high-yield scalable solution for CIM scaling by integrating multiple chiplets in one package to build an MCM-CIM system [15], [28], [29].

Fig. 2 shows a typical MCM-CIM system with four CIM chiplets. In our MCM-CIM system, workloads are partitioned to each chiplet. The CIM chiplet mainly relies on its own DRAM channel to perform SpG and SpA. The MCM architecture enlarges CIM capacity. Intermediate data are

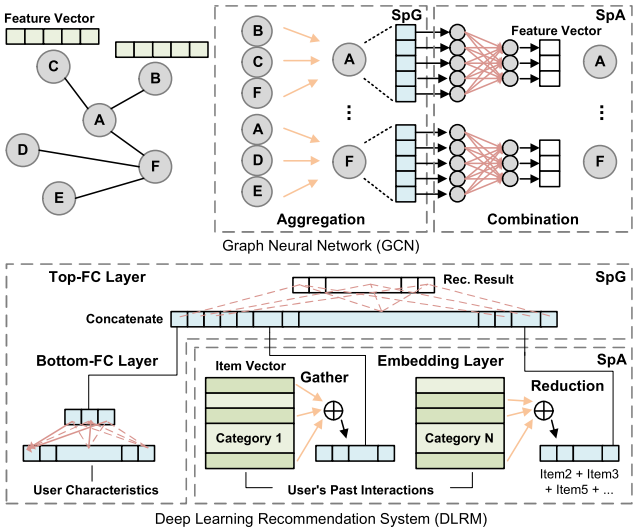


Fig. 1. Two typical beyond-NN applications: GCN and DLRM.

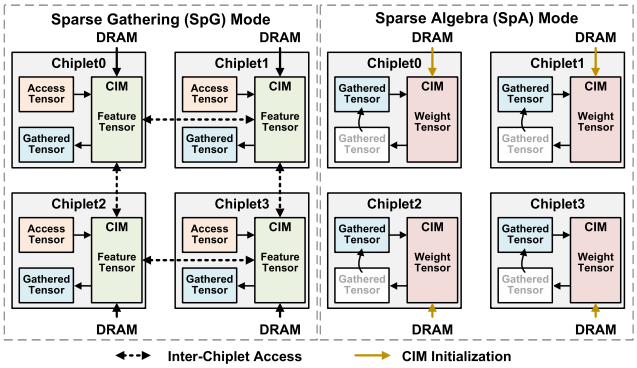


Fig. 2. MCM-CIM system with four CIM chiplets in the SpG and SpA modes.

communicated with limited interchiplet bandwidth. In beyond-NN acceleration, all chiplets are first configured in the **SpG mode**, fetching feature tensors from DRAM according to the access tensor in the input buffer. The chiplet performs feature reduction and stores the GTs in the global buffer. Then, all chiplets are configured in the **SpA mode** and initialize their CIM with weight tensors (WTs). The input buffer loads the GTs and feeds them to the CIM for computation. However, the MCM-CIM system raises new challenges for beyond-NN acceleration in both the SpG and SpA modes (see Fig. 3).

1) *Challenge1 (SpG, Redundancy):* SpG has strong gather and reduction locality [2], [6], [7], [12]. For instance, accesses to a large embedding table are often located in a small group of items in recommendation systems, and there are frequently occurred combinations of items among the users. Thus, SpG involves repeated off-chip DRAM access, interchiplet access, and redundant reduction operations (e.g., repeated accessed feature f0, f2, and f3, and repeated computed f2 + f3). This will increase interchiplet bandwidth requirements and latency.
2) *Challenge2a (SpA, CIM Initialization):* During CIM initialization, irregular tensor sparsity leads to different effective multiply-accumulation (MAC) operations across CIM macros. As shown in the example, CIM0 has fewer MAC operations than CIM1. The inter-CIM workload imbalance makes different CIMs have different computation times, leading to computation resource idleness.
3) *Challenge2b (SpA, CIM Computation):* Although balanced CIM initialization can mitigate Challenge2a, there are still idle CIM subarrays during computation. Conventionally, each CIM can only process one input row at a time. However, the current input row may only activate part of the array (like CIM1 at time step 0) due to the irregular tensor sparsity, which lowers MAC utilization.

This article proposes TensorCIM as the CIM processor chiplet, with jointly optimized SpG, SpA's CIM initialization, and SpA's CIM computation. TensorCIM overcomes the above challenges of the MCM-CIM system with the following features in one chiplet.

1) For **Challenge1**, in the SpG mode, the access tensor and SpG workload are repartitioned to maximize feature locality and minimize interchiplet access. Instead of directly accessing all feature tensors from DRAM or other chiplets, we design a Redundancy-Eliminated Gathering Manager (REGM) to dynamically maintain frequently accessed features and reduction results in the CIM for better data reuse. This eliminates redundant accesses and reductions, lowering interchip bandwidth requirements.
2) For **Challenge2a**, in the SpA mode, before CIM computation, we use an Equal Operation-based CIM Initializer (EOCI) to calculate effective MAC operations based on the nonzero distribution of GT and WT, initializing CIM macros with a balanced inter-CIM workload at the subarray level.
3) For **Challenge2b**, we exploit an Input-LookAhead CIM (ILA-CIM) architecture to look ahead at future inputs to fully utilize CIM logic, keeping all CIM subarrays busy to lower computation latency.

This article is the journal extension version of our ISSCC'23 work [17], providing higher-level insights to readers with many additional materials. The rest of this article is organized as follows. Section II provides background and motivation. Section III proposes TensorCIM's overall architecture. The three architecture features are described in Sections IV–VI, respectively. Section VII presents our experimental results on the 28-nm TensorCIM processor and four-chiplet MCM-CIM system. Section VIII concludes this article.

II. Background and Motivation

A. Beyond-NN Applications

NNs such as convolutional NNs and recurrent NNs have achieved great results in intelligent tasks like image classification, speech recognition, and natural language processing [10]. The image, audio, and text data in these tasks have regular structures that are friendly to NNs. However, real-world data has more irregular structures like graphs or embedding tables in social network analysis, knowledge graphs, and recommendation systems. SpG is usually required to gather feature

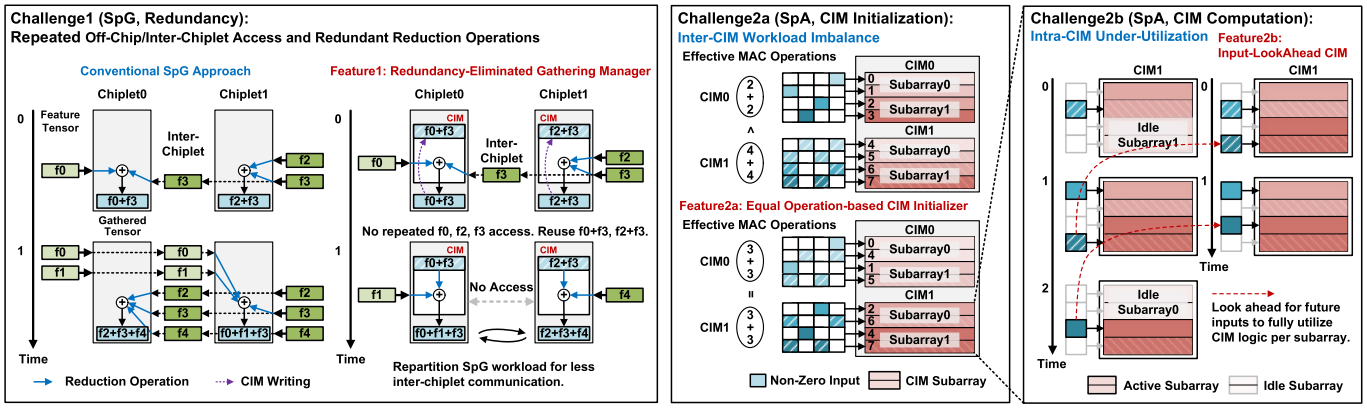


Fig. 3. MCM-CIM system raises new challenges for beyond-NN acceleration in both the SpG and SpA modes. TensorCIM has corresponding features that target the challenges: REGM, EOCI, and ILA-CIM.

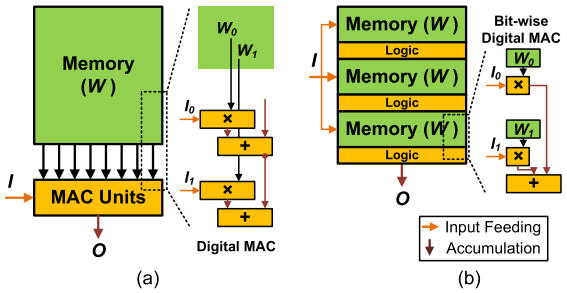


Fig. 4. AI processor architectures. (a) Digital architecture. (b) Digital CIM.

information from these irregular structures before typical NN processing, which takes sparse indices as inputs to index (ID) corresponding features and reduces them together to generate reduction results.

We can call these real-world data processing problems beyond NN applications, which have computational and data-movement requirements beyond those seen in typical NN processing. Beyond-NN applications like GCN and DLRM are widely used in more general AI scenarios [9], [22], as shown in Fig. 1. Both GCN and DLRM can be summarized as two parts: SpG and SpA. **In GCN**, the feature vector of each vertex, is computed by recursively aggregating and transforming the feature vectors of its neighbor vertices. The aggregation phase involves gathering and reducing sparsely distributed feature vectors from the feature matrix, followed by the NN-based combination phase. The aggregation phase is SpG, and the combination phase is SpA. **In DLRM**, the embedding gather operations first perform embedding table lookups for the previously interacted items. Then, embedding reduction operations perform element-wise summations of item vectors gathered from the embedding table. The reduction results are multiplied with WTs in fully connected layers to generate the final recommendation results. The embedding layer is SpG, and the fully connected layer is SpA.

