國立臺灣大學電機資訊學院電子工程學研究所

碩士論文

Graduate Institute of Electronics Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

利用多位元數位輸入之 14T 靜態隨機存取記憶體位元建構之可重構數位記憶體內運算加速器

A Reconfigurable Digital CIM-Based Accelerator Using 14T SRAM-Based Bit Cell with Multi-Bit Digital Inputs

吳宗諺

Tsung-Yen Wu

指導教授：劉宗德 博士

Advisor: Tsung-Te Liu, Ph.D.

中華民國 111 年 7 月

July 2022

doi:10.6342/NTU202202019

國立臺灣大學碩士學位論文
口試委員會審定書

利用多位元數位輸入之 14T 靜態隨機存取記憶體位元建構之可重構數位記憶體內運算加速器

A Reconfigurable Digital CIM-Based Accelerator Using 14T SRAM-Based Bit Cell with Multi-Bit Digital Inputs

本論文係吳宗諺君（R08943051）在國立臺灣大學電子工程學研究所完成之碩士學位論文，於民國 111 年 7 月 8 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

劉孝慈（指導教授）

楊家驤　　閻志達

系主任、所長　江介宏

doi:10.6342/NTU202202019

誌謝

首先，我要感謝劉宗德教授這段時間的指導，讓我在碩士期間學習做研究的精神，也從計畫的參與中得到許多與他人合作的寶貴經驗。不管是寫計劃書、與其他實驗室的同學共同合作或是處理實驗室內部的事情，老師給我很多機會磨練和人溝通與協調的能力。

同時我也要感謝實驗室的夥伴們。每當研究遇到瓶頸時，大家除了給我很多實質建議外，實驗室歡樂的氣氛讓我的壓力可以暫時得到釋放。我要特別感謝姚鈞嚴大師，第一次下線時他耐心地指導我，讓我熟悉整個下線的流程，使我更有經驗面對隔年下線的各種挑戰。另外也要謝謝 Benson 學長、瑋廷大學長、瀚中、育愷、欣哲和驊謙在我下線前那艱難的幾個月幫我非常多忙，讓我熬過充實的一段時光。

最後，謝謝家人在碩士這段期間的支持與鼓勵，每當我壓力山大時，他們會適時提醒我適度的放鬆。也謝謝女友在碩士期間和我征服台灣一座座的山巒，就像克服研究上一道道的難題；陪我走訪台北大街小巷的咖啡廳，度過無數個工作的周末。謝謝這三年幫助過我的每一個人，讓我在徬徨無助時給我一盞明燈，也讓我更有自信面對每次的挑戰。

doi:10.6342/NTU202202019

摘要

記憶體內運算 (computing-in-memory, CIM) 架構藉由大幅減少記憶體與處理元件之間的資料搬移量，突破傳統馮·紐曼架構的瓶頸 (von Neumann bottleneck)。然而，現有記憶體內運算巨集的容量並不足以處理大量的相乘累加運算 (multiply-accumulate operations, MAC)，且資料搬運速度時常受限於晶片輸入與輸出的頻寬，很難進一步提高整體吞吐量 (throughput) 及能量使用效率 (energy efficiency)。因此，近年來記憶體內運算電路研究領域已經拓展到系統層級，期望有效提升加速器的性能。

本論文提出一個可重構之數位記憶體內運算加速器。電路架構層面藉由採用多位元數位輸入之 14T 靜態隨機存取記憶體位元提升記憶體內運算巨集之面積使用效率。巨集架構層面則運用可重構之壓縮樹增加機器學習任務的位元寬度彈性以及應用範圍。系統架構層面則透過資料流程以及後期處理的改善，優化整體系統的硬體使用率及吞吐量。我們將所提出之設計實作在標準 28 奈米 CMOS 製程，模擬結果顯示此設計可以達到 98.30 TOPS 的系統吞吐量，21.28 TOPS/mm^2 的系統面積使用效率以及 430.50 TOPS/W 的系統能量使用效率。此外，單位 CIM 容量之吞吐量在模擬中高達 1.37 TOPS/Kb，領先目前最先進之記憶體內運算加速器。

關鍵字: *深層神經網路、記憶體內運算巨集、基於記憶體內運算之加速器、可重構性、高吞吐量、高能量使用效率、高面積使用效率。*

doi:10.6342/NTU202202019

Abstract

Computing-in-memory (CIM) architecture breaks through the von Neumann bottleneck by reducing data movement between the memory and the process elements. However, the size of the existing CIM macros is insufficient to perform a lot of multiply-accumulate operations (MAC). Also, the data transfer speed is often limited by the input and output bandwidth of the chip, making it difficult to improve the overall throughput and energy efficiency. Therefore, the CIM research extends to the system-level exploration, expecting that the performance of the accelerator can be effectively improved.

This thesis presents a reconfigurable digital CIM-based accelerator. The circuit-level feature improves the area efficiency of CIM macros by adopting the 14T SRAM bit cells with multi-bit digital inputs. The macro-level feature uses reconfigurable compressor trees to increase the bit-width flexibility and the application range of machine-learning tasks. The system-level feature optimizes the hardware utilization and the overall system throughput by improving data flow and post-processing. The proposed CIM design is implemented in standard 28nm CMOS technology. The simulation results shows that this work achieves a system throughput of 98.30 TOPS, system area efficiency of 21.28 TOPS/mm^2 and system energy efficiency of 430.50 TOPS/W. Moreover, this work achieves the highest throughput per CIM size of 1.37 TOPS/Kb in simulation compared with the state-of-the-art designs.

Keywords: *Deep neural network (DNN), computing-in-memory (CIM) macro, CIM-based accelerator, reconfigurability, high throughput, energy-efficient, area-efficient.*

doi:10.6342/NTU202202019

Contents

doi:10.6342/NTU202202019

doi:10.6342/NTU202202019

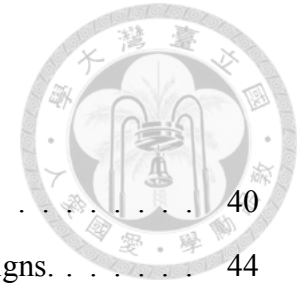
List of Figures

doi:10.6342/NTU202202019

doi:10.6342/NTU202202019

doi:10.6342/NTU202202019

List of Tables



doi:10.6342/NTU202202019

Chapter 1

Introduction

1.1 Motivation

With the rise of artificial intelligence (AI) and machine learning (ML) techniques, we can see extensive applications of many cognitive tasks such as image classification and facial recognition from cloud to edge devices. Recently, researches on neural network (NN) accelerators for edge devices have received more attention due to the requirement of higher privacy, higher energy efficiency, and lower latency. However, traditional von Neumann architectures can not meet the demands for future energy-constrained AI edge applications with many matrix-vector multiplications (MVMs). In traditional von Neumann architecture, energy efficiency is limited due to the separation of computation and storage. The frequent data movement from off-chip memory to on-chip processing cores has dominated the overall energy breakdown, as shown in Figure 1.1. This "memory wall" has become the performance bottleneck of a hardware implementation for efficient ML algorithms.

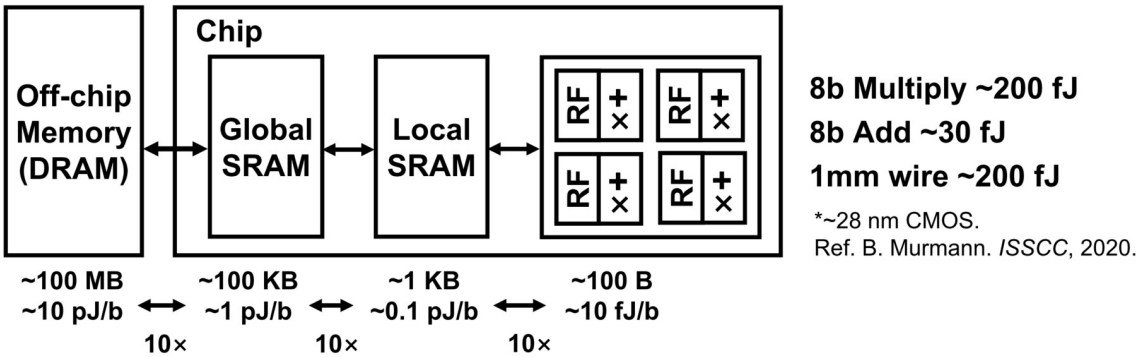
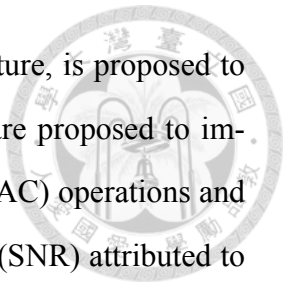


Figure 1.1: Block diagram of a general DNN accelerator [1].

doi:10.6342/NTU202202019



Computing-in-memory (CIM), an emerging computing architecture, is proposed to break through the memory wall. Early analog CIM works [2]–[4] are proposed to improve energy efficiency by enabling parallel multiply-accumulate (MAC) operations and reducing memory access. However, a degraded signal-to-noise ratio (SNR) attributed to analog CIM nonlinearity results in high accuracy loss. Consequently, adopting an analog approach to design CIM works is inappropriate for some applications that require high accuracy.

Besides, previous CIM works [2]–[11] are designed at macro level, ignoring the degraded performance of a CIM macro at system level. For instance, neural network parameters nowadays usually exceed the capacity of a CIM macro, so weight updating leads to significant performance loss. Lastly, the weight mapping method and the data flow, both of which have direct impacts on CIM macro utilization and energy efficiency, are not well explored. Based on the above issues, CIM-based accelerators for system-level solutions are on demand and continue to expand their capabilities to larger datasets and more complicated models.

1.2 Contribution

To overcome the design challenges above, we proposed a reconfigurable CIM-based accelerator integrating 36 2-Kb fully-digital CIM static random-access memory (SRAM) macros with the following design features shown in Figure 1.2:

1) The proposed 14T bit cell integrating a standard 6T SRAM and four customized NAND gates for multi-bit digital input realizes a high macro-level area efficiency with a bit-fusion input scheme.

2) The proposed reconfigurable compressor tree, supporting mixed-precision computation with signed or unsigned representation in each NN layer, can improve the energy-accuracy trade-off with various topologies and bit precisions in different NN models.

doi:10.6342/NTU202202019

3) The proposed one-shot near-memory computing blocks significantly reduce the execution time in several neural network topologies with the central-processing-unit-like (CPU-like) pipelined architecture.

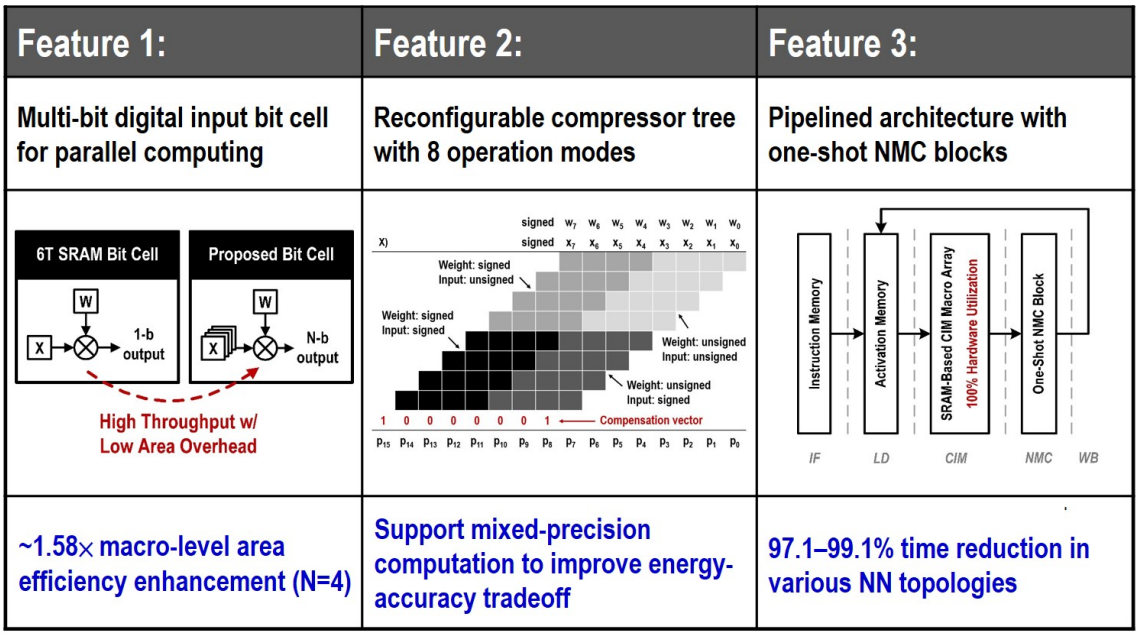


Figure 1.2: Summarized features of this design.

The proposed fully-digital CIM-based accelerator was implemented in standard 28nm complementary metal-oxide-semiconductor (CMOS) technology. The proposed 14T SRAM-based bit cell with multi-bit digital input achieves a macro area efficiency of 81.92 TOPS/mm^2 in simulation. The proposed reconfigurable compressor tree increases flexibility for various bit precision of input and weight. The simulated peak macro energy efficiency is 625.15 TOPS/W. Moreover, thanks to the CPU-like pipelined architecture and the proposed one-shot NMC blocks, the proposed design enables a 97.1%–99.1% estimated execution time reduction of convolution layers in various NN topologies. Besides, the proposed design also achieves a peak throughput of 98.3 TOPS.

doi:10.6342/NTU202202019

1.3 Thesis Organization

This thesis is organized as follows. Chapter 2 gives an overview of CIM. Chapter 3 introduces previous digital CIM macro works. Chapter 4 describes the proposed CIM macro and its operating principles. Chapter 5 introduces previous CIM-based accelerators. Chapter 6 describes our CIM-based accelerator architecture and the corresponding data flow. The chip implementation and results are shown in Chapter 7, where comparisons with the state-of-the-art results are also provided. Chapter 8 concludes the thesis.

doi:10.6342/NTU202202019

Chapter 2

CIM Overview

2.1 Conventional Computing Architecture versus CIM Architecture

Deep learning has dramatically increased the accuracy in large-scale recognition tasks recently. Nevertheless, to achieve higher accuracy, recent state-of-the-art deep learning algorithms tend to present deeper and larger network models, posing significant challenges for DNNs hardware implementations such as energy cost, computational complexity, and memory access. Many previous works reduced the energy cost via algorithm or hardware improvement. On the algorithm side, compression and pruning substantially decrease the number of nonzero parameters [12]. Moreover, quantization-aware training using low bit precision has been investigated with minimal degradation in the classification accuracy. On the hardware side, many application specific integrated circuit (ASIC) designs (e.g., Eyeriss [13]) implement the DNN algorithms into energy-efficient accelerators. Nevertheless, previous CMOS ASIC works show that "memory access" is the main bottleneck for energy-efficient computing. As shown in Figure 2.1(a), the conventional von Neumann architecture separates the processor and the memory macro. During computation, massive data movement from memory to a processor leads to high energy loss. Furthermore, even though SRAM can follow the CMOS scaling trend well, row-by-row accesses are still necessary to perform MAC operations, resulting in limited throughput and a large amount of energy consumption for read and write. Hence, the concept of CIM has been proposed in recent years to overcome the above challenges, as shown in Figure 2.1(b). Compared with conventional computation-centric ASIC works, the data-centric CIM architecture can perform parallel computation in the computational memory

doi:10.6342/NTU202202019

without row-by-row access from memory to a processor. As a result, CIM architectures achieve higher energy efficiency and throughput than von Neumann architectures owing to fewer data movement and higher on-chip bandwidth. However, it lacks bit-width flexibility and computation accuracy attributed to process-voltage-temperature (PVT) variations. In light of this, this work aims at higher computation accuracy and higher bit-width flexibility to support various ML inference topologies.

In order to achieve the above goals, choosing the appropriate design approach is crucial. Accordingly, we will first introduce the analog and digital CIM, respectively, and then compare the pros and cons of these two approaches.

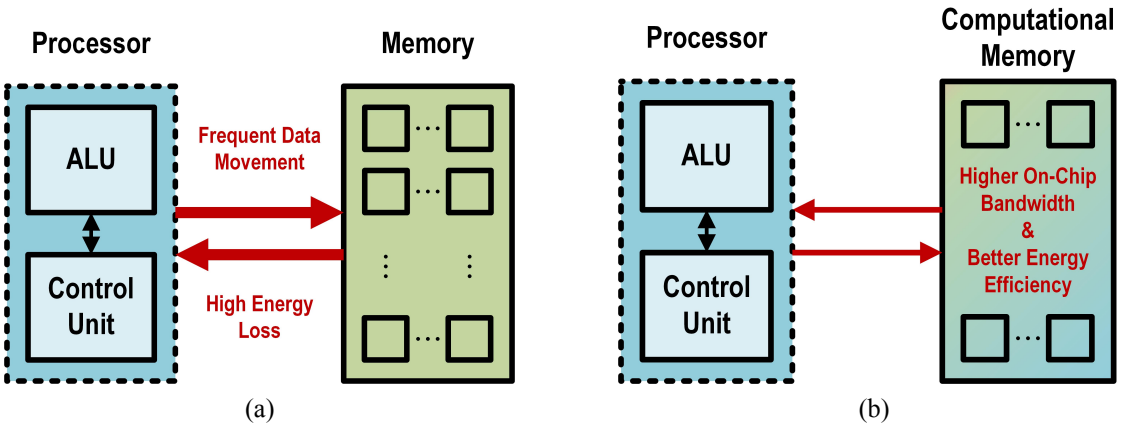


Figure 2.1: Computing architecture comparison. (a) Conventional von Neumann architecture. (b) CIM architecture.