B. Digital CIM

As shown in Fig. 4(a) and (b), compared to digital architectures with discrete MAC units and memory [1], [3], [11], [18], [24], digital CIM integrates bitwise digital MAC to SRAM, which has proven to be a promising architecture that achieves high efficiency and high accuracy [4], [5], [13], [16], [19], [20]. Besides, its power-performance-area (PPA) scales well with technology scaling [4], [5], [13]. The digital CIM works usually customize each SRAM cell with an additional two-input NOR for bitwise multiplication and place digital adders near the SRAM array for partial sum accumulation. In this way, digital CIM fuses the benefits of digital architectures and CIM: Integrating computing into memory achieves high efficiency; Digital in-memory logic avoids analog nonideality of conventional analog CIM and guarantees high accuracy.

C. Multichip-Module

Although CIM can efficiently reduce on-chip data movement, it is challenging to store all required data inside one chip for beyond-NN acceleration due to its large application size. Consequently, the data in CIM needs to be dumped and reloaded from the DRAM, which is a bottleneck toward high energy efficiency. This motivates the demand for scaling out CIM's capacity. MCM provides a high-yield scalable solution for CIM scaling by integrating multiple chiplets in one package [15], [28], [29]. Compared to a large monolithic die, MCMs can reduce fabrication costs, as a system can mix chiplets from different process nodes to improve design reuse. In addition, the acceleration of different scales of AI models can be achieved merely by adjusting the number of chiplets placed in a package [28].

D. Motivation of TensorCIM

The MCM-CIM system scales out digital CIM's capacity but also raises new challenges for beyond-NN acceleration. As summarized in Section I, the challenges lead to repeated feature accesses and reduction operations in the SpG mode, and low CIM utilization that harms the energy efficiency in the SpA mode. In this article, we will discuss how to overcome the challenges of the MCM-CIM system with the three features in the TensorCIM chiplet, for comprehensive beyond-NN acceleration. For SpG, the gather and reduction locality is exploited to eliminate redundant feature accesses and reductions (REGM, Feature1). For SpA, the joint

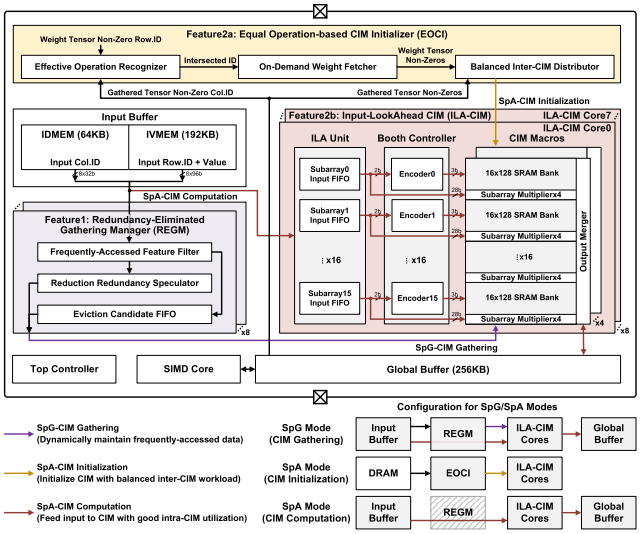


Fig. 5. TensorCIM processor's overall architecture, with three configurations for the SpG mode, the SpA mode's CIM initialization, and the SpA mode's CIM computation.

optimization of workload distribution in CIM initialization (EOCI, Feature2) and workload execution in CIM computation (ILA-CIM, Feature3) is studied to improve CIM utilization.

III. TensorCIM Processor Architecture

The TensorCIM processor's overall architecture is shown in Fig. 5, with three features highlighted in different colors. It consists of eight ILA-CIM cores with eight REGMs, an EOCI, a 256-kB input buffer, a 256-kB global buffer, a single instruction multiple data (SIMD) core, and a top controller. The arrows indicate three types of dataflows in TensorCIM: CIM gathering that dynamically maintains frequently-accessed data in the SpG mode; CIM initialization that initializes CIM with balanced inter-CIM workload in the SpA mode; CIM computation that feeds input to CIM with good intra-CIM utilization in the SpA mode.

TensorCIM has different configurations for the SpG mode, the SpA mode's CIM initialization, and the SpA mode's CIM computation. The different modes reuse the same datapath to save hardware resources. In beyond-NN acceleration, TensorCIM is first configured in the SpG mode, fetching feature tensors from DRAM according to the access tensor in the input buffer. The access tensor is also sent to the REGM to find the frequently accessed features and reduction results. Each CIM core performs feature reduction and maintains features and reduction results managed by its REGM. The GTs are stored in the global buffer. Then, TensorCIM is configured in the SpA mode. In CIM initialization, TensorCIM relies on the EOCI to initialize CIM cores with WTs that are fetched from DRAM. Meanwhile, the input buffer loads the GTs from the global buffer. In CIM computation, the input buffer feeds the GTs to ILA-CIM cores for computation. CIM outputs are written to the global buffer with activation functions performed by the SIMD core.

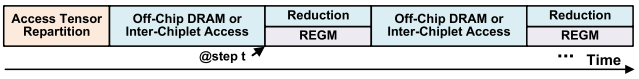


Fig. 6. Optimized SpG workflow chart. In initialization, the access tensor and SpG workload are repartitioned to minimize interchiplet access. During SpG, off-chip DRAM or interchiplet access occurs and triggers the REGM for redundancy-eliminated gathering when the required feature is not found in CIM macros.

IV. Redundancy-Eliminated Gathering Manager

A. Sparse Gathering Workflow

Fig. 6 presents the SpG workflow process. SpG fetches corresponding features in the feature tensor according to the access tensor and reduces them together to generate the GT. The access tensor records the irregular access pattern to the feature tensor. Each row represents a gather-reduce operation, and the "1" in this row indicates the required feature vector's position in this gather-reduce operation. In GCN, the access tensor is the graph adjacency matrix with each row delineating the connection of a vertex with all the other vertices in the graph. In DLRM, the access tensor is the users' past interactions with each row indicating the preferred item indices of this user.

As we have discussed in **Challenge1**, the strong gather and reduction locality leads to repeated off-chip DRAM access, interchiplet access, and redundant reduction operations in SpG. Fig. 6 contains two of our SpG workflow optimization approaches that tackle the challenge. In initialization, the access tensor and SpG workload repartition minimize interchiplet access by maximizing feature locality. During SpG, off-chip DRAM or interchiplet access occurs and triggers the REGM only when the required feature is not found in CIM macros. The main function of REGM is managing CIM to realize redundancy-eliminated gathering, which further reduces off-chip DRAM accesses, interchiplet accesses, and redundant reduction operations.

In initialization, we first reorder and repartition the access tensor for the chiplets. Features are redistributed in each chiplet's DRAM with better locality, based on the chiplet's access pattern. Then, each chiplet stores its access tensor in the input buffer and performs SpG mainly on the local DRAM with minimal interchiplet access. Fig. 7(a) uses an example to explain the benefits of access tensor and SpG workload repartition. The different orange-colored blocks represent different chiplets' feature accesses to their local DRAM. The four chiplets' local DRAM contain f0 and f1, f2 and f3, f4 and f5, f6 and f7, respectively. For Chiplet0, the access to f0 and f1 belongs to the local DRAM access, and the access to f3 belongs to the remote interchiplet accesses. The use of 2×2 orange-colored blocks around the diagonal is to highlight the data locality after access tensor reordering. Before reordering [see Fig. 7(a) (left)], most feature accesses are outside the orange blocks. In other words, most feature accesses require remote interchiplet access, which results in high feature access latency. After reordering [see Fig. 7(a) (right)], most feature accesses occur within the orange blocks, which are confined to the local DRAM of each chiplet. By comparing the elements outside orange blocks before and

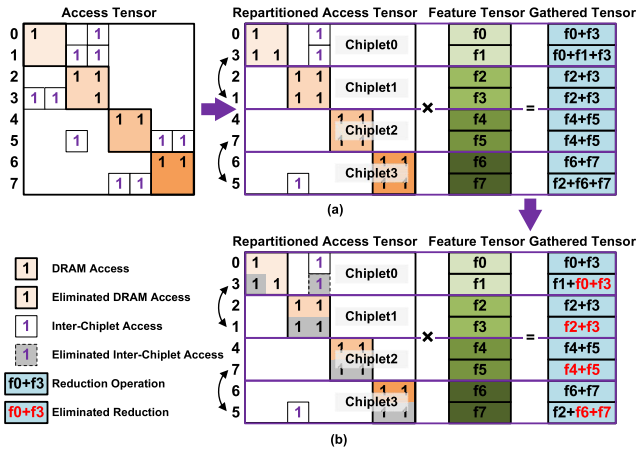


Fig. 7. Optimized SpG workflow example. (a) Access tensor and SpG workload repartition minimize interchiplet access by maximizing feature locality. (b) Redundancy-eliminated gathering via CIM further reduces off-chip DRAM accesses, interchiplet accesses, and redundant reduction operations.

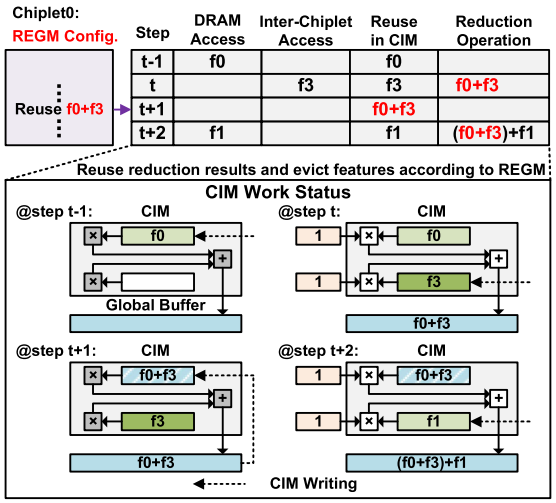


Fig. 8. Redundancy-eliminated gathering via CIM: a case study on Chiplet0.

after reordering, the effects of access tensor and SpG workload repartition can be clearly demonstrated. Only f2 and f3 are fetched through remote interchiplet accesses. Therefore, the interchiplet accesses are greatly reduced.

During SpG, due to the large tensor size like GCN's huge graph, we store frequently accessed features and reduction results in the CIM for reuse. As shown in Fig. 7(b), in this way, most features do not need to be accessed from DRAM or other chiplets, and future feature reductions can reuse previously stored results (like f0 + f3 to f6 + f7) to eliminate redundant reductions. Fig. 8 also provides more details about Chiplet0's behaviors to explain the process of redundancy-eliminated gathering via CIM. REGM manages CIM to update storage when new feature tensors are gathered onto the current chiplet. According to the reduction reuse configuration generated from REGM, Chiplet0 stores frequently-accessed features f0, f1, f3, and high-reuse reduction result f0 + f3 in CIM macros.

Generally, to reorder a random access tensor that is more complicated than the case in Fig. 7, we use a locality-enhancing reordering method. Both GCN and DLRM

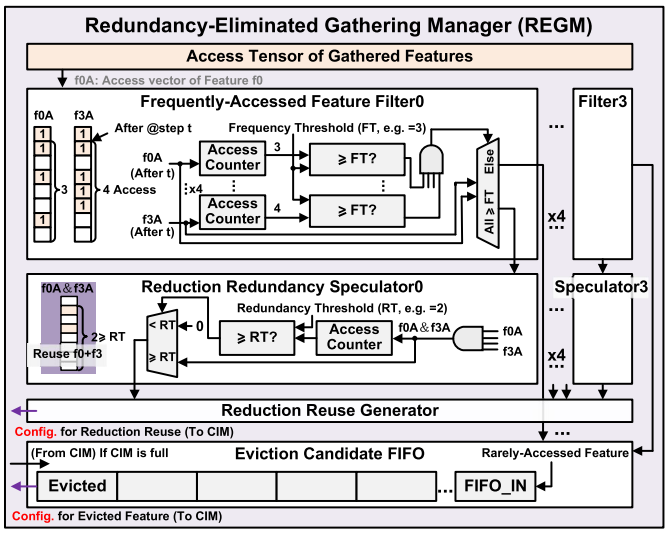


Fig. 9. REGM architecture.

have the property that a cluster of gather-reduce operations have similar feature access patterns. For instance, in GCN, the graph usually exhibits a strong internal structure, like clusters of webpages within the same domain in a web graph, or communities of common friends in a social graph. Vertices within the same community are usually accessed together by other vertices in the community, exhibiting strong feature access locality, with minimal intercommunity accesses. In DLRM, one category of users with similar characteristics tends to prefer similar types of items, with few interactions with other item types, which also shows strong locality. Therefore, our locality-enhancing reordering method assemblies a cluster of gather–reduce operations nearby in the access tensor to enhance the feature access locality. Then, we pack clusters to form four groups with a similar total number of gather–reduce operations and assign each group to one chiplet. In this way, the SpG workload assigned to each chiplet is the group of clusters, with a strong locality within each cluster. Thus, each chiplet mainly relies on its local DRAM to perform SpG, with minimal interchiplet access.

B. REGM Architecture

After discussing the SpG workflow, we introduce the REGM architecture and explain how the proposed workflow is implemented in hardware, as illustrated in Fig. 9. REGM comprises frequently accessed feature filters, reduction redundancy speculators, a reduction reuse generator, and an eviction candidate first-in-first-out (FIFO). The determination of frequently accessed features and reduction results is based on the current status, necessitating REGM's dynamic management for gathering.

The gathered features' access tensor (e.g., f0A and f3A) is loaded into the REGM and sent to frequently accessed feature filters. The future access times are counted and compared with a frequency threshold. Features with high access times are kept in the CIM. Rarely accessed features' row addresses are pushed into the eviction candidate FIFO. Since each feature's row address has only 8 bits, its storage overhead is negligible compared to the 256 × 512 SRAM size in an ILA-CIM core,

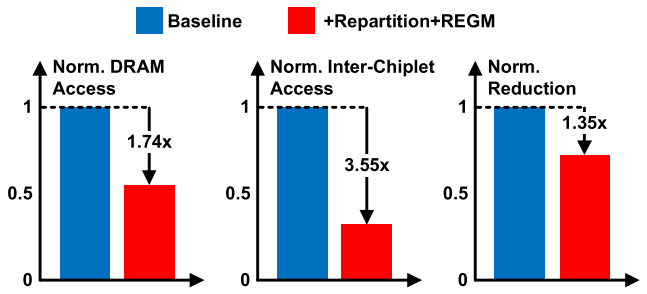


Fig. 10. Evaluation for the access tensor repartition and REGM, on GCN model's layer1 with the Pubmed dataset. Baseline: Conventional SpG approach without access tensor repartition and REGM.

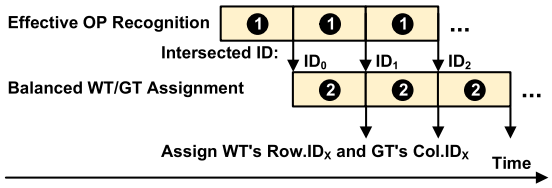


Fig. 11. Equal operation-based CIM initialization workflow chart. EOCI first finds the intersections between GT and WT's nonzero ID sequences. For each new intersected ID, a new workload assignment for the intersected WT row and GT column is triggered.

consuming only 0.3% in the area and 0.2% in the power of the chip. Besides, REGM implements reduction redundancy speculators to determine whether to store a reduction into CIM. The reduction's future reuse times are counted and compared with a redundancy threshold. In the example, f0 + f3 is predicted to be high-reuse and will be stored in CIM. After frequently accessed accesses and reduction results are determined, REGM generates configurations for reduction reuse and evicted features to CIM. If the CIM macros are full, the head feature in the eviction candidate FIFO will be popped out.

C. Technique Evaluation

Fig. 10 shows the evaluation for the access tensor and SpG workload repartition and REGM architecture. The benchmark is the aggregation phase on the GCN model's layer1 with the Pubmed dataset. The baseline (blue bar) is the conventional SpG approach without repartition and REGM. SpG workload is partitioned into four chiplets based on the access tensor's index order, and features in CIM are maintained without REGM's redundancy elimination. Compared with the baseline, our SpG optimization (red bar) achieves 1.74× fewer DRAM accesses, 3.55× fewer interchiplet accesses, and 1.35× fewer reductions on the four-chiplet system owing to the better exploitation of gather and reduction locality. Each chiplet performs SpG mainly on the local DRAM, and frequently accessed features and reduction results are maintained in the CIM.

V. Equal Operation-Based CIM Initializer

A. Equal Operation-Based CIM Initialization

As mentioned in **Challenge2a**, different effective MAC operations across CIM macros lead to inter-CIM workload imbalance in SpA. Fig. 11 illustrates the equal operation-based CIM initialization workflow that balances inter-CIM workload for SpA. EOCI performs CIM initialization for WT and GT assignment before SpA computation. First, EOCI finds the intersections between GT and WT's nonzero ID sequences, since effective MAC operations only exist between intersected columns and rows. Then, EOCI calculates the intersection's MAC operation count and reorders the GT and WT to balance operations for all CIM subarrays.

Fig. 12 presents a simple example of equal operation-based CIM initialization. The first step is effective operation recognition. The GT's nonzero column IDs are 0, 1, 2, 3, 4, and 7. The WT's nonzero row IDs are 0, 4, 5, 6, and 7. We find the intersections between the two ID sequences (e.g., ID0, 4, 7 marked on the left). An intersection implies effective MACs exist between the corresponding GT column and WT row, and nonintersected IDs will not produce effective MACs.

The second step is balanced WT and GT assignments. For each new intersected ID, a WT and GT assignment is triggered. The operation count of the current intersected ID is obtained based on the nonzero number in the GT column and WT row. Rather than the index order, the WT row can be assigned to an arbitrary subarray to achieve inter-CIM workload balance at the subarray level. As shown on the right, we fill up the CIM with the WT's nonzero values in Row0, 10, 4, and 13 for Core0, and Row7, 15, 9, and 16 for Core1. With corresponding GT columns assigned to the input buffer, the MAC operation count is the same for all subarrays.

Fig. 13 presents the CIM-adaptive WT mapping process, where the whole effective WT is divided into multiple small WT tiles. Each WT tile is mapped onto CIM in turn during CIM initialization, whose process has been elaborated in Fig. 12. The size of the effective WT is larger than the CIM capacity due to the large-scale beyond-NN applications, so the whole effective WT requires multiple times to map onto CIM. During each CIM initialization, we only select the proper WT tile that suits the CIM size and store it in CIM. Each row of the WT tile has the same number of nonzero elements as the weight count in an SRAM row. After CIM computation is completed for this WT tile, we continue the mapping process by vertically traversing the whole effective WT. Once the vertical traversal is completed, we check for any remaining nonzero elements that have not been mapped. If any exist, a new round of vertical traversal begins for these elements. Using the CIM-adaptive WT mapping, although the whole effective WT may have arbitrary sparse patterns, we can always keep each SRAM row in subarray store weights from the same WT row within the current WT tile. Therefore, TensorCIM's MAC utilization can remain high no matter whether different rows of the whole effective WT have different nonzero element numbers. Even though CIM's utilization decreases under extremely high sparsity, where the nonzero elements may not be enough to fill an SRAM row, ILA-CIM's utilization still achieves 78.9%–96.7% when the sparsity ratio ranges from 10% to 90%.