2.2 Challenges of Analog CIM

As shown in Figure 2.2, there are several challenges in SRAM-based analog CIM architectures [4], [14]–[16]. First, BL with multi-row access in CIM macros may narrow the BL dynamic range and cause write disturbance (see Figure 2.2(a)). In addition, the nonlinear MAC results caused by PVT variations could further decrease the inference accuracy, as shown in Figure 2.2(b). The area and power overheads of digital-to-analog converters (DACs) and analog-to-digital converters (ADCs) should also be seriously considered since they are bulky and power-hungry and strictly influence the performance of a CIM macro (see Figure 2.2(c)). At last, DACs and ADCs typically operate with lower resolu-

doi:10.6342/NTU202202019

tion, so analog CIM architecture may not be suitable for high-precision edge computing applications, as shown in Figure 2.2(d).

SNR degradation is the main disadvantage of analog CIM architecture. As shown in Figure 2.3, the low SNR is attributed to the following reasons [17]:

1) Signal degradation in analog accumulation.

2) Error caused by non-ideality in DACs and ADCs.

3) Error caused by bit cell PVT variation.

4) Error amplified by a nonlinear activation function.

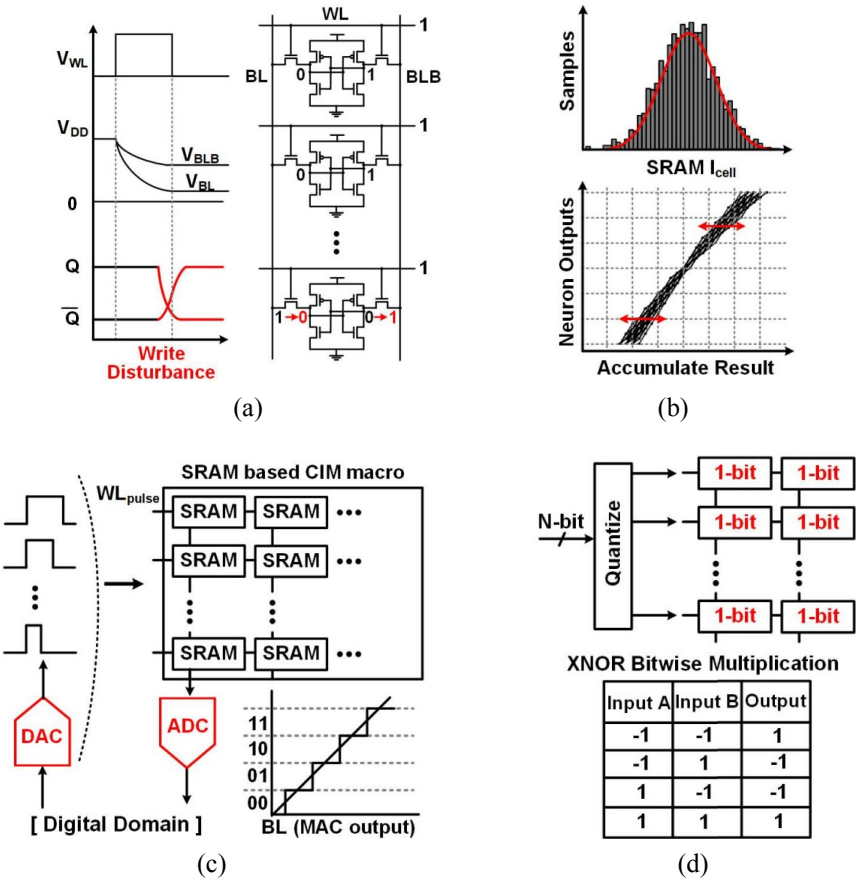


Figure 2.2: Design challenges of SRAM-based analog CIM architectures [18]. (a) Write disturbance and data-flipping. (b) Nonlinearity. (c) DACs and ADCs overhead. (d) Limited computing precision.

doi:10.6342/NTU202202019

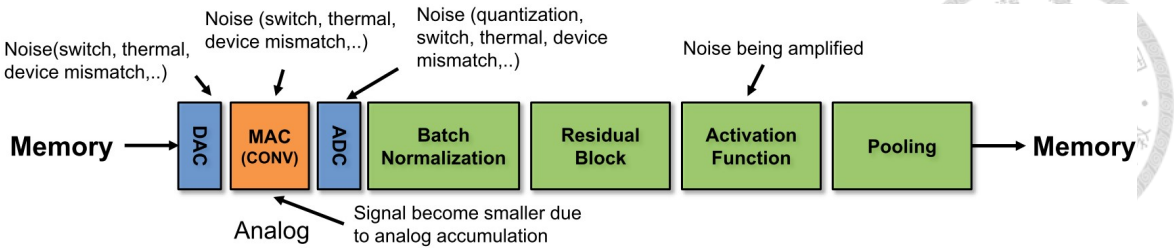


Figure 2.3: Low signal-to-noise ratio (SNR) with analog CIM [17].

Recent analog CIM macros [6], [9], [15], [16], [19] avoid the write disturbance problem with modified SRAM bit cells, which decouple the BL and the RBL for standard read/write operations and MAC operations. Nevertheless, this method increases the bit cell area, resulting in lower memory density. On the other hand, on-chip training methods [20], [21], performing self-calibration on weight offset, can be resistant to PVT variation. However, the related area and power overheads diminish the advantage of analog CIM architecture.

2.3 Analog CIM versus Digital CIM

Before starting a CIM macro design, the design challenges and the main differences between analog and digital CIM are illustrated in Figure 2.4 and summarized as follows:

1) *Analog CIM:*

   We can roughly classify this computing scheme into current-based CIMs and charge-based CIMs. Current-based CIMs are susceptible to the PVT variation and nonlinearity in the bit cell. Charge-based CIMs sacrifice the area to mitigate the variations via charge sharing for MAC operations. Unfortunately, the mixed-signal nature of both computing scheme makes analog CIM vulnerable to multiple noise sources in advanced process nodes. For example, increased coupling noise due to strict layout constraints can reduce charge-domain computation accuracy.

2) *Digital CIM:*

   Reference [22] made a comparison between digital CIM and a charge-based analog CIM based on the same 5nm technology. First, since a digital CIM macro supports

doi:10.6342/NTU202202019

higher input and weight bit-precision without SNR degradation, it can achieve better accuracy than analog CIM. Second, the digital CIM designs realize $> 1.3\times$ better energy efficiency and $> 1.4\times$ better throughput and area efficiency than analog CIM. Although the digital computing scheme omits the complexity of DACs and ADCs, it introduces more area overhead and higher power consumption owing to the considerable number of full adders.

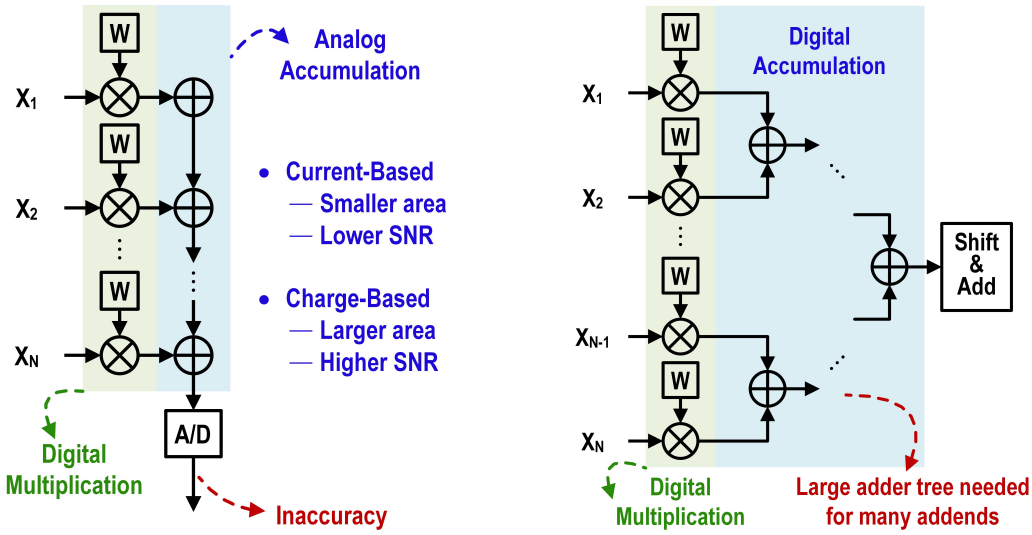


Figure 2.4: Comparison between analog and digital CIM architecture. (a) Analog CIM architecture [6], [9], [15], [16], [19]. (b) Digital CIM architecture [17], [22].

To sum up, there are many disadvantages in analog CIM designs, including nonlinearity, area and power overhead of DACs and ADCs, and the limited bit precision they can support. Therefore, we finally adopt a digital approach to design our CIM macro since it realizes no accuracy loss, achieves a better PPA and bit-width flexibility than the analog CIM counterpart, and supports simple design for testability (DFT).

2.4 From CIM Macro to CIM-Based Accelerators

Although an SRAM-based CIM macro shows promising energy efficiency at the macro level, the advantages of a CIM macro may disappear from a system perspective.

doi:10.6342/NTU202202019

There are several critical issues and missing pieces toward building a system-level CIM-based accelerator using these macros.

1) *Integration of multiple CIM SRAM macros:*

   Most CIM designs only implemented a CIM macro with a small size at the range of Kbs. The whole system needs to integrate many of these CIM macros with a top controller, leading to higher area costs for inter-macro communication.

2) *Post-processing circuits:*

   To evaluate accuracy for DNNs, these designs employ many post-processing modules, including partial sum accumulation, linear operation, ReLU, max pooling, and activation functions. These post-processing circuits also increase the additional area and power consumption at the system level.

3) *Additional hardware resources for data storage:*

   The input activations, weights, and instructions must be stored in a digital SRAM. Therefore, the area and power consumption would further increase at the system level. However, the area and power breakdown of the CIM macro design typically does not consider these overheads.

doi:10.6342/NTU202202019

Chapter 3

Related Works of Digital CIM Macro

In the previous chapters, we learned that CIM architectures can be classified into two categories according to their computing scheme: analog and digital CIM. Analog CIM may not be suitable for some applications requiring high accuracy. As a result, we further introduce some related digital CIM works.

3.1 Digital CIM Design Proposed by Kim *et al.*

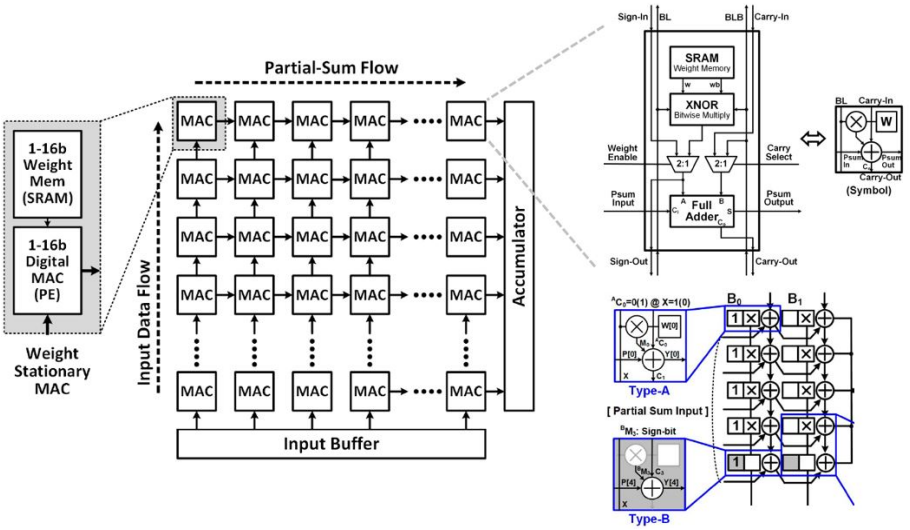


Figure 3.1: Digital CIM design proposed by Kim *et al.* [18].

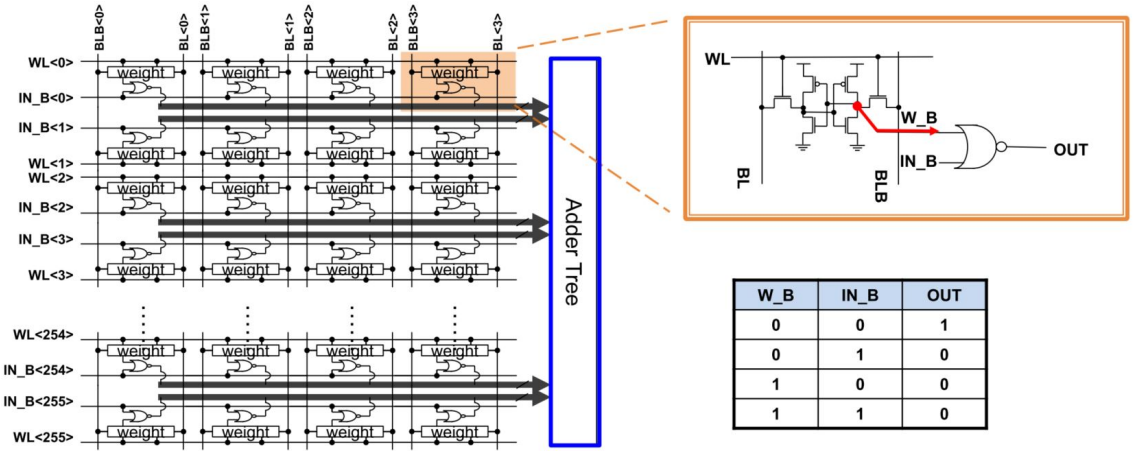
As shown in Figure 3.1, Colonnade [18], proposed by Kim *et al.*, provides an alternative fully digital approach that avoids the DACs and ADCs required in the conventional analog CIM designs. The bit cell includes a standard 6T SRAM for weight storage, an XNOR gate for multiplication, and a full adder for accumulation or sign extension. However, the full adder circuit dominates the CIM bit cell area, contributing to a larger cell than other state-of-the-art CIM works. Moreover, the throughput of the CIM macro is limited due to the bit-serial input scheme. Lastly, the CIM hardware utilization rate and area

doi:10.6342/NTU202202019

efficiency are further degraded because some bit cells are only for sign extension without activating the storage.

3.2 Digital CIM Design Proposed by Chih *et al.*

As shown in Figure 3.2, a fully-digital SRAM CIM macro, proposed by Chih *et al.* [17], supports multi-bit input and weight MAC operations with a bit-serial input scheme. It can also support various NN topologies via different configurations of multiple macros. The overall architecture is based on customized bit cell arrays (for digital multiplication) and adder trees (for partial product addition). However, the area efficiency is limited since each 10T bit cell can only produce a binary output, which is area-inefficient with one additional NOR gate. Lastly, a compact implementation of a wide adder tree is necessary because it usually dominates the area of a CIM macro.



W_B	IN_B	OUT
0	0	1
0	1	0
1	0	0
1	1	0

Figure 3.2: Digital CIM design proposed by Chih *et al.* [17].