B. EOCI Architecture

Next, we go deeper into EOCI's detailed architecture to explain how it works to support the above two steps, as shown

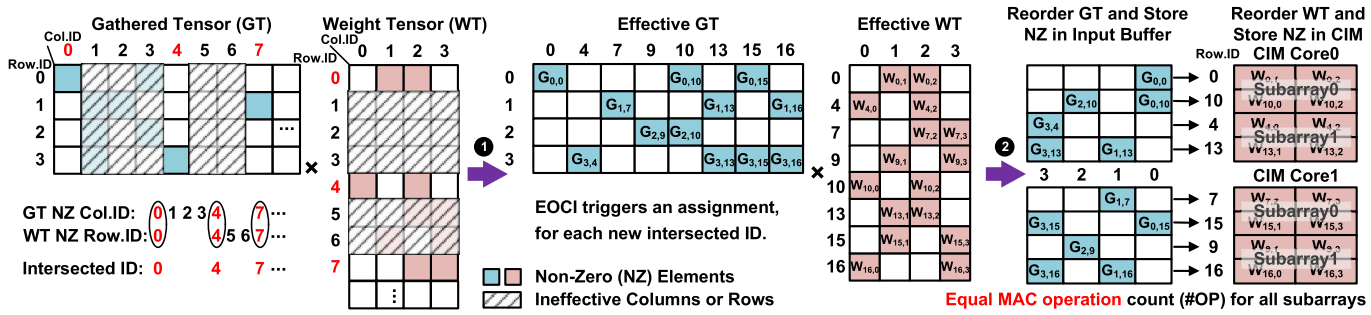


Fig. 12. Equal operation-based CIM initialization example: ❶ effective OP recognition and ❷ balanced WT/GT assignment.

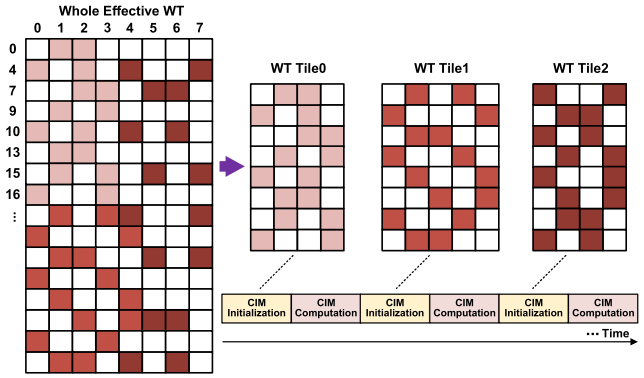


Fig. 13. CIM-adaptive WT mapping. The whole effective WT is divided into multiple small WT tiles and mapped onto CIM during each CIM initialization.

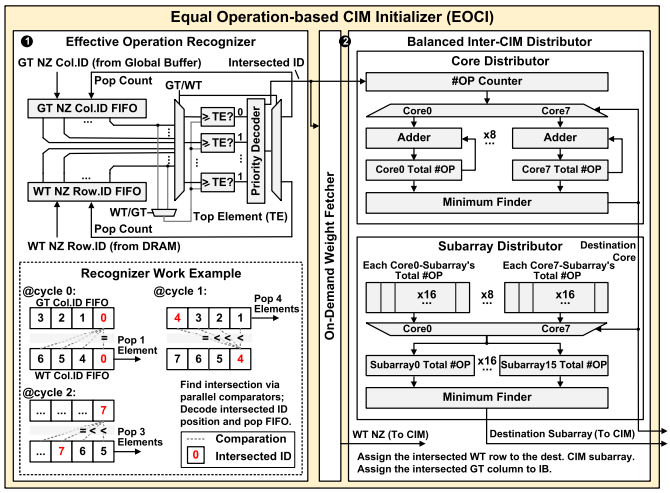


Fig. 14. EOCI architecture: ❶ effective OP recognition and ❷ balanced WT/GT assignment.

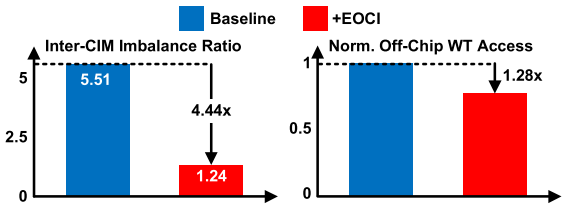


Fig. 15. Evaluation for the EOCI, on GCN model's layer1 with the Pubmed dataset. Baseline: Conventional CIM initialization approach without EOCI. The inter-CIM imbalance ratio is the maximal subarray operation count divided by the minimal subarray operation count.

in Fig. 14. First, the effective operation recognizer reads the GT's nonzero column IDs from the global buffer and the WT's nonzero row IDs from DRAM and stores them in the two FIFOs. Then, it finds their intersections via parallel comparators. In the recognizer work example at the bottom left, at cycle 1, EOCI compares the top element in the WT FIFO with all elements in the GT FIFO, and the fourth element in the GT FIFO matches. Since 1, 2, and 3 in the GT FIFO will not have intersections with other elements in the WT FIFO, they can be directly popped to avoid unnecessary comparisons. In the next cycle, the GT FIFO's first element will be compared with all elements in the WT FIFO. This alternating process continues until all intersection IDs are found.

Then, for each new intersected ID, the EOCI triggers a new workload assignment. The input buffer loads the GT column from the global buffer. The on-demand weight fetcher loads the WT row from DRAM. Since only intersected WT rows are loaded on the chip, unnecessary off-chip accesses are avoided.

Finally, EOCI utilizes the balanced inter-CIM distributor to manage GT and WT assignments. The distributor maintains the current MAC count assigned to each subarray. After calculating the intersected ID's MAC count, the distributor first finds the least-busy core and then assigns the new workload to its least-operation subarray. The intersected WT row is written to the destination CIM subarray, and the intersected GT column is written to the corresponding position of the input buffer.

C. Technique Evaluation

Fig. 15 shows the evaluation for the EOCI architecture. The benchmark is the combination phase on the GCN model's layer1 with the Pubmed dataset. The hidden layer dimension is 64. The baseline (blue bar) is the conventional CIM initialization approach without EOCI. WT and GT are assigned in the index order, and the whole WT should be loaded on the chip. We define the inter-CIM imbalance ratio as the maximal subarray operation count divided by the minimal subarray operation count, to evaluate the imbalance degree of the distributed workloads across different CIMs. Besides, since nonintersected WT rows will not be involved in effective MAC operations, loading them on a chip incurs unnecessary off-chip accesses in the baseline. We also evaluate the off-chip WT access reduction from our on-demand weight fetcher that only loads intersected WT rows. Our EOCI's balanced

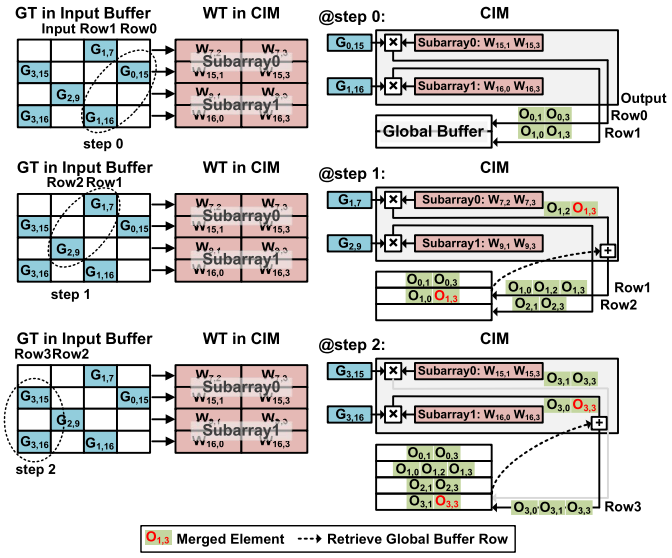


Fig. 16. ILA-CIM computation example.

inter-CIM distributor reduces the imbalance ratio from 5.51 to 1.24, and the on-demand weight fetcher avoids unnecessary WT accesses by 1.28×. This demonstrates that the EOCI can effectively balance the MAC operation count for all subarrays.

VI. Input-Lookahead CIM Core

A. Input-Lookahead CIM Computation

Although EOCI can mitigate Challenge2a, there are still idle CIM subarrays during computation that lead to intra-CIM under-utilization in SpA, as discussed in **Challenge2b**. For instance, conventionally, CIM Core1 in Fig. 12 can only process one input row at a time, so only part of the array can be activated. ILA-CIM looks ahead at future inputs to fully utilize intra-CIM MAC logic, which reduces idle CIM subarrays and lowers computation latency.

We begin with a simple example of ILA-CIM computation, as shown in Fig. 16. Instead of processing one input row at a time, ILA-CIM provides the opportunity to process multiple input rows simultaneously and keep all subarrays busy. In step 0, we look ahead for the input Row1's G_1, 16 to activate both Subarray0 and Subarray1. Based on the input's Col.ID (15 and 16), one row of each subarray is read out and multiplied by the nonzero input. Then, the output Row0 and Row1 are written to the global buffer.

Similarly, step 1 reads G_1, 7 and looks ahead for G_2, 9 to activate all subarrays, and the output Row1 and Row2 are written to the global buffer. Since the output $O_{1,3}$'s partial sum has been stored in the global buffer previously, it is loaded back for sparse accumulation.

Finally, step 2 reads $G_{3,15}$ and $G_{3,16}$ and finishes the computation with only three steps. Then, the output Row3 is written to the global buffer. In comparison, the conventional method requires four steps for the same computation.

B. ILA-CIM Architecture

As shown in Fig. 17, CIM computation has three steps: input lookahead, CIM multiplication, and output merging. These

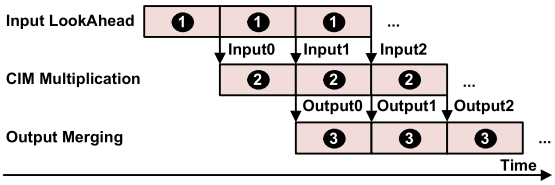


Fig. 17. ILA-CIM computation workflow chart. The three steps of input lookahead, CIM multiplication, and output merging form a pipeline in the workflow.

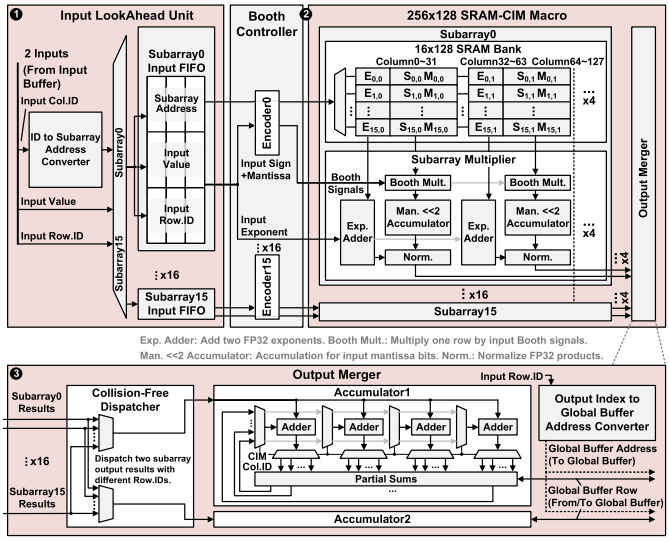


Fig. 18. ILA-CIM architecture: ❶ input lookahead, ❷ CIM multiplication, and ❸ output merging.

three steps form a pipeline in the workflow, which serves as the basis of input lookahead realization. The preparation of each subarray's inputs is the first pipeline stage, whose latency is hidden during subsequent CIM computation once the pipeline is filled. Without pipeline operation, the input preparation may take a substantial amount of time before CIM computation, offsetting the benefits from the fully utilized CIM logic.

Fig. 18 demonstrates the ILA-CIM's detailed architecture. In the first step, the input lookahead unit reads two nonzero inputs per cycle and pushes their values and IDs to the subarray FIFOs. The input lookahead unit and Booth controller are shared by the four ILA-CIM macros in each CIM core. The macro includes 16 subarrays and processes 16 required inputs together when all FIFOs are ready. Thus, the macro can look ahead for future inputs and activate all subarrays' CIM logic simultaneously.

When all FIFOs are ready, the workflow goes to the second step: CIM multiplication. Each subarray has 16 × 128 6T-SRAM cells and four Booth FP32 subarray multipliers. We use a full-digital CIM architecture with 16 rows time-sharing the subarray multipliers to balance computing accuracy and memory density [5]. The FP weights' exponent, sign, and mantissa are stored in CIM. In floating point multiplication, one row of the subarray is read out based on the input GT's Col.ID, and multiplied by the FP input mantissa's bitwise Booth signals with a ∼50% cycle reduction [16]. After shift-accumulation for input bits, the final mantissa products

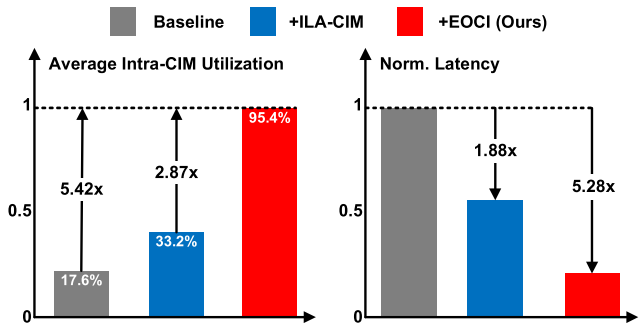


Fig. 19. Evaluation for the EOCI and ILA-CIM architecture, on GCN model's layer1 with the Pubmed dataset. Baseline: Conventional CIM initialization approach without EOCI and computation approach without ILA-CIM. "+ILA-CIM": Optimize the baseline with ILA-CIM in computation. "+EOCI" (Ours): Optimize the baseline with ILA-CIM and EOCI.

are generated. According to the sum of the input exponent and weight exponent, the FP products are normalized to the FP32 format, facilitating the following floating point additions.

Finally, we go to the third step: output merging. The output merger collects all the results from subarrays and converts the output indexes to global buffer addresses. The previous partial sums are loaded back from the global buffer and added up with the current CIM outputs. Then the accumulated outputs are written back to the global buffer.

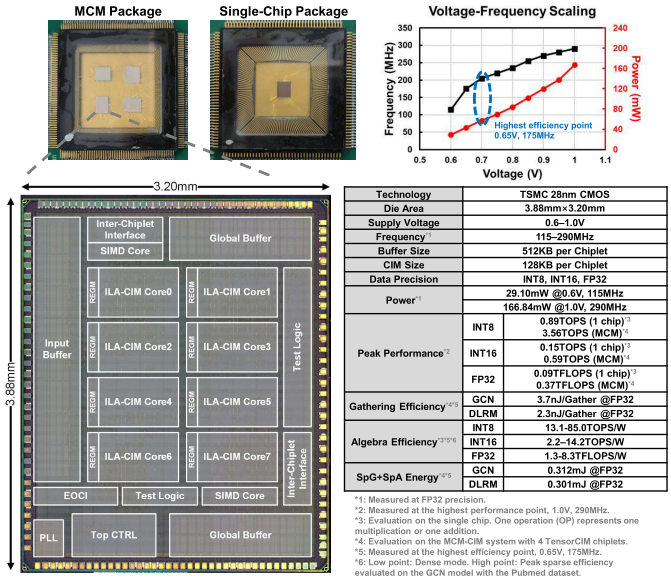
Besides, the ILA-CIM is also applicable to SpG's reduction operations. Since the feature tensor is also sparse, we can reuse the same datapath in the ILA-CIM to support the reduction of sparse feature tensors, eliminating the need for loading zeros in feature tensors onto the chip.

C. Technique Evaluation

Fig. 19 shows the evaluation for the two features in SpA optimization. The benchmark is the combination phase on the GCN model's layer1 with the Pubmed dataset. The hidden layer dimension is 64. The baseline (gray bar) is the conventional CIM initialization approach without EOCI and the computation approach without ILA-CIM. WT and GT are assigned in the index order, and each CIM core can only process one input row at a time. The ILA-CIM architecture improves intra-CIM utilization from 17.6% to 33.2% and brings 1.88× speedup. EOCI further improves intra-CIM utilization to 95.4% with a 5.28× total speedup. EOCI balances the MAC operation count for all subarrays from the workload distribution side. ILA-CIM keeps all CIM subarrays busy from the workload execution side. Therefore, the computation time of different subarrays is nearly the same, achieving high intra-CIM utilization and minimizing computation resource idleness. EOCI only incurs 1.96% area overhead, and the additional logic added to CIM for ILA support only incurs 2.64% area overhead.

VII. Experimental Results

This section first describes TensorCIM's measurement results from a 28-nm fabricated chip and four-chiplet MCM-CIM system. We then provide the test platform setup for evaluation. We conduct comprehensive studies on the TensorCIM processor with typical beyond-NN applications (GCN and DLRM), including overall evaluation, layerwise analysis, and technique breakdown analysis. Finally, we compare TensorCIM with state-of-the-art works.



Technology	TSMC 28nm CMOS	
Die Area	3.88mm×3.20mm	
Supply Voltage	0.6–1.0V	
Frequency*1	115–290MHz	
Buffer Size	512KB per Chiplet	
CIM Size	128KB per Chiplet	
Data Precision	INT8, INT16, FP32	
Power*1	29.10mW @0.6V, 115MHz	
	166.84mW @1.0V, 290MHz	
Peak Performance*2	INT8	0.89TOPS (1 chip)*3 3.56TOPS (MCM)*4
	INT16	0.15TOPS (1 chip)*3 0.59TOPS (MCM)*4
	FP32	0.09TFLOPS (1 chip)*3 0.37TFLOPS (MCM)*4
Gathering Efficiency*4*5	GCN	3.7nJ/Gather @FP32
	DLRM	2.3nJ/Gather @FP32
Algebra Efficiency*3*5*6	INT8	13.1-85.0TOPS/W
	INT16	2.2–14.2TOPS/W
	FP32	1.3-8.3TFLOPS/W
SpG+SpA Energy*4*5	GCN	0.312mJ @FP32
	DLRM	0.301mJ @FP32

*1: Measured at FP32 precision.
*2: Measured at the highest performance point, 1.0V, 290MHz.
*3: Evaluation on the single chip. One operation (OP) represents one multiplication or one addition.
*4: Evaluation on the MCM-CIM system with 4 TensorCIM chiplets.
*5: Measured at the highest efficiency point, 0.65V, 175MHz.
*6: Low point: Dense mode. High point: Peak sparse efficiency evaluated on the GCN model with the Pubmed dataset.

Fig. 20. TensorCIM's MCM package, single-chip package, chip micrograph, VF scaling curves, and specifications.

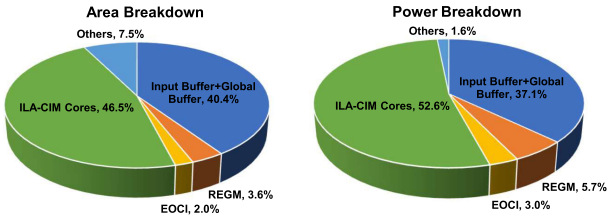


Fig. 21. TensorCIM's area and power breakdown.

A. Chip Micrograph and Summary

Fig. 20 presents the TensorCIM's MCM package, single-chip package, chip micrograph, voltage–frequency scaling curves, and specifications. TensorCIM is fabricated in 28-nm CMOS technology, with a 3.88 × 3.20 mm^2 die area. The chip can work at 0.6–1.0-V supply, 115–290 MHz with power consumption of 29.10–166.84 mW. The MCM package integrates four TensorCIM processors with 320-Mb/s interchiplet bandwidth, which is built to verify the effectiveness of the proposed techniques in solving the MCM-CIM system's challenges. TensorCIM can support INT8, INT16, and FP32 precision. INT support is realized by bypassing exponent-related processing in the ILA-CIM, which enhances TensorCIM's applicability to other NN-only applications. To support the INT8 operation, three INT8 weights are concatenated and stored in the mantissa part of the original FP32 weight, while the exponent part is gated. To support INT16 operation, one INT16 weight is stored in the mantissa part, while the other storage is gated. The mantissa alignment logic in output merger's adders is also bypassed.