3.3 Digital CIM Design Proposed by Fujiwara *et al.*

A 5nm fully-digital CIM macro, proposed by Fujiwara *et al.* [22], supports wide-range dynamic voltage-frequency scaling and simultaneous MAC and write operations. Figure 3.3 shows the overall architecture of the digital CIM macro, which comprises 64 CIM macro arrays with 64-Kb of 12T bit cells.

doi:10.6342/NTU202202019

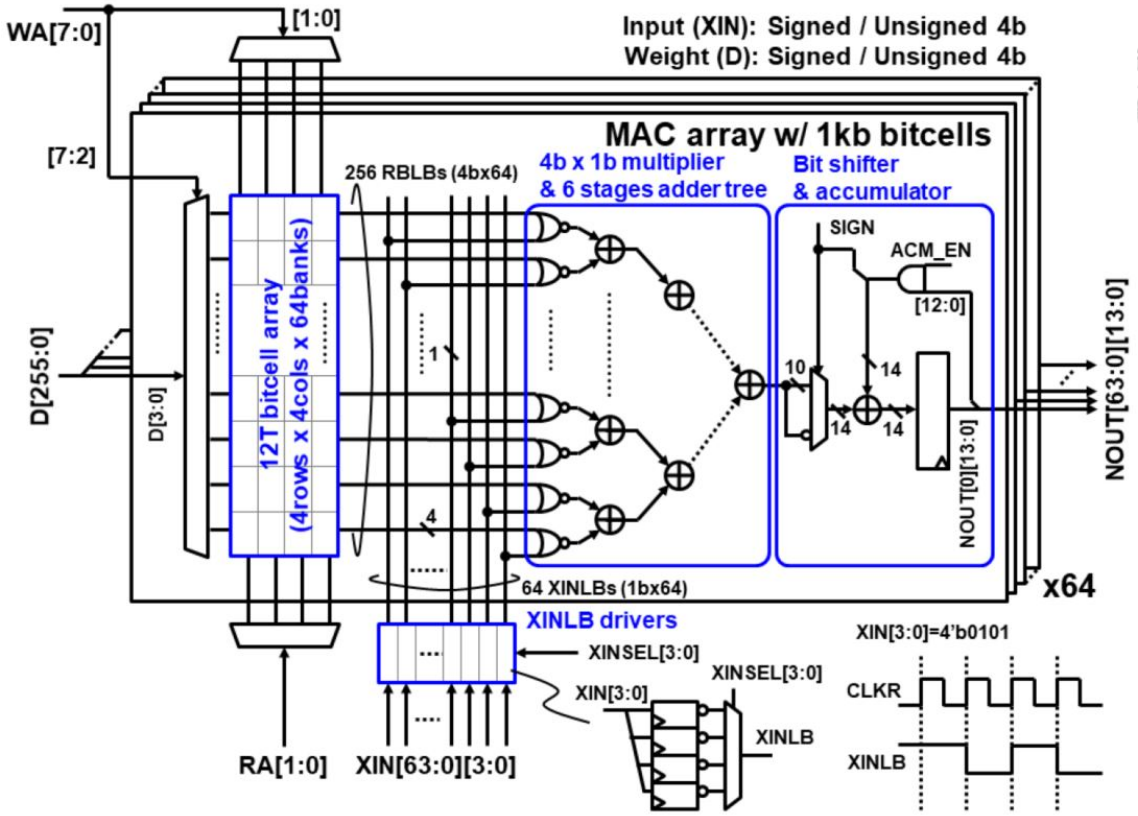
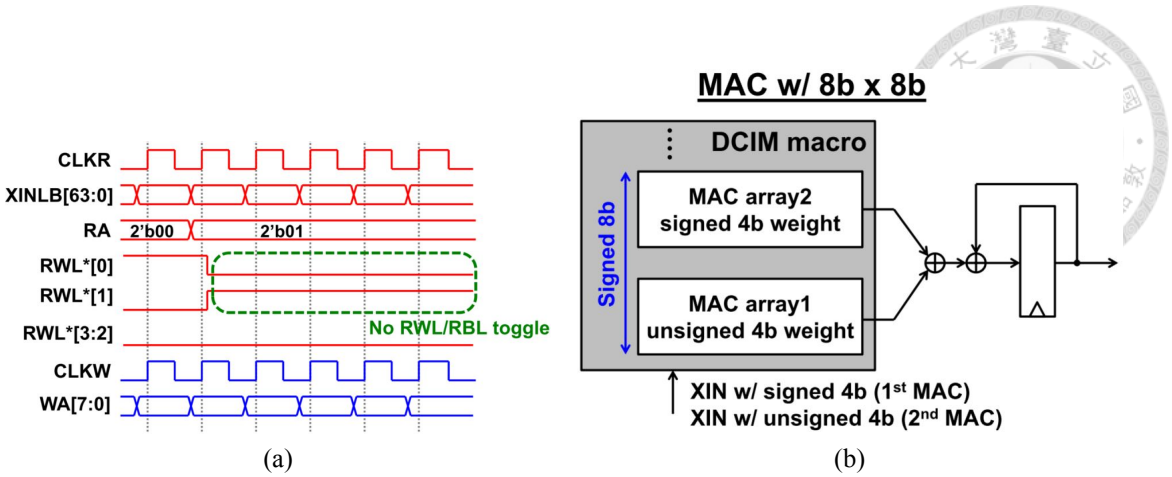


Figure 3.3: Digital CIM design proposed by Fujiwara *et al.* [22].

Figure 3.4(a) presents the timing diagram of the MAC operation. The customized 12T bit cell supports asynchronous read and write operations via independent clocks. Therefore, concurrent MAC operations and weight updating are available as long as the access addresses of write operations and MAC operations are different. Thus, this technique achieves higher system throughput because of the reduced idle weight-updating cycles.

However, a 4-to-1 multiplexer in the XINLB driver generates 1b of XIN in each clock cycle in a bit-serial manner. As a result, it needs five CLKR cycles to complete a MAC operation with 64 4-b inputs and 4-b weights since NOUT comes out one clock cycle later. However, it has the same problem as their previous design [17], that is, the throughput of the CIM macro is limited due to the bit-serial input scheme.

doi:10.6342/NTU202202019

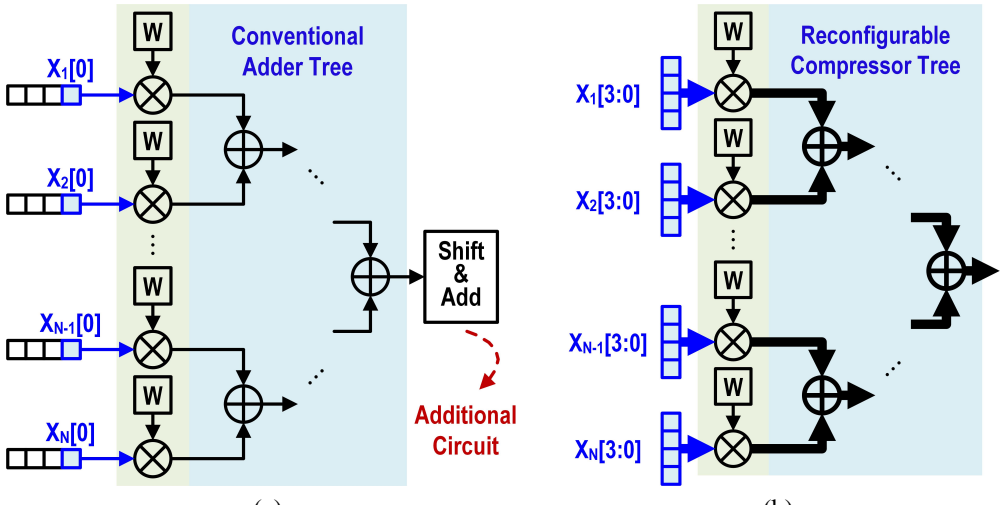


(a) (b)

Figure 3.4: (a) Simultaneous MAC and weight update. (b) Bit width flexibility.

3.4 Summary of Previous Digital CIM Works

Figure 3.5 shows the design challenges of digital CIM. In [17], [18], [22], the bit-serial input scheme leads to a low area efficiency owing to the high transistor count overhead used to perform 1-b×1-b MAC operations. On the other hand, the conventional adder trees spend many transistor counts for addition and require additional shift-add circuits to support MAC operations of multi-bit input and weight. Apart from the low area efficiency, the lack of flexibility makes the post-processing more complex. Consequently, the proposed digital CIM design aims to improve throughput, area efficiency, and flexibility in the case of different bit widths and representations of signed and unsigned numbers.



(a) (b)

Figure 3.5: (a) Schematic of the conventional digital CIM macro with bit-serial input scheme. (b) Schematic of the proposed digital CIM macro.

doi:10.6342/NTU202202019

Chapter 4

Proposed CIM Macro

4.1 CIM Macro Architecture

Figure 4.1 shows the schematic of the proposed digital CIM macro, which includes eight processing elements (PEs) and 2-Kb 14T bit cells. Each PE comprises 0.25-Kb SRAM-based bit cells for signed and unsigned weight storage and a reconfigurable compressor tree for partial product summation.

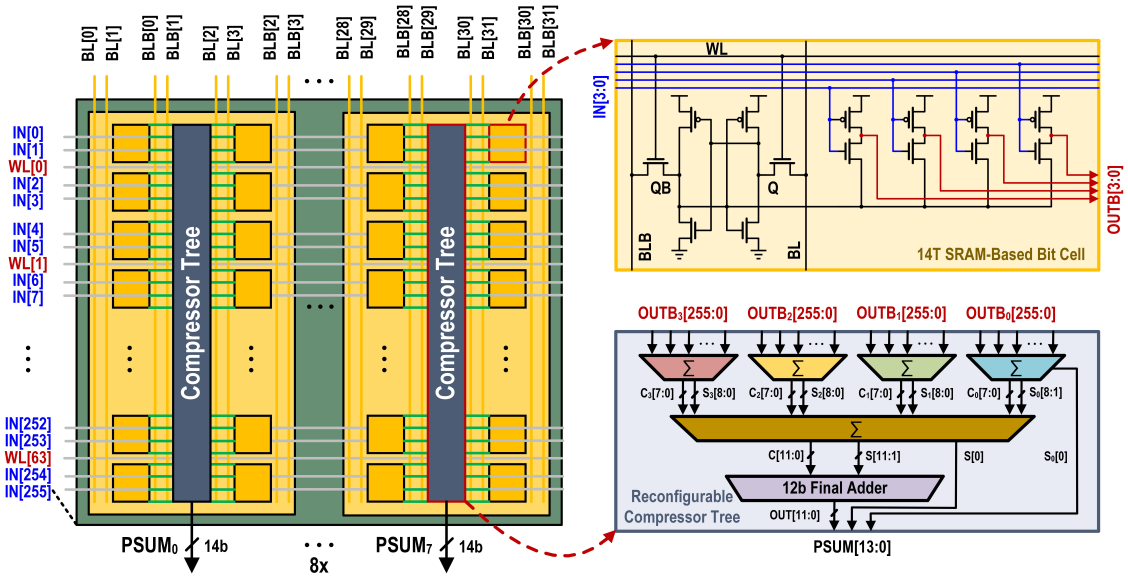


Figure 4.1: Architecture of the proposed CIM macro design.

There are two operating modes in the proposed CIM macro: standard read and write mode and CIM mode. Standard read and write mode aims to perform standard read or write, and CIM mode aims to perform MAC operations. During the CIM operation, 64 4-b inputs are parallelly sent into a CIM macro in a bit-fusion manner from LSB. Each PE performs 64 multiplications of 4-b inputs and 4-b weights in one clock cycle. Then

doi:10.6342/NTU202202019

a reconfigurable compressor tree sums up the results of 64 multiplications. Therefore, it only takes a single cycle to complete 64-dimensional 4-b $\times$ 4-b MAC operations. The operation can be expressed as

$$\mathbf{P_j} = \mathbf{x} \cdot \mathbf{w_j} = \sum_{i=0}^{63} x_i[3:0] \times w_{ij}[3:0] \tag{4.1}$$

where **x** is a 64-dimensional input vector whose bit precision in each scalar term x_i is 4, $\mathbf{w_j}$ is a 64-dimensional weight vector whose bit precision in each scalar term w_{ij} is 4, i represents the row number from 0 to 63 and j represents the PE number from 0 to 7.

4.2 Proposed 14T SRAM-Based Bit Cell

Figure 4.2 presents the detailed schematic and the NAND operation of the proposed 14T SRAM-based bit cell with 4-b digital input. Besides the 6T SRAM bit cell, four customized 2T NAND gates are used for bit-wise multiplication of 1-b weight and 4-b input activation. Assuming that QB is 0 and IN changes from 0 to 1, OUTB will start to discharge, and the voltage at QB will temporally increase and perhaps cross the write trip point. Accordingly, we performed a thousand Monte Carlo simulations and analyzed under the worst case (QB is 0 and all INs change from 0 to 1). The simulation result shows that at most 24-b input can still pass five corners at 0.6V supply voltage. QB faces heavier loads during a standard write operation if INs are 1, which leads to long latency. Thus, it is necessary to reset INs to 0 during weight updating.

After addressing the above design issues, we will present several advantages of the proposed SRAM-based bit cell architecture:

1) *High throughput:*

   The bit-fusion input scheme, with quadruple parallelization of input bit precision within an SRAM-based bit cell, increases the on-chip bandwidth compared with the bit-serial input scheme.

2) *High area efficiency:*

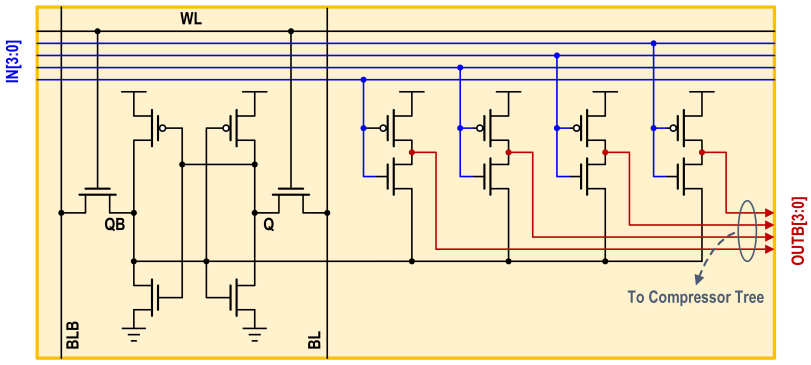
doi:10.6342/NTU202202019

Only a 2T transistor count overhead for an input bit precision is required, so the customized NAND gates integrated into the proposed 14T SRAM-based bit cell can achieve higher area efficiency.

3) *Compact layout design (see Section 7.1):*

The oxide diffusion (OD) of VDD and QB can be reused in layout design in the customized NAND gates thanks to its natural gate symmetry. Besides, a wider bit cell can fit the layout size of the compressor tree at the back-end.

The design details and the performance analyses are further described in Chapter 7.



W	QB	IN	OUTB
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0

Figure 4.2: Proposed 14T SRAM-based bit cell. (a) Schematic. (b) NAND operation.

As many modern NNs (e.g., face detection) only need 4-b and 8-b precision [23], this work aims to support operations with 4-b and 8-b precision. Figure 4.3 presents the throughput, CIM macro area, and CIM macro area efficiency with different input bit-width parallelization. Although the 8-b input bit-parallel scheme manifests the best area efficiency in an 8-b input layer, its area efficiency is 1.83× lower than the 4-b input bit-parallel scheme in a 4-b input layer. Moreover, in an 8-b input bit-parallel scheme, routing congestion will occur on IN buses since there are too many vertical metal wires (see Figure 7.2 for the detailed layout of the proposed bit cell). Therefore, we finally choose the 4-b input bit-parallel scheme as our specification.

doi:10.6342/NTU202202019

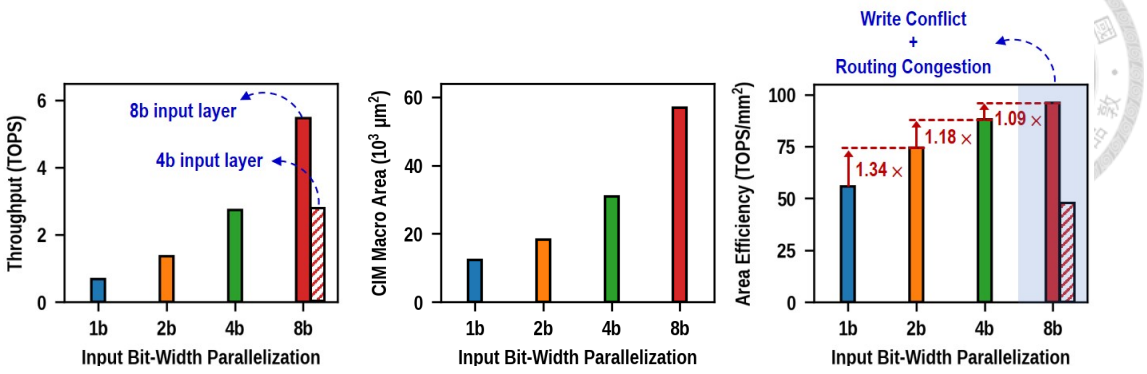


Figure 4.3: CIM macro area efficiency analysis with different input bit-width parallelization in an 8-b input layer. (a) Throughput, (b) area, and (c) area efficiency.

4.3 Reconfigurable Compressor Tree

To accumulate the partial products in a single PE, we adopt the proposed reconfigurable compressor tree instead of the conventional adder tree since it can achieve better PPA and flexibility.

Since the bit-width of the partial products are very high, it is significantly desirable to efficiently make the multiplication simultaneously support both signed and unsigned notations. Therefore, we introduce the Baugh-Wooley algorithm [24] to handle the sign bits cost-effectively. This technique has been developed to design regular multipliers suitable for two's complement representation. Let's consider a M-b signed weight (W) $\times$ N-b unsigned input (X) multiplication. W and X can be represented as

$$W = -w_{M-1} \cdot 2^{M-1} + \sum_{j=0}^{M-2} (w_j \cdot 2^j) \ , \quad X = \sum_{i=0}^{N-1} (x_i \cdot 2^i) \ , \tag{4.2}$$

where w and x represent the bit values in W and X, respectively, and w_{M-1} represents the sign bit. The full precision product, $P = W \times X$, is then given by the following equation:

doi:10.6342/NTU202202019

$$
\begin{aligned}
P &= W \times X \\
&= \left(-w_{M-1} \cdot 2^{M-1} + \sum_{j=0}^{M-2}(w_j \cdot 2^j)\right) \times \left(\sum_{i=0}^{N-1}(x_i \cdot 2^i)\right) \\
&= \sum_{j=0}^{M-2}\sum_{i=0}^{N-1} w_j x_i \cdot 2^{i+j} - \sum_{i=0}^{N-1} w_{M-1} x_i \cdot 2^{i+M-1} \\
&= \sum_{j=0}^{M-2}\sum_{i=0}^{N-1} w_j x_i \cdot 2^{i+j} + \left(\sum_{i=0}^{N-1}(\sim w_{M-1} x_i) \cdot 2^{i+M-1} + 1\right)
\end{aligned} \tag{4.3}
$$

where $\sim$ represents negation.

The first term of the above equation is unsigned multiplication, and the last term is the subtraction term. In order to calculate the product, it is feasible to sum up the two's complements instead of subtracting the last term. Figure 4.4 shows the example of the 4-b signed weight $\times$ 4-b unsigned input Baugh-Wooley multiplier.

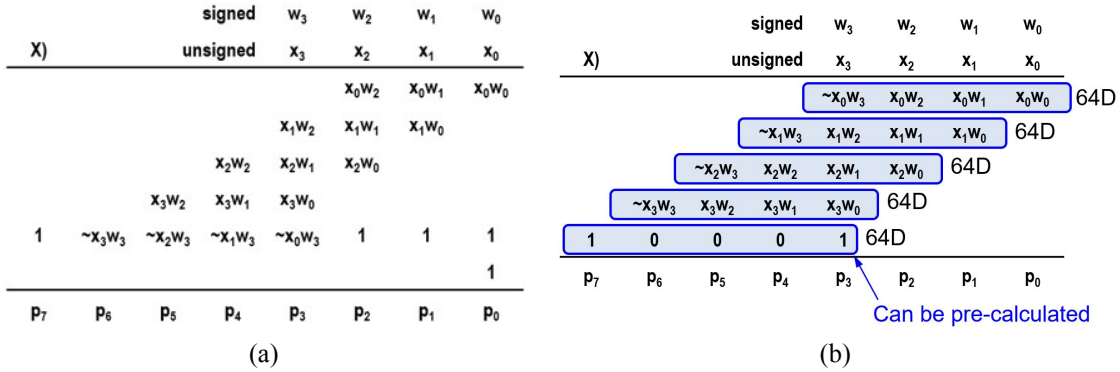
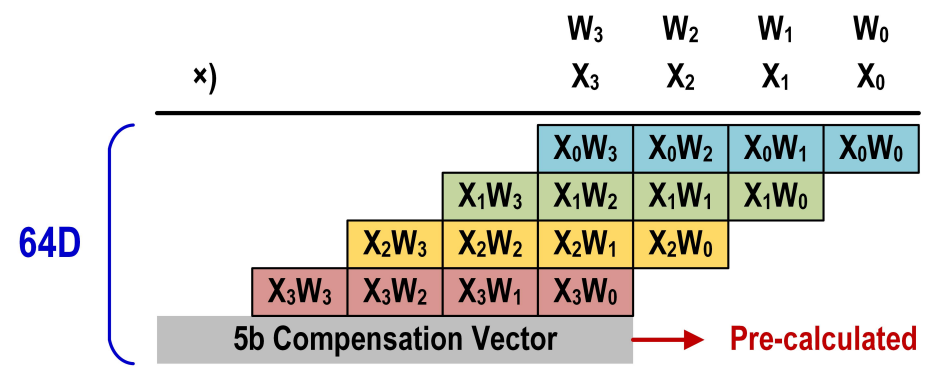


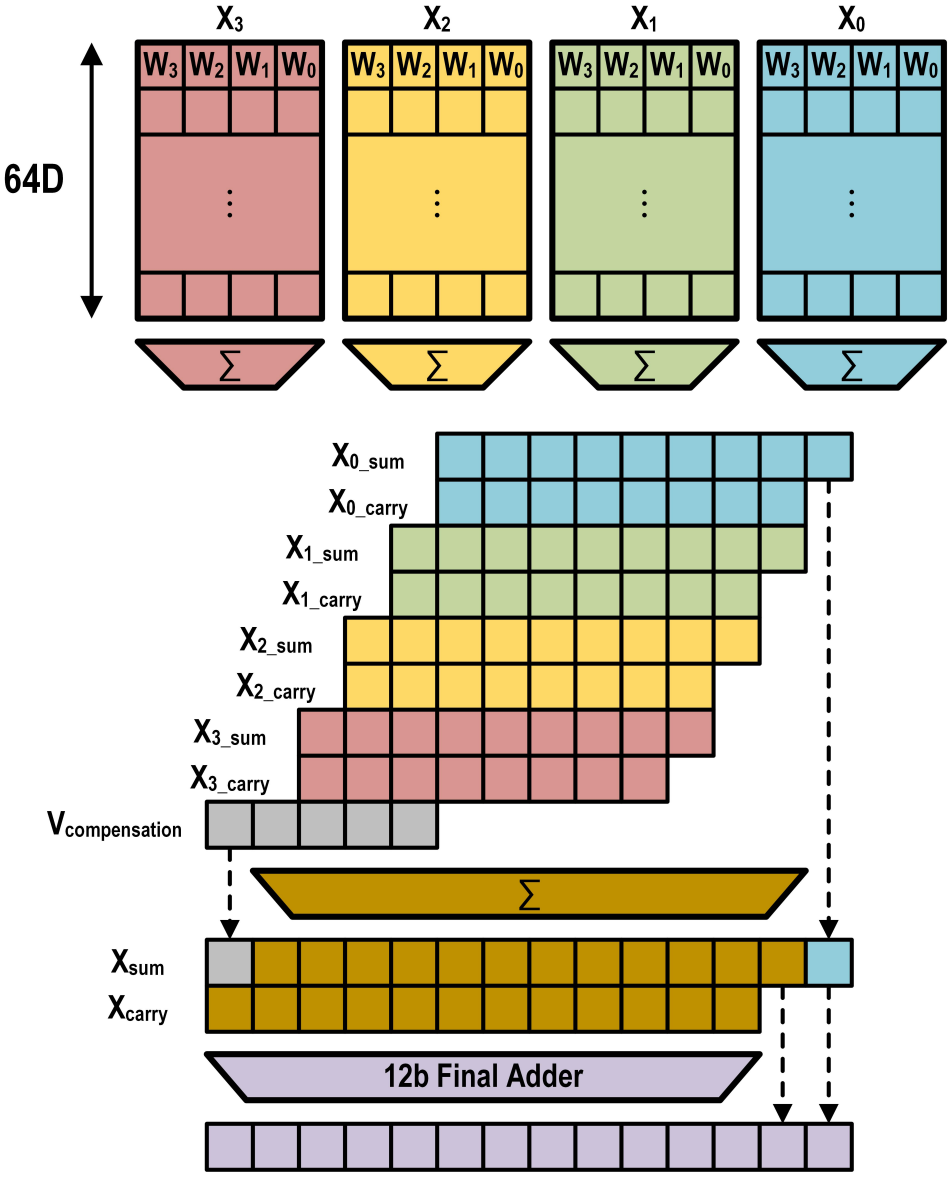
Figure 4.4: MAC operation of 4-b signed weight and 4-b unsigned input. (a) Original operation. (b) Permutated operation.

Figure 4.5 shows the schematic of the proposed reconfigurable compressor tree. The hardware for 64-dimension 4-b$\times$4-b MAC operations consists of four 64-dimension 4-b$\times$1-b TDM-based compressors [25], a TDM-based compressor for the sum of four partial MAC results and a compensation vector, and a 12-b final adder for the final MAC result. Since parallel multiplier implementation has uneven delays through a full adder, the basic concept of the TDM method is to make the delay of each path as close as possible by connecting the full adders appropriately. This method creates a compressor that globally minimizes the critical path of the multiplier.

doi:10.6342/NTU202202019



(a)



(b)

Figure 4.5: Proposed reconfigurable compressor tree. (a) Schematic of a 64-D 4-b×4-b multiplier. (b) Schematic of the operation process.

doi:10.6342/NTU202202019

4.4 Bit-Width Flexibility

Nowadays, AI accelerators need to support different numerical precision for various NN topologies. The proposed digital CIM macro supports not only 4-b×4-b MAC operations but also higher bit-precision configurations. An 8-b signed×8-b signed multiplier structure is implemented with the Baugh-Wooley algorithm and the decomposition logic, and its operation can be expressed as

$$\begin{aligned} P &= W \times X \\ &= \left(-w_{M-1} \cdot 2^{M-1} + \sum_{j=0}^{M-2}(w_j \cdot 2^j)\right)\left(-x_{N-1} \cdot 2^{N-1} + \sum_{i=0}^{N-2}(x_i \cdot 2^i)\right) \\ &= \sum_{j=0}^{M-2}\sum_{i=0}^{N-2} w_j x_i \cdot 2^{i+j} - \sum_{i=0}^{N-2} w_{M-1} x_i \cdot 2^{i+M-1} \\ &- \sum_{j=0}^{M-2} x_{N-1} w_j \cdot 2^{j+N-1} + w_{M-1} x_{N-1} \cdot 2^{(M-1)(N-1)} \end{aligned} \tag{4.4}$$

Figure 4.6 shows an 8-b×8-b multiplier schematic using the decomposition logic. In the first cycle, two partial sums will be generated by two 64-dimensional 4-b×4-b multipliers and then aggregated to form a 64-dimensional 8-b×4-b partial sum. This partial sum will be accumulated with the second-cycle partial sum and the pre-calculated compensation vector to get the final 64-dimensional 8-b×8-b MAC results.

Figure 4.7 shows the hardware implementation with bit-with flexibility. In a signed 8-b weight case, the left CIM macro stores 64 4-b signed weights, and the right CIM macro stores 64 4-b unsigned weights. Appling an 8-b signed input requires two operations: an unsigned X_N[3:0] for the first MAC operation and a signed X_N[7:4] for the second MAC operation. Although there are additional shift-add circuits outside two digital CIM macros to support different numerical precision of input and weight with a signed or an unsigned format, the energy-accuracy trade-off can be improved with a well-trained mixed-precision model. For CIM macros with 64 4-b weights and 4-b input activations, the bit precision of each PE output is 14-b without truncation and SNR degradation. In contrast, the limited resolution of ADCs degrades the linearity of analog CIM architec-

doi:10.6342/NTU202202019

tures. All in all, the proposed digital CIM macro has no post-processing computation error due to low SNR.

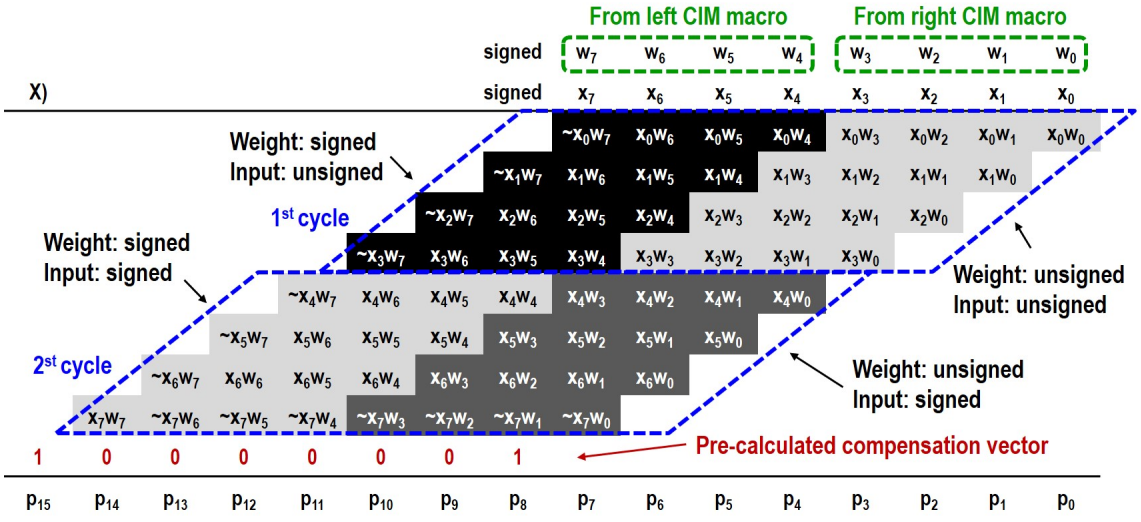


Figure 4.6: MAC operation of 8-b signed weight and 8-b signed input.

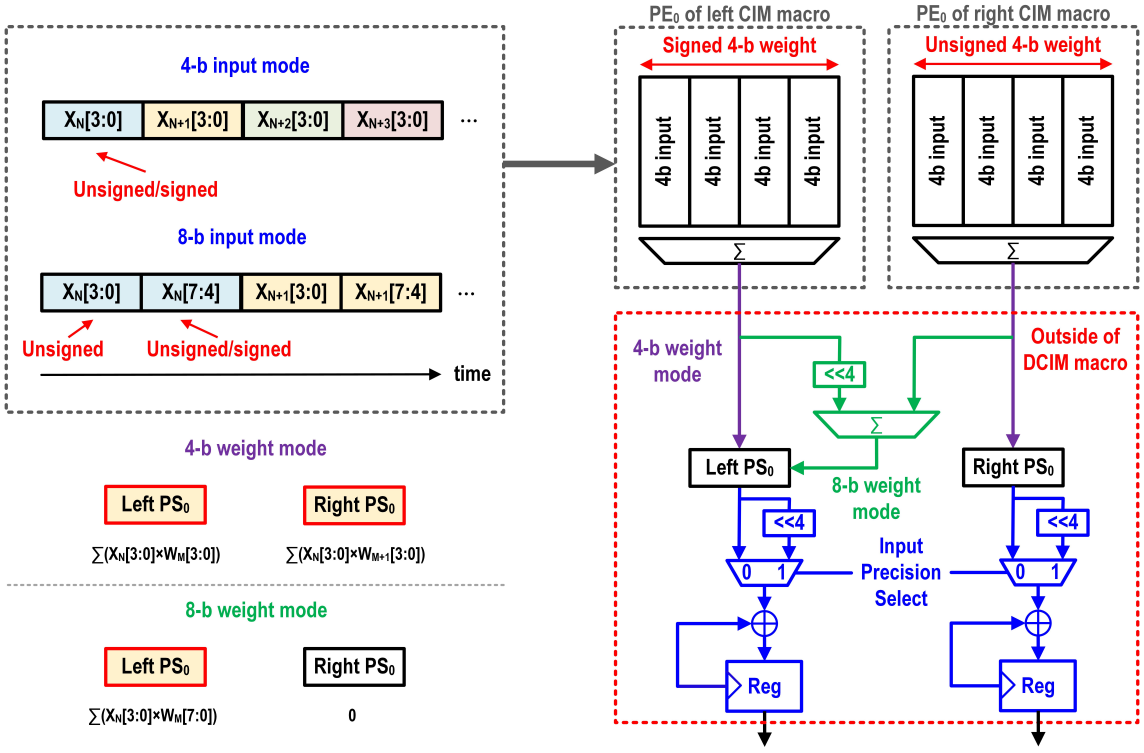


Figure 4.7: Schematic of the hardware implementation with bit-width flexibility.

doi:10.6342/NTU202202019

Chapter 5

Related Works of CIM-Based Neural Network Accelerators

Although the CIM macros mentioned in the previous chapter show good throughput and energy efficiency at the macro level, they might lose their upper hands at the system level since they do not consider the energy consumption of activation storage, on-chip communication, and other processes related to NNs. As a result, we want to introduce some related works of CIM-based accelerators that deal with system-level issues before we decide on our proposed architecture.

5.1 CIM-Based Accelerator Proposed by Liu *et al.*

A 65nm CIM-based CNN accelerator, proposed by Liu *et al.* [26], integrates four CIM cores with sparsity optimization. Their main contributions include:

1) An input-sparsity-aware acceleration to increase the system throughput and a weight-sparsity-aware power-saving to increase the system energy efficiency.

2) A kernel-order and channel-order mapping for higher macro utilization.

3) An inter-macro and intra-macro scheduling for efficient data reuse.

However, Figure 5.1 shows that the dynamic sparsity performance scaling architecture (DSS) requires additional index memory to record the corresponding zero index. Moreover, to support input sparsity, the input sparsity controller can only activate eight rows of the CIM macro for MAC operations, resulting in lower throughput. Moreover, this work supports limited NN layer types since non-MAC operations are not supported.

doi:10.6342/NTU202202019

Figure 5.2 presents the mechanism of the CIM-friendly input and weight sparsity controller. Although the weight-sparsity-aware technique can power off ADCs, it still has to store zero-valued weights.