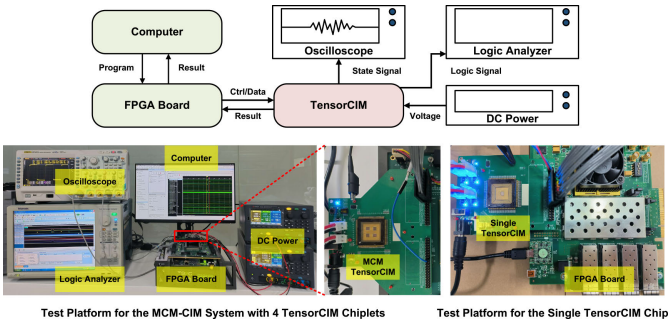


Fig. 22. Test platforms for the MCM-CIM system with four TensorCIM chiplets and the single TensorCIM chip.

For the single chip, the peak performance reaches 0.89 TOPS at INT8, 0.15 TOPS at INT16, and 0.09 TFLOPS at FP32 under 1.0 V, 290 MHz. For the four-chiplet MCM-CIM system, the peak performance reaches 3.56 TOPS at INT8, 0.59 TOPS at INT16, and 0.37 TFLOPS at FP32 under 1.0 V, 290 MHz. The highest algebra energy efficiency is 85.0 TOPS/W at IN8, 14.2 TOPS/W at INT16, and 8.3 TFLOPS/W at FP32 under 0.65 V, 175 MHz, which is measured on the sparse GCN model with the Pubmed dataset. The gathering efficiency under this operation point is 3.7 nJ/Gather at FP32. The peak performance, calculated by dividing the actual completed operations by latency, is the dense throughput that is consistent across various executed models. The peak energy efficiency, calculated by dividing the equivalent operations of nonsparse models by the energy of actually executed sparse models [27], is the sparse efficiency related to the executed model. In other words, zeros in WTs and GTs are skipped during actual execution, but these useless operations are counted in the operation number when calculating energy efficiency. This calculation method is consistent with previous works' settings [25], [26], [27], [28].

Although the main focus of TensorCIM is beyond-NN acceleration that requires FP32 precision, TensorCIM's INT efficiency is also remarkable thanks to the efficient support for sparse tensor multiplication. With less latency required for completing an MAC operation, INT8 and INT16 can deliver higher performance and energy efficiency than that of FP32. For instance, comparing INT16 and FP32, the weight counts contained in CIM are equivalent to them since the FP32 weight's mantissa part can only store one INT16 weight. In the shift-accumulation process for input bits, FP32 mantissa requires 25-b input [21] and INT16 only requires 16-b input. Therefore, INT16 requires less latency to complete an operation, and INT16's metrics are better than FP32's.

TensorCIM's area and power breakdown are depicted in Fig. 21. REGM is attached to each ILA-CIM core for dynamic gathering management, consuming only 3.6% in area and 5.7% in power. EOCI manages ILA-CIM cores' initialization, which consumes only 2.0% in area and 3.0% in power.

B. Test Platform Setup

Fig. 22 illustrates the test platform for the MCM-CIM system with four TensorCIM chiplets and a single TensorCIM chip. Besides the test MCM-CIM system or single chip, the test platform comprises Xilinx VC709 FPGA board, dc power, logic analyzer, oscilloscope, and host computer. The test dataset, beyond-NN application model, and intermediate data produced during computation are stored in the FPGA board's DDR3 memory. The FPGA sends control signals and data to TensorCIM via the FPGA mezzanine connector (FMC). The dc power supplies 0.6–1.0-V core voltage for the test chip. The oscilloscope and logic analyzer observe the chip's state signals and logic signals. TensorCIM's computing results are collected by the FPGA and then sent to the host computer for further analysis.

TABLE I

OVERALL EVALUATION ON TYPICAL BEYOND-NN APPLICATIONS[*1]

Task	Graph-based Learning	Recommendation
Dataset	Pubmed	MovieLens
Model	GCN	DLRM
Data Precision[*2]	FP32	FP32
Execution Time	2.11ms	2.07ms
Gathering Efficiency[*3]	3.7nJ/Gather	2.3nJ/Gather
Algebra Efficiency[*3*4]	7.6TFLOPS/W	6.4TFLOPS/W
Total Energy Saving[*3*5]	4.58x	3.52x

[*1]: Tested on the MCM-CIM system with 4 TensorCIM chiplets. Measured at 0.65V, 175MHz for high-efficiency evaluation.
[*2]: Work at FP32 to keep no accuracy loss for beyond-NN applications.
[*3]: Energy excludes off-chip memory to focus on the MCM evaluation.
[*4]: One operation (OP) represents one multiplication or one addition.
[*5]: The baseline is the same MCM but without the proposed techniques.

TABLE II

LAYERWISE ANALYSIS FOR GCN ON THE PUBMED DATASET

Layer Description	Latency (µs)	Energy (µJ)
Layer1 Aggregation Phase	541.547	72.480
Layer1 Combination Phase	989.454	161.080
Layer2 Aggregation Phase	537.904	71.993
Layer2 Combination Phase	39.822	6.483
Total	**2.109ms**	**0.312mJ**

C. Evaluation on Beyond-NN Applications

1) Overall Evaluation: Table I presents the overall evaluation of typical beyond-NN applications, GCN with the Pubmed dataset, and DLRM with the MovieLens dataset. Since they require high-precision computation, we use FP32 to maintain accuracy. All the results are obtained at 0.65 V, 175 MHz for the highest-efficiency measurement. Off-chip memory is excluded to focus on the MCM evaluation. The baseline is the same MCM-CIM system with four CIM chiplets, each of which is implemented on TensorCIM without the proposed techniques. To measure the energy consumption of SpG, we use a metric "gathering efficiency," which equals the average energy consumption of gathering and reducing tensors to generate one reduction result. Algebra efficiency is the energy efficiency of sparse tensor multiplication in SpA. On GCN, our work achieves 3.7-nJ/Gather gathering efficiency and 7.6-TFLOPS/W algebra efficiency, saving 4.58× energy over the baseline for the entire model. On DLRM, our work achieves 2.3-nJ/Gather gathering efficiency and 6.4-TFLOPS/W algebra efficiency, saving 3.52× energy for the entire model.

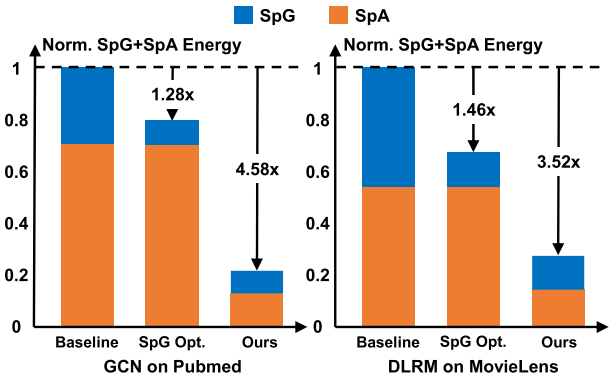


Fig. 23. Technique breakdown analysis on GCN with the Pubmed dataset and DLRM with the MovieLens dataset. "Baseline": The same MCM but without the proposed techniques. "SpG Opt.": Baseline + SpG Optimization (Feature1). "Ours": Baseline + SpG, SpA Optimization (Feature1, 2, 3). The baseline results are based on postsynthesis simulation, evaluated at 28 nm, 0.65 V, 175 MHz.

2) Layerwise Analysis: Tables II and III provide layerwise analysis on GCN and DLRM at 0.65 V, 175 MHz. For GCN, the model has two layers, with hidden layer dimension = 64. Since each layer comprises an aggregation phase and a combination phase, we separate the latency and energy for each of them. The aggregation phase (SpG) takes up 51.2% in latency and 46.3% in energy. The combination phase (SpA) takes up 48.8% in latency and 53.7% in energy. SpG and SpA have comparable latency and energy consumption proportions, necessitating the optimization for each of them. In the combination phase, as Layer2's output dimension is the label count (=3), which is much smaller than Layer1's output dimension (hidden layer dimension = 64), the Layer2 combination phase only consumes little latency and energy. For DLRM, the fully connected layers comprise a bottom-FC layer and a top-FC layer, which process dense inputs and compute the final prediction score for the recommendation task, respectively. Therefore, we report evaluation results for each of them. The embedding layer (SpG) consumes 59.1% in latency and 54.4% in energy. The fully connected layer (SpA) consumes 40.9% in latency and 45.6% in energy. SpG and SpA in DLRM also have comparable overheads.

3) Technique Breakdown Analysis: Fig. 23 demonstrates a technique breakdown analysis on GCN with the Pubmed dataset and DLRM with the MovieLens dataset, to clearly illustrate how the energy consumption is saved with TensorCIM's three features. The baseline is the same MCM-CIM system but without the proposed techniques. Thus, the four chiplets in the baseline have repeated feature accesses, redundant reduction operations, and low algebra efficiency. We then enhance the baseline with SpG optimization (Feature1). The eliminated redundant accesses and reductions lower the feature access latency. Therefore, the SpG energy reduces by 3.55× for GCN and 3.41× for DLRM, contributing to 1.28× and 1.46× energy saving for GCN and DLRM's entire model. Our TensorCIM further enhances the baseline with SpA optimization (Feature2 and 3). The optimized workload distribution (Feature2) and workload execution (Feature3) balance the MAC operation count and keep all subarrays busy, largely improving the CIM utilization of sparse tensor multiplication and algebra efficiency. In this way, the SpA energy reduces by 5.95× for GCN and 4.28× for DLRM, and the entire saving reaches 4.58× for GCN and 3.52× for DLRM.