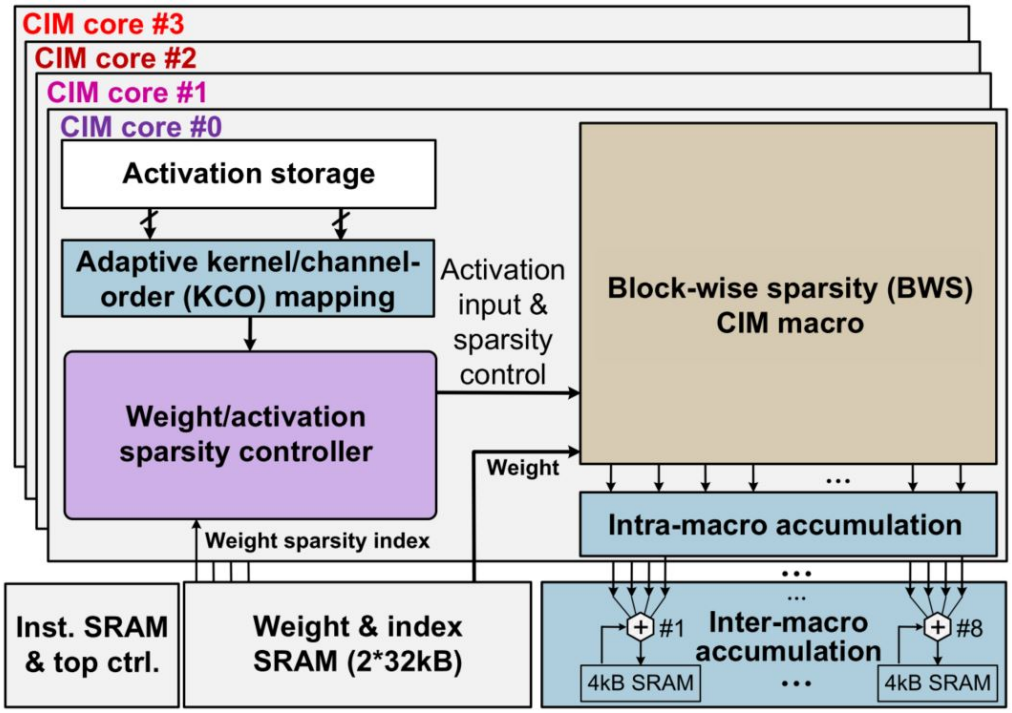
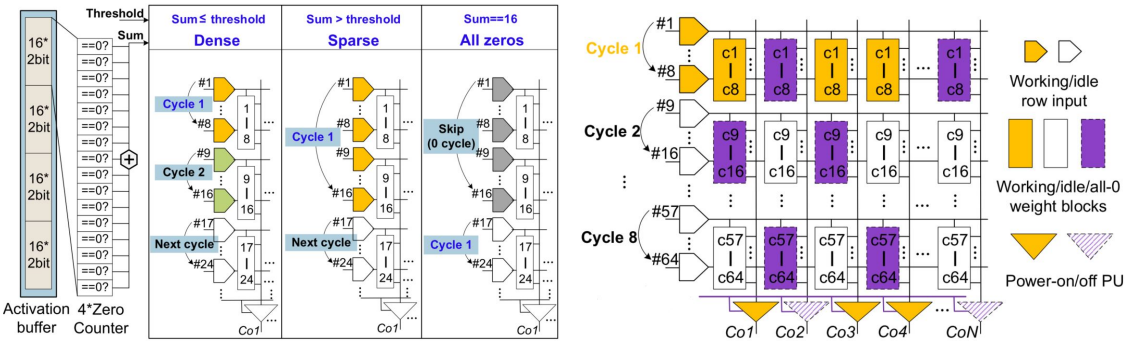


Figure 5.1: CIM-based accelerator proposed by Liu *et al.* [26].



(a) (b)

Figure 5.2: (a) Input sparsity controller for adaptive speedup. (b) Weight sparsity controller with adaptive power-off processing units (PU).

doi:10.6342/NTU202202019

5.2 CIM-Based Accelerator Proposed by Yue *et al.*

A 65nm CIM-based NN accelerator, proposed by Yue *et al.* [27], is shown in Figure 5.3. There are two features in this work. First, a set-associative block-wise zero-skipping architecture was proposed to skip the computation of zero-valued inputs and weights (see Figure 5.4). Therefore, it can reduce storage, power, and the number of execution cycles. Second, a ping-pong technique enables concurrent computing and weight updating operations. Although this work further reduces the execution time via the zero-skipping technique, an additional zero detection circuit and index memory are still required, increasing the system area overhead. At last, this work only supports MAC operation as its previous work [26].

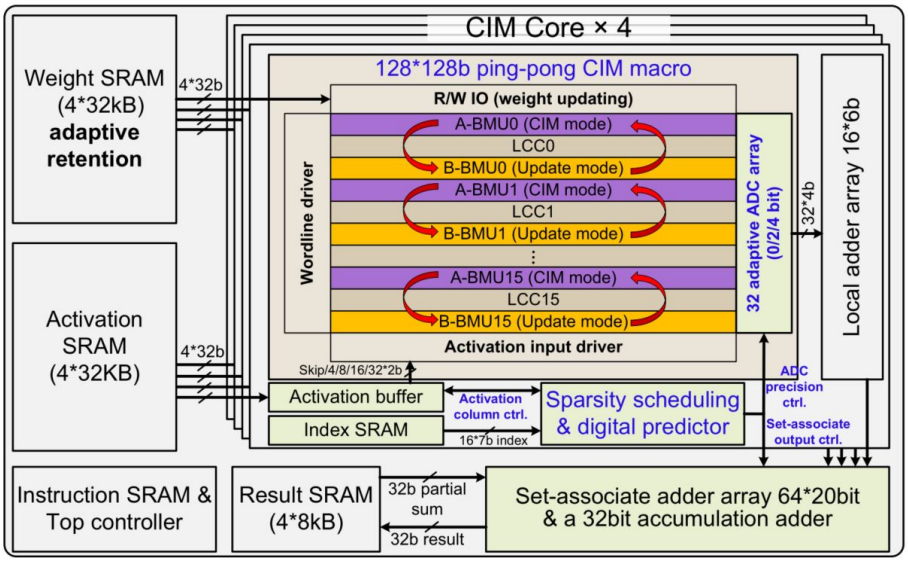


Figure 5.3: CIM-based accelerator proposed by Yue *et al.* [27].

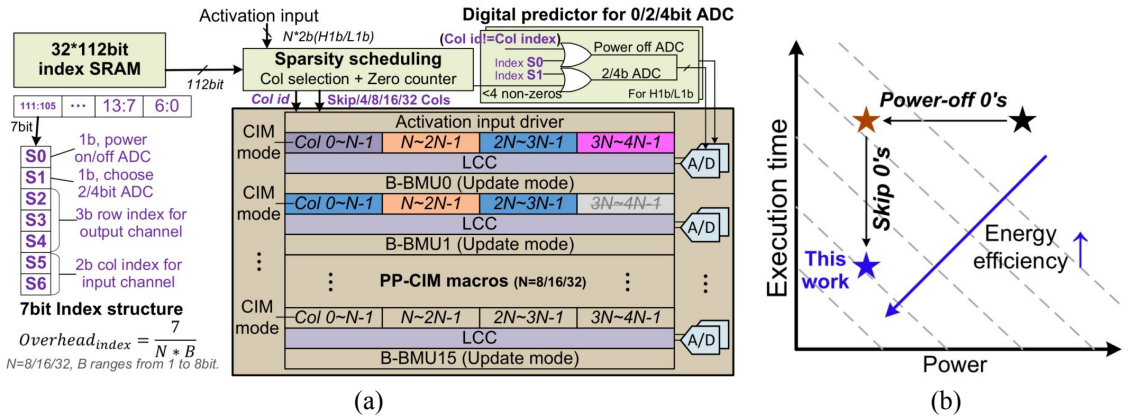


Figure 5.4: (a) Set-associative block-wise zero-skipping architecture. (b) Advantages of the zero-skipping.

doi:10.6342/NTU202202019

5.3 CIM-Based Accelerator Proposed by Seo *et al.*

A 28nm programmable in-memory computing accelerator (PIMCA), proposed by Seo *et al.* [28], demonstrates the most extensive integrations among CIM-based accelerators compared with [26], [27]. It can support various kernel sizes and NN layer types. Furthermore, a programmable single-instruction-multiple-data (SIMD) processor is integrated to support eight types of non-MAC operations.

Figure 5.5 depicts the overall six-stage CPU-like pipelined architecture, where 108 CIM macros are evenly deployed in six PEs. However, only one of the six PEs performs MVM using 1–18 macros at each cycle, reducing hardware utilization during computation. Besides, this work only supports 1-b and 2-b precision, so it is hard to be applied to large-scale datasets such as ImageNet.

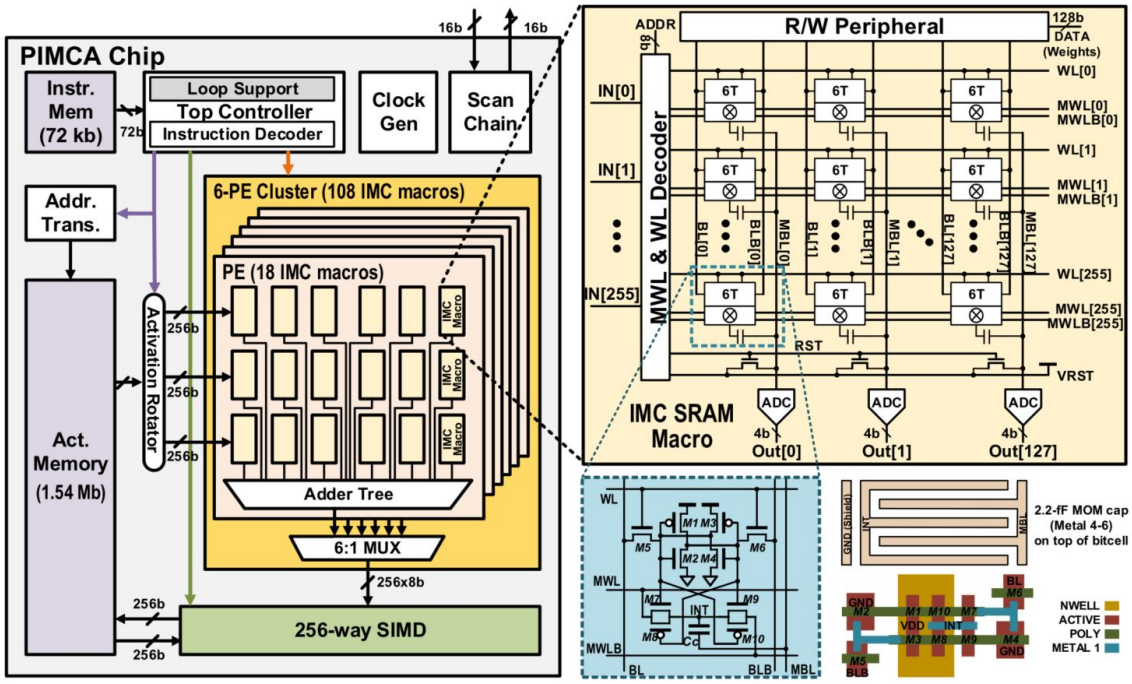


Figure 5.5: CIM-based accelerator proposed by Seo *et al.* [28].

doi:10.6342/NTU202202019

Chapter 6

Proposed CIM-Based Accelerator

6.1 Overall System Architecture

Figure 6.1 shows the hierarchical CIM-based accelerator design. The overall architecture integrates 36 digital CIM macros evenly deployed in two cores. In each core, 18 CIM macros are partitioned into two 3×3 CIM macro arrays, and each one supports eight 64-dimensional 4-b×4-b MAC operations.

The operation principle is described as follows. Initially, the inputs, weights, and instructions are fetched from an off-chip DRAM and preloaded in the top global activation memory (GAM), the global weight memory (GWM), and the global instruction memory (GIM), respectively. Then, weights are sent from GWM to 36 CIM macros by the top

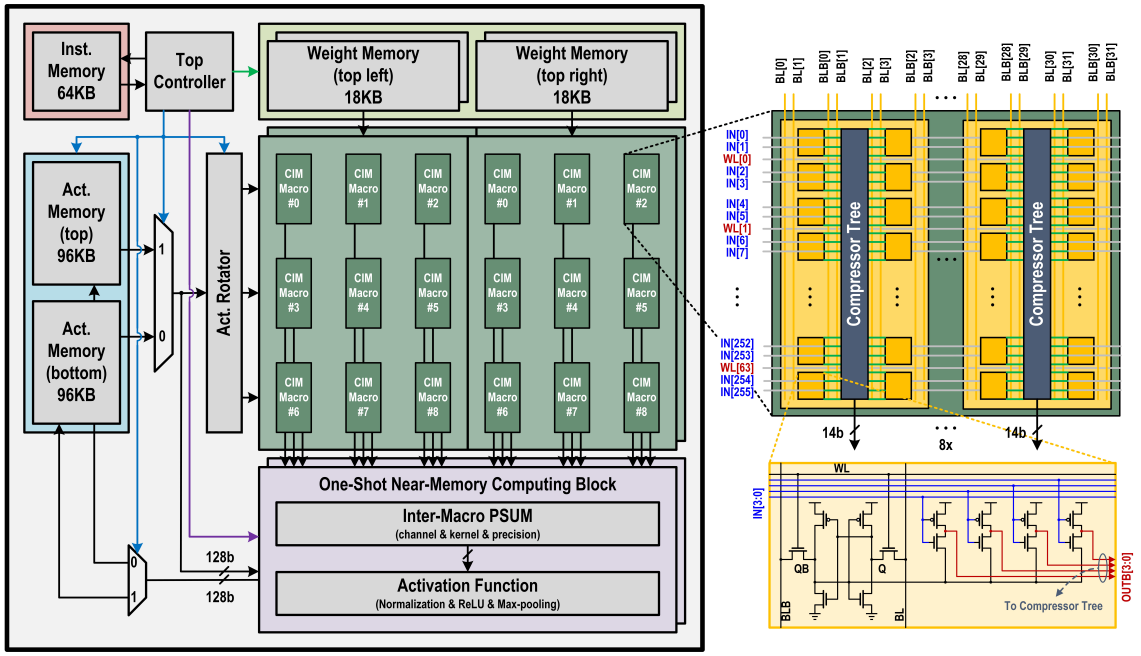


Figure 6.1: Overall architecture of the proposed CIM-based accelerator.

doi:10.6342/NTU202202019

controller. In CIM mode, the two CIM cores share the same inputs, performing MAC operations for up to 32 (= 8×4) output channels in parallel. Each core performs MVMs using two 3×3 macro arrays to support convolutions with the 3×3 kernel size. The proposed reconfigurable compressor tree then adds the partial products and sends the partial result to the proposed one-shot NMC block for post-processing (linear operation, ReLU, and max pooling). Finally, the top controller stores the output feature map results into the other GAM, which will perform computation for the next layer similarly.

Section 4.4 states that the proposed digital CIM macro can support 4b and 8b precision. Certainly, additional circuitry is also necessary to support bit-width flexibility. Therefore, the 8-b input accumulation and the 8-b weight aggregation are implemented in the proposed one-shot NMC blocks. The right part of Figure 6.2 shows the corresponding timing diagram of 64-dimensional 8-b input × 4-b weight MAC operation in one PE. The adder tree sums up the outputs from the compressors, and the shift accumulator in a one-shot NMC block accumulates the partial sum of each cycle in a pipelined manner (see Figure 6.10).

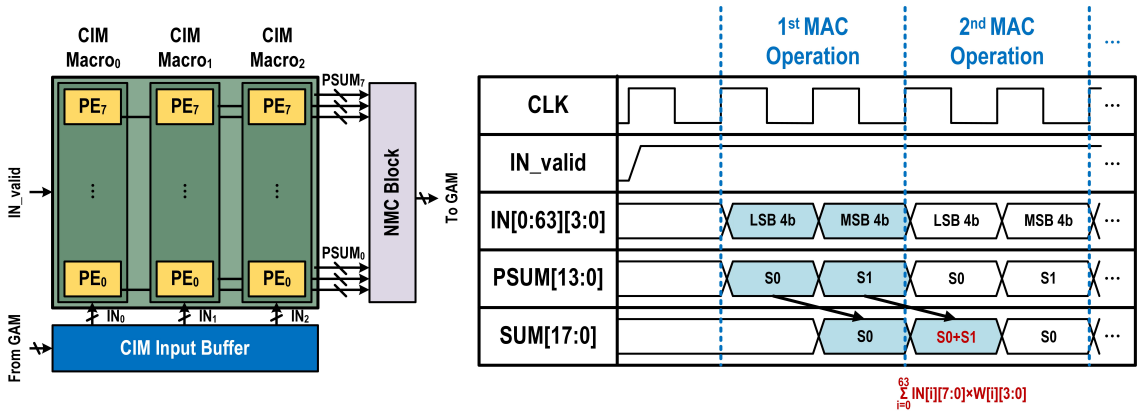


Figure 6.2: Timing diagram of MAC operation in 8-b input × 4-b weight scenario.

doi:10.6342/NTU202202019

6.2 System Pipelined Architecture

Figure 6.3 depicts the CPU-like five-stage pipelined architecture consisting of GIM, GAM, 36 CIM macros, and the NMC blocks. Initially, the top controller preloads weight data in 36 CIM macros. An instruction is fetched in the instruction fetch (IF) stage one cycle by one cycle, and input vectors are loaded from GAM in the load data (LD) stage. Next, MAC operations are executed with the weight-stationary data flow in the CIM stage, and other non-MAC computations are executed in the NMC stage. Finally, the output feature map results are written back to GAM in the write-back (WB) stage. A regular instruction performing MAC and non-MAC operations contains three major fields:

1) Addresses and enabling signals for standard read and write.

2) Resolution (4-b or 8-b) and representation (signed or unsigned) mode selection.

3) Near-memory computation operands and operation code for post-processing.

Unlike PIMCA [28], the proposed pipelined architecture can activate 36 CIM macros simultaneously with 100% hardware utilization in theory during the CIM stage, resulting in higher energy and area efficiency. Moreover, we use one-shot NMC blocks instead of SIMD processors to avoid massive data access in GAM to increase the system throughput.

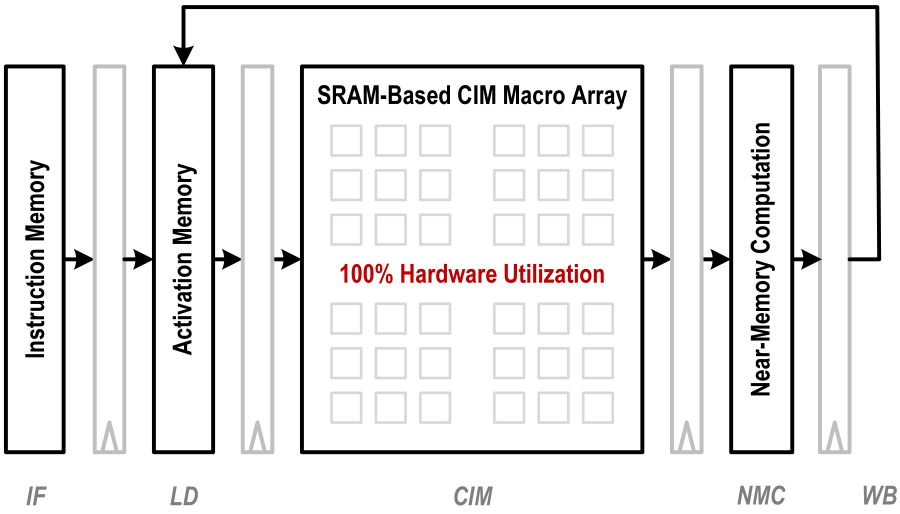


Figure 6.3: Architecture of CPU-like pipeline integrating 36 CIM macros.

doi:10.6342/NTU202202019

6.3 Macro Array Dataflow

Figure 6.4 illustrates the horizontal pipelined architecture similar to [28] and the corresponding data mapping method. To support the kernel size of 3×3, we partition the 3×6 macros into two 3×3 CIM macro arrays in a CIM core, so we can map 64 (16) input (output) channels in a 4-b convolution layer case or 64 (8) input (output) channels in an 8-b convolution layer case in a CIM core. In each 3×3 CIM macro array, we introduce the horizontal pipelined architecture to reuse the convolutional data. Therefore, CIM macros adopt the weight-stationary (WS) dataflow while macro arrays adopt the row-stationary (RS) dataflow. With the help of the horizontal pipelined architecture, the system throughput can be reduced from

$$N_{\text{cycle}} = \frac{H_{\text{o}} \times (3W_{\text{o}}) \times C_{\text{i}} \times C_{\text{o}} \times B_{\text{X}} \times B_{\text{W}}}{64 \times 32 \times 4 \times 4} \tag{6.1}$$

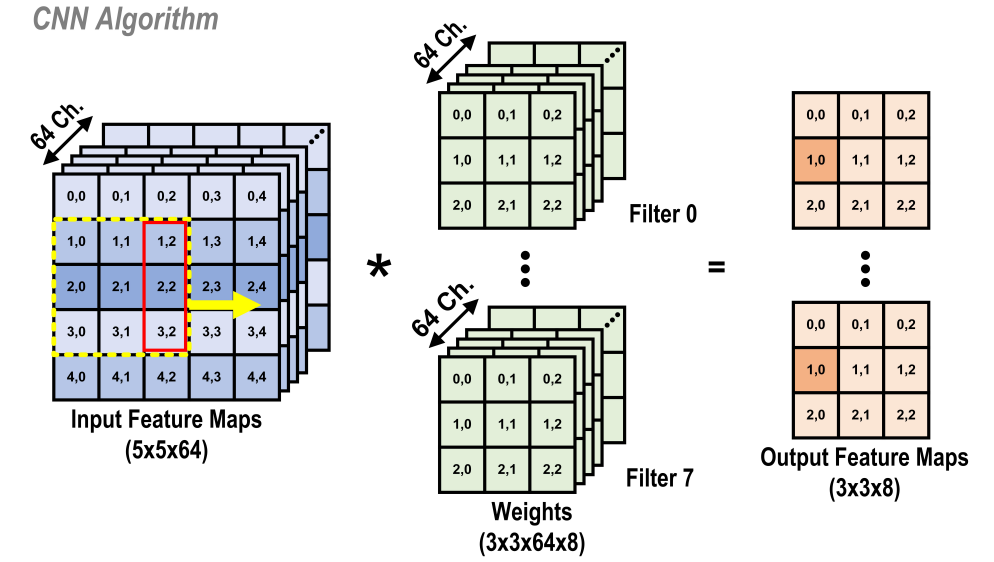
to

$$N_{\text{cycle}} = \frac{H_{\text{o}} \times (W_{\text{o}} + 2) \times C_{\text{i}} \times C_{\text{o}} \times B_{\text{X}} \times B_{\text{W}}}{64 \times 32 \times 4 \times 4}\ , \tag{6.2}$$

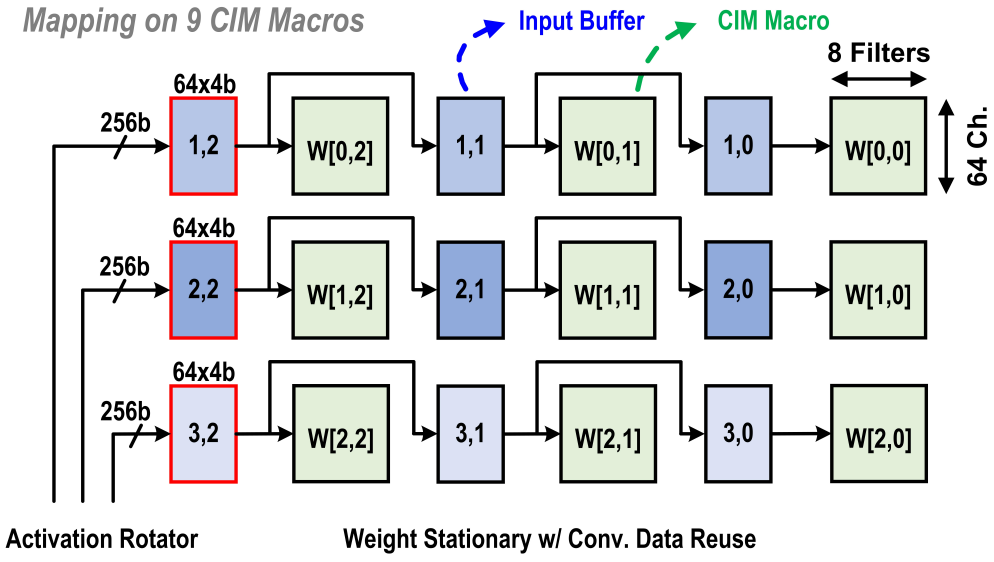
where N_{cycle} represents the number of execution cycles, H_{o} represents the height of the output feature map, W_{o} represents the width of the output feature map, C_{i} represents the number of the input channels, C_{o} represents the number of the output channels, B_{X} represents the bit-width of input, and B_{W} represents the bit-width of weight.

Figure 6.5 shows the estimated execution time of convolution layers in various NN topologies. The system pipeline with the horizontal pipeline can significantly reduce the execution time by 84.6%, 72.0%, and 89.8% in VGG16, ResNet18, and ResNet20 topology on a CIFAR-10 dataset, respectively.

doi:10.6342/NTU202202019



(a)



(b)

Figure 6.4: Schematic of an efficient streaming process in CIM macro arrays. (a) Algorithm. (b) Horizontal pipelined architecture [28].

doi:10.6342/NTU202202019

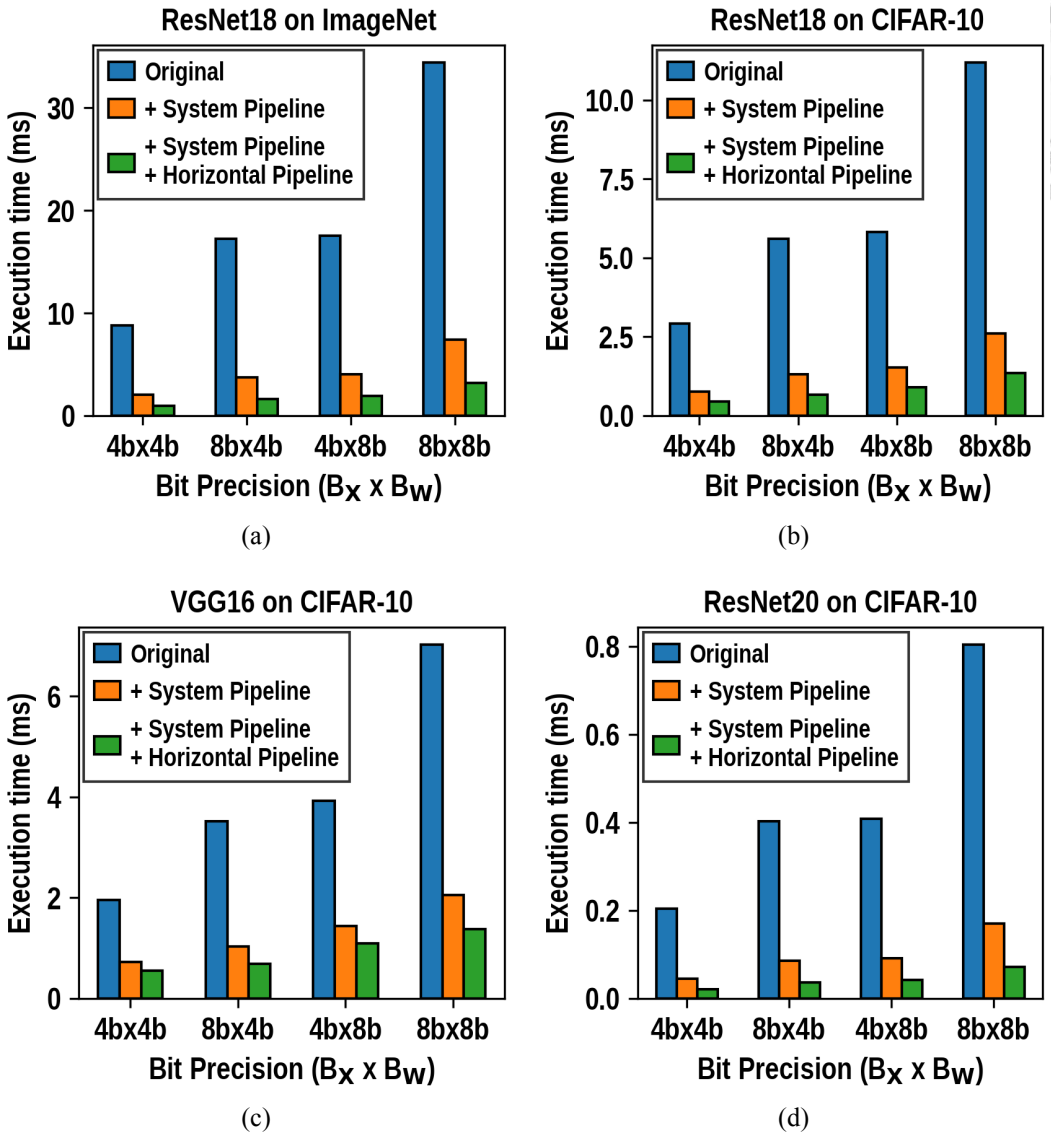


Figure 6.5: Estimated execution time of convolution layers in various NN topologies. (a) ResNet18 on ImageNet. (b) ResNet18 on CIFAR-10. (c) VGG16 on CIFAR-10. (d) ResNet20 on CIFAR-10.

6.4 Global Weight Memory

As shown in Figure 6.6, this work integrates a 576-Kb global weight memory (GWM) using digital single-port SRAMs for weight storage. Specifically, GWM comprises two cores with 72 banks (512 rows $\times$ 32 bits) for each. In standard write (STDW) mode, the CIM macro array can access any row from those banks and write 4.5-Kb (= 2 cores $\times$ 72 banks $\times$ 32 bits) weight data into any row of 36 CIM macros in one clock cycle. Therefore,

doi:10.6342/NTU202202019

GWM can mitigate off-chip weight accesses for a range of NN topologies. In this way, weight updating of 36 CIM macros only needs 64 clock cycles. In addition, GWM supports eight-rounded weight updating before the off-chip weight access in the next round.

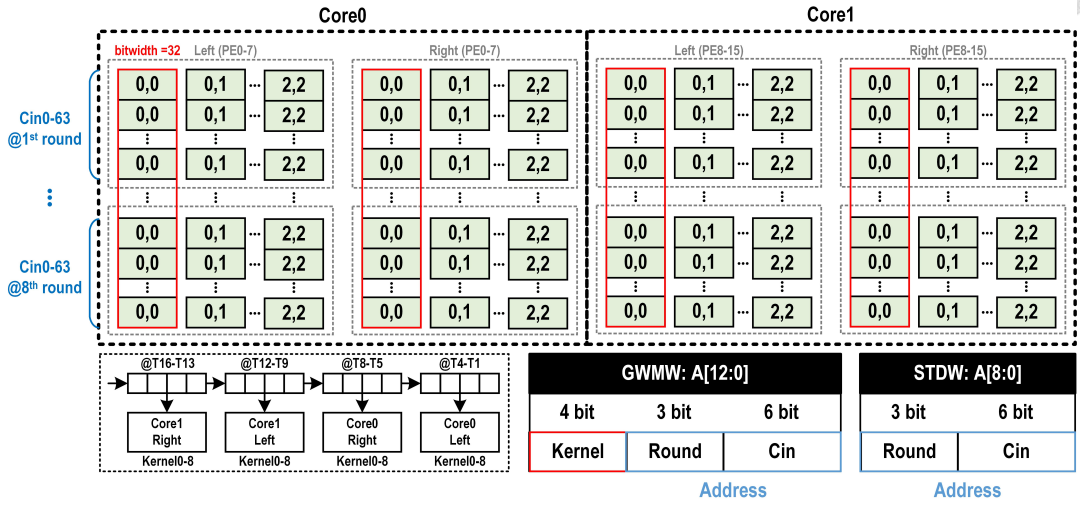
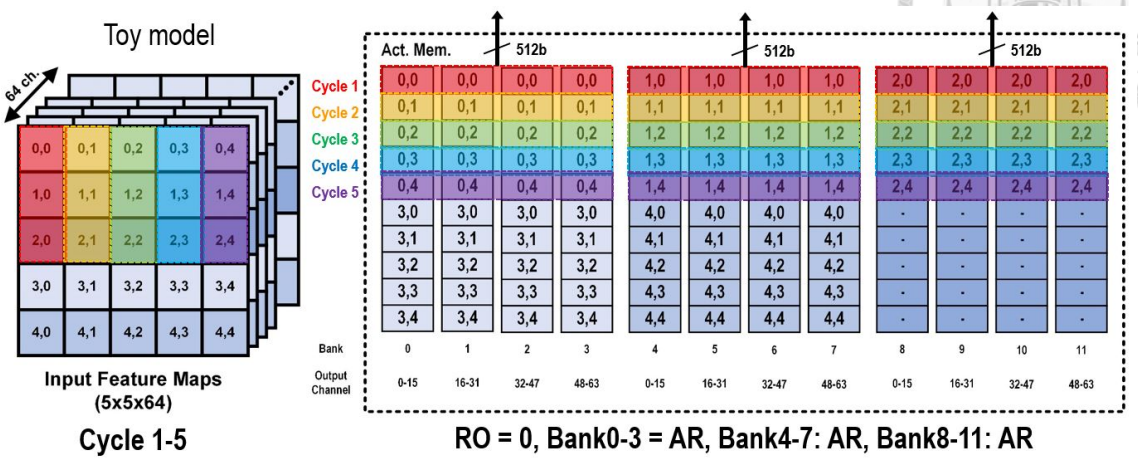


Figure 6.6: Schematic of the global weight memory architecture.

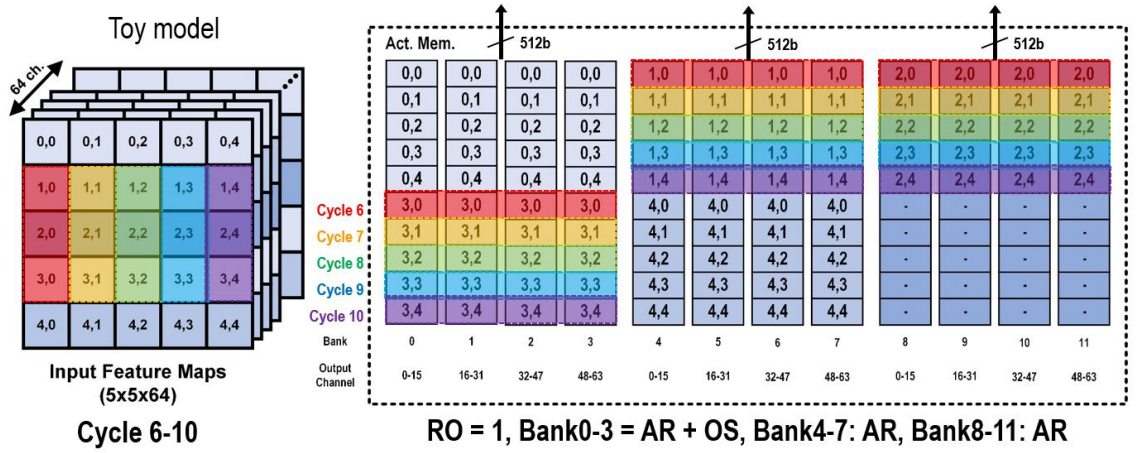
6.5 Activation Storage and Efficient Processing

As shown in Figure 6.7, this work also integrates 1.5-Mb GAM (digital SRAM) to store input feature maps and final outputs. The input and output data of a layer must simultaneously access the GAM. Therefore, similar to PIMCA [28], we separate the GAM into two groups for time-interleaved storage across layers, each composed of twelve banks (512 rows×128 bits) that can support high-bandwidth GAM access. Thus, two CIM cores are available to access up to 1536-b (height×width×channel×input precision) inputs from GAM banks in one clock cycle. To support this access, we implement an activation rotator to send the input activation to the corresponding CIM macros, as shown in Figure 6.8. Unlike [29], which needs a word-to-bit reshaping buffer for input activation reuse in convolutions, we implement the activation rotator with a combinational circuit, simplifying the data processing without additional buffering between GAM and CIM core. As a result, the system area efficiency is further improved.