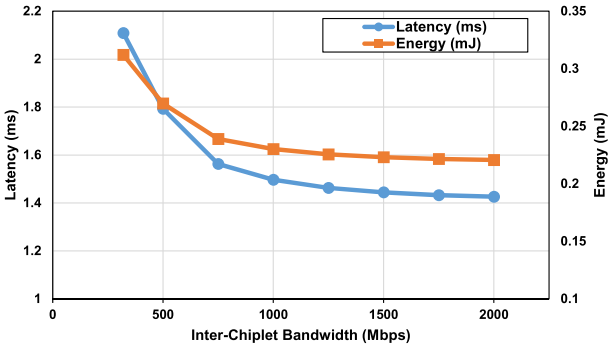


Fig. 24. Latency and energy scaling with increasing interchiplet bandwidth from 320 Mb/s to 2 Gb/s, on GCN model with the Pubmed dataset.

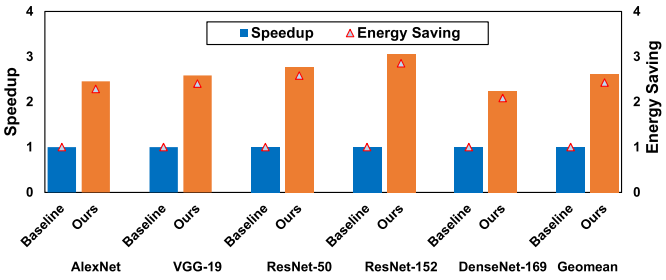


Fig. 25. TensorCIM's speedup and energy saving over the baseline on typical NNs.

TABLE III
LAYERWISE ANALYSIS FOR DLRM ON THE MOVIELENS DATASET

Layer Description	Latency (μs)	Energy (μJ)
Embedding Layer	1222.436	163.610
Bottom-FC Layer	418.667	68.158
Top-FC Layer	427.389	69.578
Total	**2.069ms**	**0.301mJ**

4) Latency and Energy Scaling Analysis: Fig. 24 presents the analysis of how latency and energy scale with changes in the interchiplet bandwidth. The implemented four-chiplet MCM-CIM system works at 0.65 V, 175 MHz with 320-Mb/s interchiplet bandwidth. We build a simulation system that scales the interchiplet bandwidth from 320 Mb/s to 2 Gb/s. Each chiplet has the same settings as the measurement results on the implemented MCM-CIM system, operating at the highest efficiency point (0.65 V, 175 MHz). The benchmark is the GCN model with the Pubmed dataset. With the interchiplet bandwidth increasing from 320 Mb/s to 2 Gb/s, the latency decreases from 2.109 to 1.426 s, and the system energy consumption decreases from 0.312 to 0.221 mJ. The curves indicate limited latency and energy scaling with increasing interchiplet bandwidth. This is mainly because the proposed MCM access optimization technique (Repartition + REGM) reduces requirements for interchiplet bandwidth, so increasing beyond 1 Gb/s gets saturated latency and energy.

TABLE IV
COMPARISON WITH STATE-OF-THE-ART CIM PROCESSORS

	ISSCC'22-11.6 [5]	ISSCC'22-15.3 [28]	ISSCC'22-15.5 [16]	This Work
CIM Implementation	Digital CIM	Analog CIM	Digital CIM	Digital CIM
Data Precision	INT4/8	INT3/4	INT8/16, BF16, ✓FP32	INT8/16, ✓FP32
Work Mode	Dense Algebra*1	Dense Algebra*1	Dense Algebra*1	✓Sparse Gathering/Sparse Algebra
MCM Access Optimization	×	×Direct Access	×	✓Reduce Access by Locality
Technology	5nm	28nm	28nm	28nm
Supply Voltage (V)	0.5∼0.9	0.63∼0.95	0.6∼1.0	0.6∼1.0
Frequency (MHz)	360∼1440	30∼105	50∼220	115∼290
Die Area (mm^2)*2	0.0133 (macro)	N/A	6.69	12.42
Power (mW)*2	18.7*3	29.5	12.5∼69.4	29.1∼166.8
Peak Performance*2	0.74	0.97	1.35 (INT8)	0.89 (1 chip), 3.56 (MCM) INT8*4
(TOPS or TFLOPS)	N/A	N/A	0.14 (FP32)	0.09 (1 chip), 0.37 (MCM) FP32*4
Algebra Efficiency*2	39.1 (INT8)*3	32.9 (INT4/3)	36.5 (INT8)	13.1∼85.0 (INT8)*5*6
(TOPS/W or TFLOPS/W)	No FP precision	No FP precision	3.7 (FP32)	1.3∼8.3 (FP32)*5*6
SpG+SpA Energy*7	N/A	N/A	SpG: 35.1nJ/Gather	SpG: 3.7nJ/Gather*5
on GCN, Pubmed Dataset	(w/o FP32 support)	(w/o FP32 support)	Total: 1.76mJ (5.64x)	Total: 0.31mJ (1x)*5

*1: Prior CIM processors can exploit sparsity for power reduction, but not for cycle saving. They don't have optimization for SpG or MCM access.
*2: Evaluation on the single chip. One operation (OP) represents one multiplication or one addition.
*3: On-chip buffers are included for fair comparison. Buffer capacity is 512KB, same as TensorCIM.
*4: Measured at the highest performance point, 1.0V, 290MHz.
*5: Measured at the highest efficiency point, 0.65V, 175MHz.
*6: Low point: Dense mode. High point: Peak Sparse efficiency evaluated on the GCN model with the Pubmed dataset.
*7: Evaluation on the MCM-CIM system with 4 chiplets. Prior work's results are based on simulation.

D. Evaluation on NN-Only Applications

Although TensorCIM mainly targets beyond-NN applications, its efficient support for sparse tensor multiplication from the EOCI (Feature2) and ILA-CIM (Feature3) is also applicable to NN-only applications. Fig. 25 shows TensorCIM's speedup and normalized energy over the baseline on typical NNs, including AlexNet, VGG-19, ResNet-50, ResNet-152, and DenseNet-169. The baseline is the same MCM-CIM system but without the proposed EOCI and ILA-CIM techniques for SpA optimization. Although the baseline only stores nonzero weight elements in CIM and feeds nonzero input elements to CIM for computation, the inter-CIM workload imbalance and intra-CIM under-utilization lead to significant computation resource idleness. Such inefficiencies severely limit the available speedup from leveraging sparsity, indicating that the irregular sparsity is quite difficult to exploit on CIM, which has a rigid crossbar structure. In contrast, with the joint optimization of workload distribution (Feature2) and workload execution (Feature3), TensorCIM realizes efficient sparse tensor multiplication with high CIM utilization. As a result, TensorCIM delivers 2.61× average speedup and 2.43× energy saving on five benchmarks.

E. Comparison With State-of-the-Art Works

Table IV shows a detailed comparison with state-of-the-art CIM processors. ISSCC'22-11.6 [5] is a digital CIM processor designed by TSMC, with only INT support. ISSCC'22-15.3 [28] is a state-of-the-art MCM-CIM based on analog CIM, with INT4/3 support. ISSCC'22-15.5 [16] is a digital CIM processor with both FP and INT support. TensorCIM is designed on the promising digital CIM-based architecture to support high-precision (up to FP32), featuring optimized SpG/SpA support for beyond-NN applications with redundancy-eliminated MCM access. TensorCIM's peak algebra energy efficiency reaches 85.0 TOPS/W at INT8 (2.17× over [5]) and 8.3 TFLOPS/W at FP32 (2.24× over [16]) at 0.65 V, 175 MHz, mainly due to the better sparsity exploitation. Since [5] and [16] can only support dense tensor multiplication, a substantial amount of zeros in WTs and GTs incur useless operations. TensorCIM effectively skips zeros and supports sparse tensor multiplication in CIM with high intra-CIM utilization. Besides the better sparsity exploitation in SpA, TensorCIM's advantages over [16] also come from the optimized SpG support. The support for sparse feature reduction in ILA-CIM avoids the overhead in loading zeros in feature tensors (discussed in Section VI-B). The eliminated redundant accesses and reductions further lower the feature access latency. Owing to the optimized SpG support, TensorCIM consumes only 3.7-nJ/Gather gathering efficiency, which is 9.49× lower than a simulated MCM-CIM system based on the digital CIM that supports FP32 [16]. With the joint SpG/SpA optimization, the total energy consumption for GCN on Pubmed is 0.31 mJ, which is 5.64× lower than [16]. Reference [28] is a state-of-the-art MCM-CIM based on analog CIM. Its INT4/3 precision is not suitable for beyond-NN applications. Moreover, its direct MCM access may cause high communication during SpG.

VIII. CONCLUSION

This work proposes TensorCIM as the CIM processor, with optimized SpG/SpA support. The SpG optimization eliminates redundant feature accesses and reductions. The SpA optimization balances the MAC operation count and keeps all

subarrays busy. While most previous research only focuses on CIM-based NN processor design, TensorCIM explores a new direction of acceleration for beyond-NN applications, which are widely used in more general AI scenarios when processing real-world data. Besides, the efficient support for sparse tensor multiplication largely enhances digital CIM's capability, paving the way for digital CIM's deployment in more general sparse tensor operations like scientific computing and graph processing. Furthermore, as AI models are getting larger and more complex, scaling out CIM processors becomes increasingly crucial [23]. The fusion of CIM and MCM presents a promising scalable approach for CIM scaling, essential for more future large-scale NN and beyond-NN applications.

REFERENCES

[1] A. Agrawal et al., "A 7 nm 4-core ai chip with 25.6 TFLOPS hybrid FP8 training, 102.4 TOPS INT4 inference and workload-aware throttling," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 64, Feb. 2021, pp. 144–146.