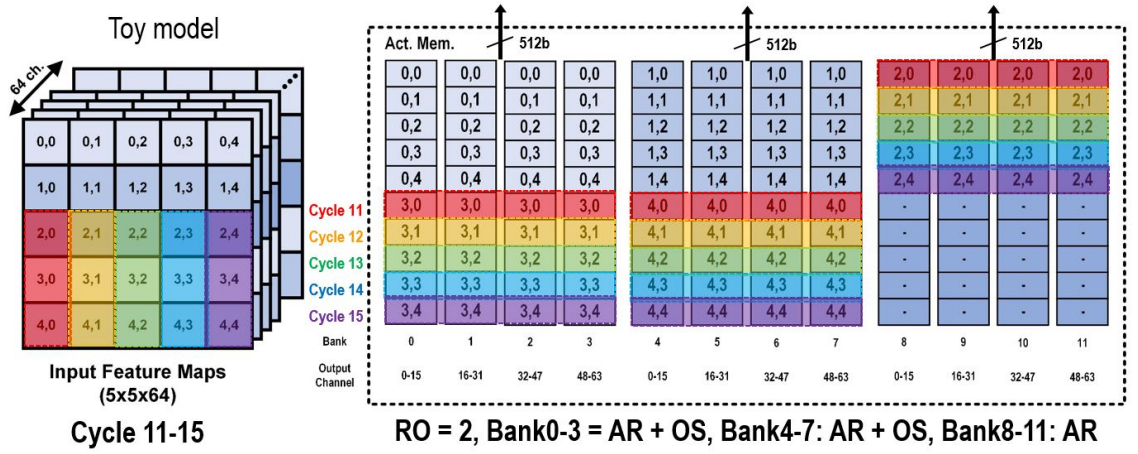
doi:10.6342/NTU202202019



(a)



(b)



(c)

Figure 6.7: Global activation memory architecture and the corresponding hardware mapping method. (a) Read order (RO) = 0. (b) Read order = 1. (c) Read order = 2.

doi:10.6342/NTU202202019

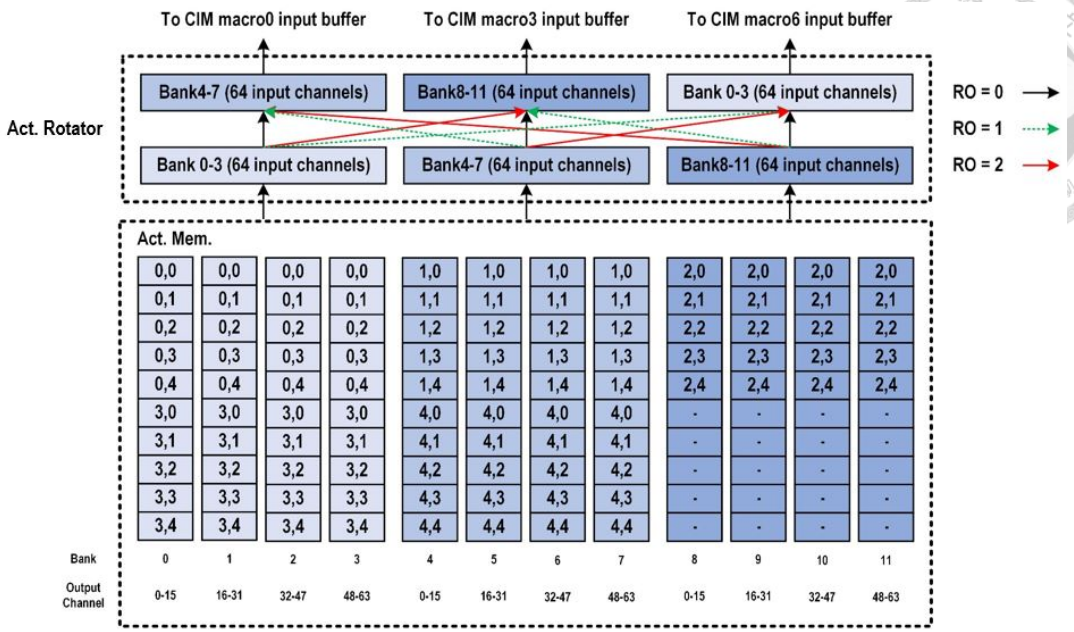


Figure 6.8: Schematic of the activation rotator (RO represents the read order).

6.6 One-Shot NMC Block

Figure 6.9 shows the data path in the proposed one-shot NMC block. The NMC data path consists of a series of hardware computation stages, including partial sum accumulation (PSA), global offset (GO), global left-shift (GLS), ReLU, and max pooling, where the former three stages are for linear operations.

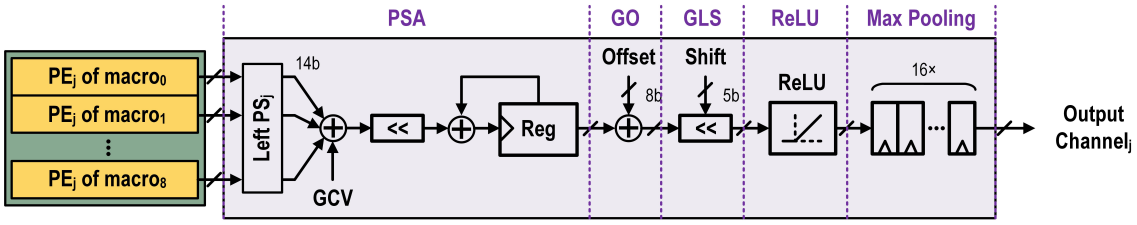


Figure 6.9: Data path of the proposed one-shot NMC block.

Moreover, the one-shot NMC blocks need to support 4-b and 8-b computation. In 4-b weight mode, one-shot NMC blocks add the partial sum of each PE in macro0–8 to output 32 results representing 3×3 kernel partial sum, as shown in Figure 6.10(a); in 8-b weight mode, one-shot NMC blocks add the partial sum of each PE in macro0–8 to output

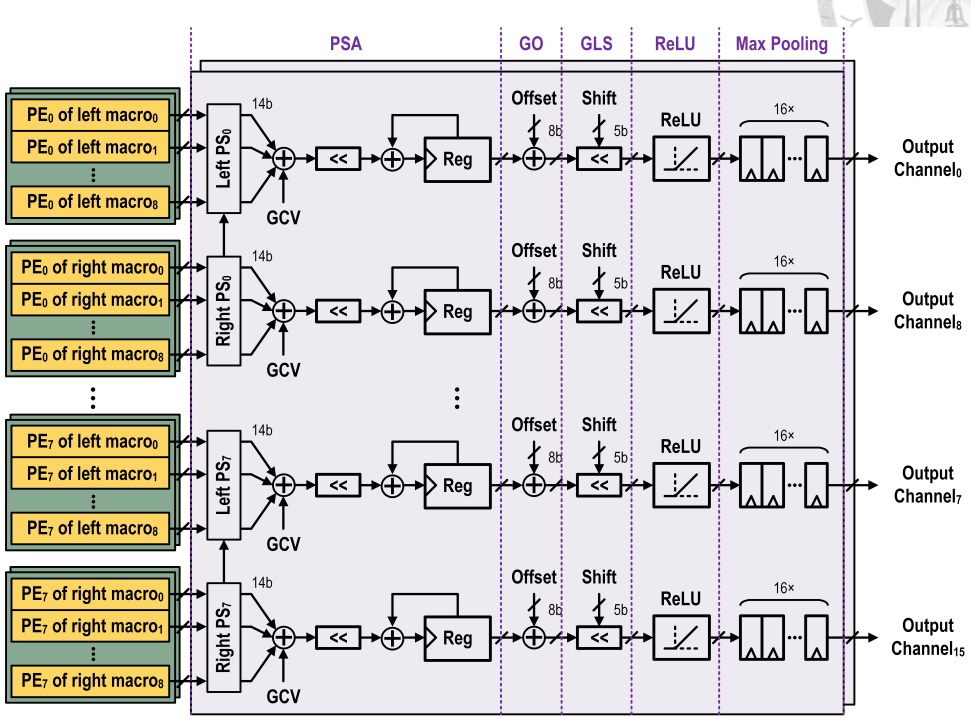
doi:10.6342/NTU202202019

16 results, as shown in Figure 6.10(b).

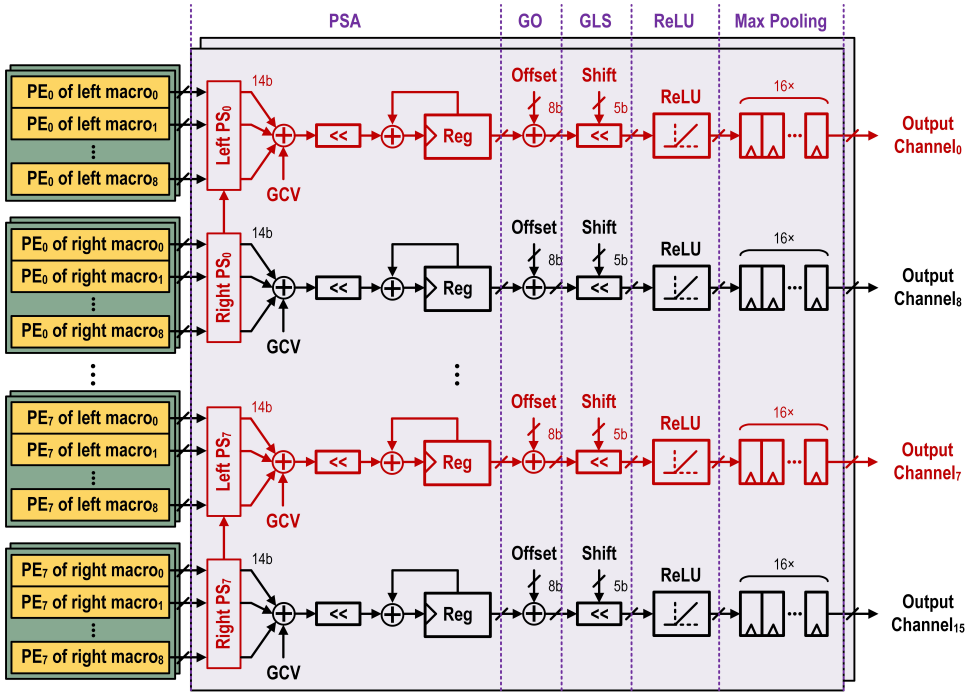
In addition, to support a 2×2 max-pooling operation in the CIFAR-10 dataset, every 16 registers in a max-pooling stage are in charge of output feature map storage. For example, if there are 32 pixels in each row of an input feature map, the 16 registers can temporally store the 16 larger pixels in the first row. Then, the 32 pixels in the second row are compared with 16 pixels in the registers, and the 16 maximum pixels will be preserved in this way. Finally, the top controller writes the output feature map results to the other GAM as the input feature map of the following layer.

Figure 6.11 shows the estimated execution time of convolution layers in various NN topologies. Compared with the SIMD processor [30], the one-shot NMC blocks can further reduce the execution time by 78.8%, 85.5%, and 88.3% in VGG16, ResNet18, and ResNet20 topology on a CIFAR-10 dataset, respectively.

doi:10.6342/NTU202202019



(a)



(b)

Figure 6.10: Schematic of the proposed one-shot NMC block. (a) 4-b weight mode and (b) 8-b weight mode.

doi:10.6342/NTU202202019

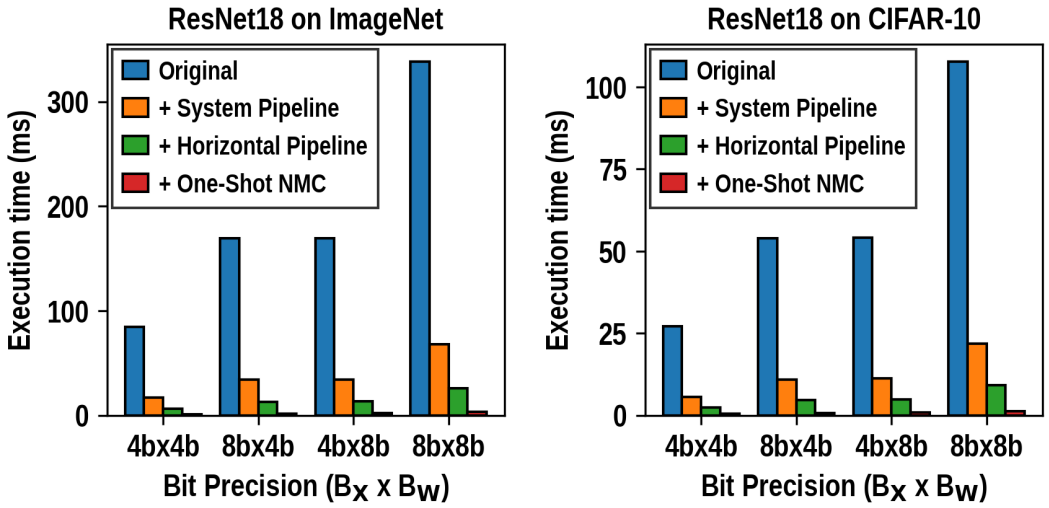


Figure 6.11: Estimated execution time of convolution layers in various NN topologies. (a) ResNet18 on ImageNet. (b) ResNet18 on CIFAR-10. (c) VGG16 on CIFAR-10. (d) ResNet20 on CIFAR-10.

doi:10.6342/NTU202202019

Chapter 7

Chip Implementation and Simulation Results

The proposed CIM-based accelerator was designed and implemented with standard 28nm CMOS technology. Figure 7.1 shows the place-and-route result of the CIM test chip. The overall design is implemented through the hybrid full-custom and cell-based design flow. Table 7.1 summarizes the simulation results under different numerical precisions. The SRAM-based bit cell achieves a high area efficiency of 81.92 TOPS/mm^2 at 0.9 V and 166 MHz. Moreover, with the aid of the reconfigurable compressor tree and one-shot NMC blocks, the proposed CIM-based accelerator achieves a peak energy efficiency of 625.15 TOPS/W at 0.9 V and 166 MHz.

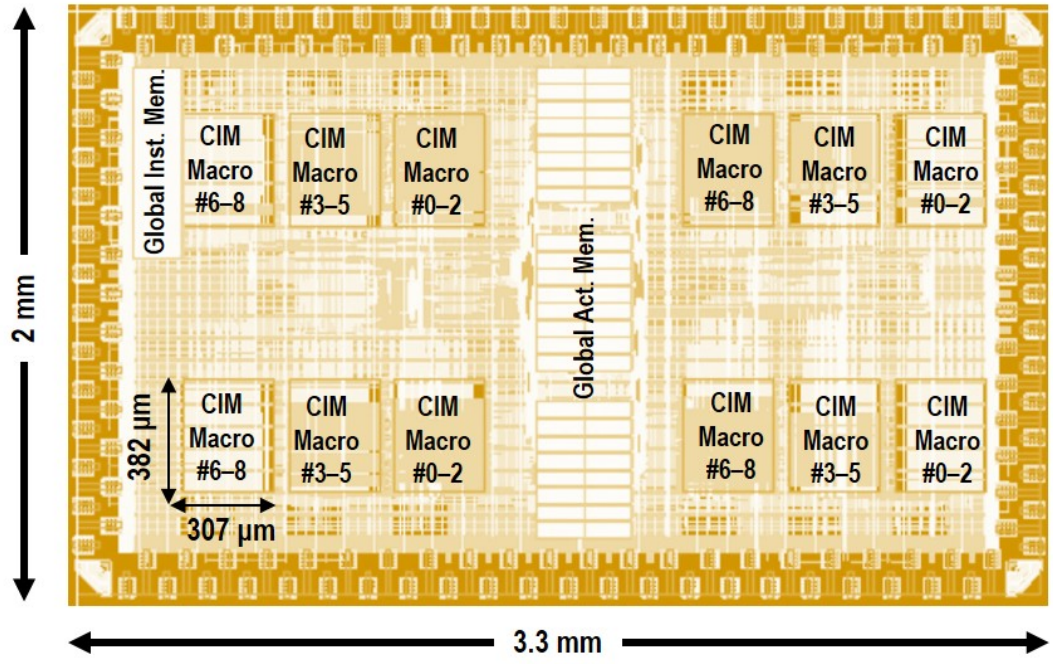


Figure 7.1: Place-and-route result of the proposed CIM-based accelerator.

doi:10.6342/NTU202202019

Table 7.1: Chip specification based on simulation results.