[2] C. Chen, K. Li, Y. Li, and X. Zou, "ReGNN: A redundancy-eliminated graph neural networks accelerator," in *Proc. IEEE Int. Symp. High-Perform. Comput. Architecture (HPCA)*, Apr. 2022, pp. 429–443.

[3] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[4] Y.-D. Chih et al., "An 89TOPS/W and 16.3TOPS/mm^2 all-digital SRAM-based full-precision compute-in memory macro in 22 nm for machine-learning edge applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2021, pp. 252–254.

[5] H. Fujiwara et al., "A 5-nm 254-TOPS/W 221-TOPS/mm^2 fully-digital computing-in-memory macro supporting wide-range dynamic-voltage-frequency scaling and simultaneous MAC and write operations," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 65, Feb. 2022, pp. 1–3.

[6] T. Geng et al., "I-GCN: A graph convolutional network accelerator with runtime locality enhancement through islandization," in *Proc. 54th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2021, pp. 1051–1063.

[7] H. Kal, S. Lee, G. Ko, and W. W. Ro, "SPACE: Locality-aware processing in heterogeneous memory for personalized recommendations," in *Proc. ACM/IEEE 48th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2021, pp. 679–691.

[8] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–14.

[9] Y. Kwon, Y. Lee, and M. Rhu, "TensorDIMM: A practical near-memory processing architecture for embeddings and tensor operations in deep learning," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, Columbus, OH, USA, Oct. 2019, pp. 740–753.

[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 7553.

[11] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: A 50.6TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 218–220.

[12] Y. Lee et al., "MERCI: Efficient embedding reduction on commodity hardware via sub-query memoization," in *Proc. 26th ACM Int. Conf. Architectural Support Program. Lang. Operating Syst.* New York, NY, USA: Association for Computing Machinery, Apr. 2021, pp. 302–313.

[13] H. Mori et al., "A 4 nm 6163-TOPS/W/b 4790-TOPS/mm^2/b SRAM based digital-computing-in-memory macro supporting bit-width flexibility and simultaneous MAC and weight update," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2023, pp. 132–134.

[14] M. Naumov et al., "Deep learning recommendation model for personalization and recommendation systems," 2019, *arXiv:1906.00091*.

[15] Y. S. Shao et al., "Simba: Scaling deep-learning inference with multi-chip-module-based architecture," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2019, pp. 14–27.

[16] F. Tu et al., "A 28 nm 29.2 TFLOPS/W BF16 and 36.5 TOPS/W INT8 reconfigurable digital CIM processor with unified FP/INT pipeline and bitwise in-memory booth multiplication for cloud deep learning acceleration," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 65, Feb. 2022, pp. 1–3.

[17] F. Tu et al., "TensorCIM: A 28 nm 3.7nJ/gather and 8.3TFLOPS/W FP32 digital-CIM tensor processor for MCM-CIM-based beyond-NN acceleration," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2023, pp. 254–256.

[18] F. Tu et al., "Evolver: A deep learning processor with on-device quantization-voltage-frequency tuning," *IEEE J. Solid-State Circuits*, vol. 56, no. 2, pp. 658–673, Feb. 2021.

[19] F. Tu et al., "A 28 nm 15.59 μJ/token full-digital bitline-transpose CIM-based sparse transformer accelerator with pipeline/parallel reconfigurable modes," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 65, Feb. 2022, pp. 466–468.

[20] F. Tu et al., "MuITCIM: A 28nm 2.24 μJ/token attention-token-bit hybrid sparse digital CIM-based accelerator for multimodal transformers," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2023, pp. 248–250.

[21] F. Tu et al., "ReDCIM: Reconfigurable digital computing-in-memory processor with unified FP/INT pipeline for cloud AI acceleration," *IEEE J. Solid-State Circuits*, vol. 58, no. 1, pp. 243–255, Jan. 2023.

[22] M. Yan et al., "HyGCN: A GCN accelerator with hybrid architecture," in *Proc. IEEE Int. Symp. High Perform. Comput. Architecture (HPCA)*, Feb. 2020, pp. 15–29.

[23] B. Yang, J. Chen, and F. Tu, "Towards efficient generative AI and beyond-AI computing: New trends on ISSCC 2024 machine learning accelerators," *J. Semicond.*, vol. 45, no. 4, Apr. 2024, Art. no. 040204.

[24] S. Yin et al., "A high energy efficient reconfigurable hybrid neural network processor for deep learning applications," *IEEE J. Solid-State Circuits*, vol. 53, no. 4, pp. 968–982, Apr. 2018.

[25] J. Yue et al., "A 2.75-to-75.9 TOPS/W computing-in-memory NN processor supporting set-associate block-wise zero skipping and ping-pong CIM with simultaneous computation and weight updating," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2021, pp. 238–240.

[26] J. Yue et al., "A 28 nm 16.9–300TOPS/W computing-in-memory processor supporting floating-point NN inference/training with intensive-CIM sparse-digital architecture," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, 2023, pp. 1–3.

[27] J. Yue et al., "A 65 nm computing-in-memory-based CNN processor with 2.9-to-35.8TOPS/W system energy efficiency using dynamic-sparsity performance-scaling architecture and energy-efficient inter/intra-macro data reuse," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 234–236.

[28] H. Zhu et al., "COMB-MCM: Computing-on-memory-boundary NN processor with bipolar bitwise sparsity optimization for scalable multi-chiplet-module edge machine learning," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 65, Feb. 2022, pp. 1–3.

[29] B. Zimmer et al., "A 0.32–128 TOPS, scalable multi-chip-module-based deep neural network inference accelerator with ground-referenced signaling in 16 nm," *IEEE J. Solid-State Circuits*, vol. 55, no. 4, pp. 920–932, Apr. 2020.

Yiqi Wang (Graduate Student Member, IEEE) received the B.S. degree from the School of Integrated Circuits, Tsinghua University, Beijing, China, in 2021, where he is currently pursuing the Ph.D. degree.

His research interests include deep learning, in-memory computing, and computer architecture.

Zihan Wu received the B.S. degree from the School of Microelectronics, Shandong University, Jinan, China, in 2020. He is currently pursuing the M.S. degree with the School of Integrated Circuits, Tsinghua University, Beijing, China.

His research interests include in-memory computing, computer architecture, and deep learning.

Weiwei Wu received the B.S. degree in electronic science and technology from Beijing University of Technology, Beijing, China, in 2016, and the M.S. degree from the Institute of Microelectronics, Tsinghua University, Beijing, in 2019, where she is currently pursuing the Ph.D. degree with the School of Integrated Circuits.

Her current research interests include deep learning, computer architecture, and VLSI design.

Leibo Liu (Senior Member, IEEE) received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree from the School of Integrated Circuits, Tsinghua University, in 2004.

He is currently a Professor with the Institute of Microelectronics, Tsinghua University. His research interests include reconfigurable computing, mobile computing, and VLSI DSP.

Yang Hu (Member, IEEE) received the B.S. degree from Tianjin University, Tianjin, China, in 2007, the M.S. degree from Tsinghua University, Beijing, China, in 2011, and the Ph.D. degree from the University of Florida, Gainesville, FL, USA, in 2017.

He was an Assistant Professor with the ECE Department, The University of Texas at Dallas, Richardson, TX, USA, from 2017 to 2021. He is currently an Associate Professor with the School of Integrated Circuits, Tsinghua University.

Shaojun Wei (Fellow, IEEE) was born in Beijing, China, in 1958. He received the Ph.D. degree from the Faculte Polytechnique de Mons, Mons, Belgium, in 1991.

He is currently a Professor with the School of Integrated Circuits, Tsinghua University, Beijing. His main research interests include VLSI SoC design, EDA methodology, and communication ASIC design.

Dr. Wei is a Senior Member of the Chinese Institute of Electronics (CIE).

Fengbin Tu (Member, IEEE) received the B.S. degree from the School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing, China, in 2013, and the Ph.D. degree from the Institute of Microelectronics, Tsinghua University, Beijing, in 2019.

He was a Post-Doctoral Scholar at the Scalable Energy-efficient Architecture Laboratory (SEAL), Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA, USA, from 2019 to 2022, and a Post-Doctoral Fellow at the AI Chip Center for Emerging Smart Systems (ACCESS), Hong Kong, China, from 2022 to 2023. He is currently an Assistant Professor with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong. He has authored two books titled *Architecture Design and Memory Optimization for Neural Network Accelerators* and *Artificial Intelligence Chip Design*. His research works appeared at top conferences and journals on integrated circuits and computer architecture, including International Solid-State Circuits Conference (ISSCC), IEEE JOURNAL OF SOLID-STATE CIRCUITS (JSSC), Design Automation Conference (DAC), International Symposium on Computer Architecture (ISCA), and International Symposium on Microarchitecture (MICRO). His research interests include AI chips, computer architecture, reconfigurable computing, and computing in memory.

Dr. Tu designed the AI Chip Thinker and received the 2017 ISLPED Design Contest Award. His Ph.D. thesis was recognized by the Tsinghua Excellent Dissertation Award.

Shouyi Yin (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electronic engineering from Tsinghua University, Beijing, China, in 2000, 2002, and 2005, respectively.

He was with Imperial College London, London, U.K., as a Research Associate. He is currently a Full Professor and the Vice Director of the School of Integrated Circuits, Tsinghua University. He has authored more than 100 journal articles and more than 50 conference papers. His research interests include reconfigurable computing, AI processors, and high-level synthesis.

Dr. Yin has served as a Technical Program Committee Member for the top VLSI and EDA conferences, such as Asian Solid-State Circuits Conference (A-SSCC), International Symposium on Microarchitecture (MICRO), Design Automation Conference (DAC), International Conference on Computer-Aided Design (ICCAD), and Asia and South Pacific Design Automation Conference (ASP-DAC). He is an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I: REGULAR PAPERS (TCAS-I), *ACM Transactions on Reconfigurable Technology and Systems* (TRETS), and *Integration, the VLSI Journal*.