Technology	28 nm			
CIM Size	72 Kb			
Bit Cell Area	1.512 μm^2			
CIM Macro Area	33890.904 μm^2			
Digital SRAM Size	2.28 Mb			
Core Area	4.62 mm^2			
Supply Voltage	0.9 V			
Clock Rate	166 MHz			
Input / Weight Precision	4b / 4b	4b / 8b	8b / 4b	8b / 8b
Throughput (TOPS)	6.14	3.07	3.07	1.54
Core Area Efficiency (GOPS/mm^2)	1330	665	665	332.5
Core Energy Efficiency (TOPS/W)	26.906	13.453	13.453	6.727

7.1 Layout of the Proposed SRAM-Based Bit Cell

As shown in Figure 7.2, the bit cell area of the proposed 14T SRAM-based bit cell demonstrates a compact footprint of 1.512 μm^2. The area of the standard 6T SRAM bit cell is 0.756 μm^2, and the transistor count overhead of an input bit is only 2T (customized NAND gate), which occupies a 33.3% area overhead according to the transistor placement. As a result, the proposed digital CIM macro achieves a high area efficiency of 81.92 TOPS/mm^2 at 0.9 V and 166 MHz.

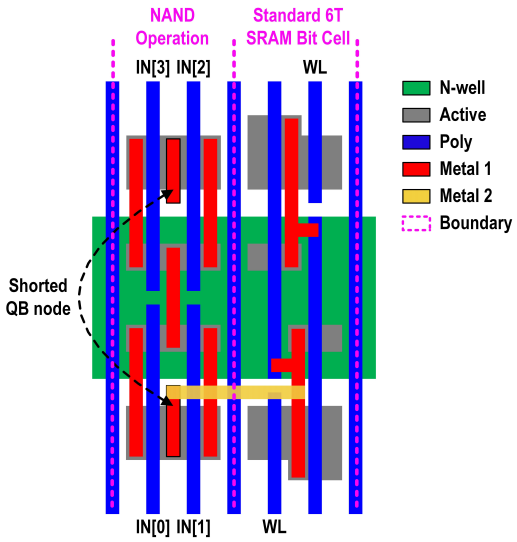
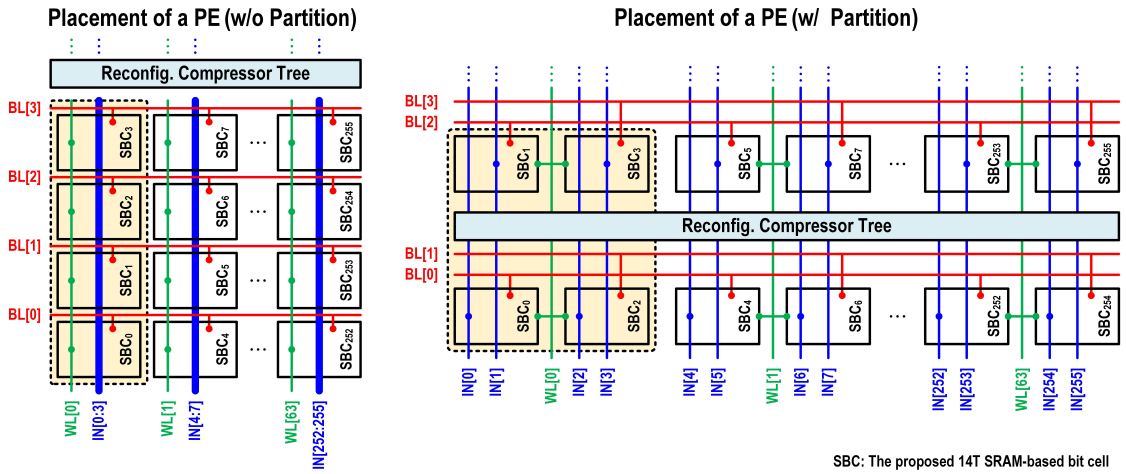


Figure 7.2: Layout implementation of the proposed 14T SRAM-based bit cell.

doi:10.6342/NTU202202019

7.2 Partitioned Bit Cell Placement

Figure 7.3 presents the placement optimizations of a PE. The layout of the bit cell array fits the size of the reconfigurable compressor tree, making the overall layout design more compact (10.5% area overhead reduction in one PE). Also, the sandwiched compressor tree placement mitigates the horizontal routing congestion, and the interleaved BL mitigates vertical routing congestion. As a result, we achieve a compact layout design with the proposed technique.



(a) (b)

Figure 7.3: Schematic of the SRAM bit cell placement in a PE. (a) W/o partition. (b) W/ partition.

7.3 Floorplan of the Overall Architecture

Figure 7.4 shows the floorplan of the overall architecture, which mainly comprises 36 digital CIM macros, a 2.28-Mb digital SRAM (GAM, GWM, and GIM), two activation rotators, and two one-shot NMC blocks. Figure 7.5 shows the floorplan of three CIM macros. Each macro includes eight PEs and the peripheral circuit, and three macros share one CIM input buffer for the horizontal pipeline.

doi:10.6342/NTU202202019

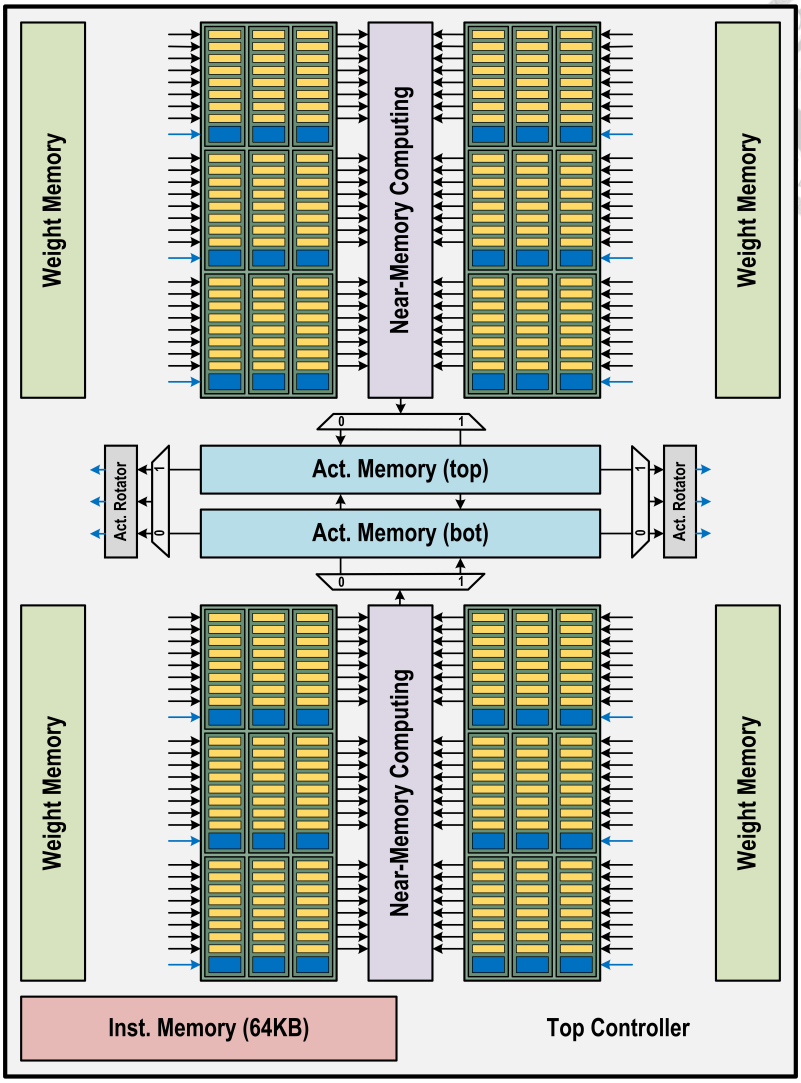


Figure 7.4: Floorplan schematic of the overall architecture.

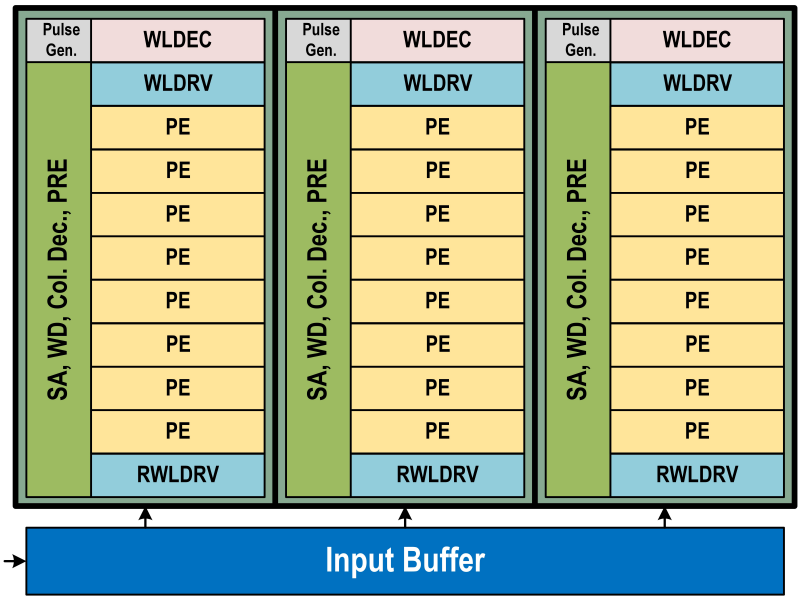


Figure 7.5: Floorplan schematic of the three CIM macros.

doi:10.6342/NTU202202019

7.4 Area Breakdown

The area breakdown (Figure 7.6) shows that the 72-Kb SRAM CIM macro array dominates the area cost of hard macros, and the area of the 2.28-Mb digital SRAM (including 24 global activation SRAMs, 36 global weight SRAMs, and a global instruction SRAM) only accounts for 15.7% of the core area.

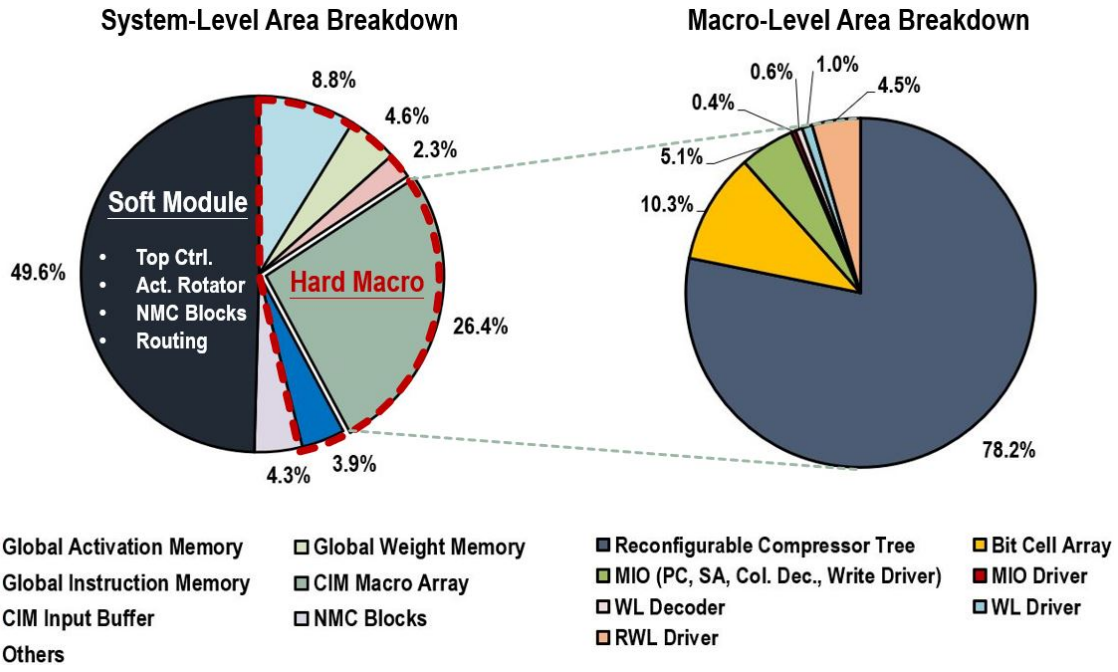


Figure 7.6: Area breakdown.

7.5 Comparison to Other CIM Designs

Table 7.2 compares the proposed CIM-based accelerator with the state-of-the-art system-level CIM designs that support bit-width flexibility. To observe the effect of the technology scaling, we normalize the technology to 28nm. The way we normalize the technology from Xnm to 28nm is to multiply the throughput by X/28, the area efficiency by $(X/28)^3$, and the energy efficiency by $(X/28)^2$. Thanks to the CPU-like pipelined architecture and the proposed one-shot NMC blocks, the proposed design achieves a 97.1%–99.1% execution time reduction of convolution layers in various NN topologies while

doi:10.6342/NTU202202019

maintaining competitive area and energy efficiency performance. Moreover, with the proposed 14T SRAM-based bit cells, this work achieves a 6.7× higher throughput per CIM size than the previous CIM-based NN accelerator [26]. Lastly, the proposed CIM macro utilizes a partitioned bit cell placement structure to maintain symmetric layout implementation and mitigate routing congestion of the compressor tree, substantially benefiting the area efficiency of the large-scale CIM-based computation systems.

Table 7.2: Performance comparison with the state-of-the-art CIM designs.

	ISSCC'20 [26]	ISSCC'21 [27]	VLSI'21 [28]	ISSCC'21 [31]	ISSCC'21 [1]	ISSCC'22 [32]	**This Work[c]**
Technology (nm)	65	65	28	16	28	28	**28**
MAC Operation	Analog	Analog	Analog	Analog	Analog	Digital	**Digital**
Operating Voltage (V)	0.9	1.0	0.85–1.2	0.8	0.9	0.63–0.95	**0.9**
Frequency (MHz)	100	100	40	200	216	30–105	**166**
CIM Size (Kb)	16	64	3546	4608	1024	1728	**72**
Input Bits (b)	2/4/6/8	2/4/6/8	1/2	1–8	4/8	4	**4/8**
Weight Bits (b)	4/8	1–8	1/2	1–8	2/4/8	3	**4/8**
Throughput[a] (TOPS)	0.95[b]	14.67[b]	4.90	107.89[b]	16.39[b]	0.97	**98.30**
Throughput /CIM Size[a] (GOPS/Kb)	59.43[b]	229.24[b]	1.38	23.41[b]	16[b]	0.56	**1365.33**
Macro Area Eff.[a] (TOPS/mm^2)	8.66	46.16	2.67	7.97	>1.83[b]	10.49	**81.92**
System Area Eff.[a] (TOPS/mm^2)	0.90	9.50	0.23	N/A	N/A	0.11	**21.28**
Macro Energy Eff.[a] (TOPS/W)	580.88	3339.1	588	632.16	383[b]	45.7	**625.15**
System Energy Eff.[a] (TOPS/W)	33.86	404.98	437	N/A	N/A	32.9	**430.50**

[a]Normalized to 28nm and 1b×1b MACs (1 MAC is 2 OPs).
[b]Estimated value.
[c]Simulation results.

doi:10.6342/NTU202202019

Chapter 8

Conclusion and Future Work

8.1 Conclusion

This thesis presents a 28nm reconfigurable CIM-based accelerator that integrates 36 2-Kb digital CIM SRAM macros. The proposed digital CIM macro with compact 14T SRAM-based bit cells can support parallel 4-b inputs compared with PVT-variation-sensitive ACIM designs. In addition, the proposed reconfigurable compressor tree supporting mixed-precision computation and different sign notations can improve the energy-accuracy tradeoff. Lastly, the proposed one-shot NMC blocks avoid massive data movements with GAM and reduce 78.8%–88.3% execution time in convolution layers of various NN topologies. The simulation results show that the proposed CIM-based accelerator achieves a peak throughput of 98.30 TOPS, a peak area efficiency of 21.28 TOPS/mm^2 and a peak energy efficiency of 430.50 TOPS/W. In conclusion, this work achieves high throughput and area efficiency while providing enhanced flexibility and programmability.

8.2 Future Work

Based on our preliminary analyzed results, this work still has much room for improvement in hardware. On the one hand, the appropriate application of digital and analog architectures can be further explored. On the other hand, the sparsity-aware technique can be taken into account in the future so that the hardware can achieve higher energy efficiency and system throughput in sparse models.

doi:10.6342/NTU202202019

References

[1] R. Guo *et al.*, "15.4 A 5.99-to-691.1TOPS/W tensor-train in-memory-computing processor using bit-level-sparsity-based optimization and variable-precision quantization," *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, 2021, pp. 242–244.

[2] J. Zhang, Z. Wang, and N. Verma, "In-memory computation of a machine-learning classifier in a standard 6T SRAM array," in *IEEE Journal of Solid-State Circuits*, vol. 52, no. 4, pp. 915–924, Apr. 2017.

[3] S. K. Gonugondla, M. Kang, and N. R. Shanbhag, "A variation-tolerant in-memory machine learning classifier via on-chip training," in *IEEE Journal of Solid-State Circuits*, vol. 53, no. 11, pp. 3163–3173, Nov. 2018.

[4] A. Biswas and A. P. Chandrakasan, "CONV-SRAM: An energy-efficient SRAM with in-memory dot-product computation for low-power convolutional neural networks," in *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 217–230, 2019.

[5] J. Yang *et al.*, "24.4 Sandwich-RAM: An energy-efficient in-memory bwn architecture with pulse-width modulation," *2019 IEEE International Solid- State Circuits Conference (ISSCC)*, 2019, pp. 394–396.

[6] H. Kim, Q. Chen, and B. Kim, "A 16K SRAM-based mixed-signal in-memory computing macro featuring voltage-mode accumulator and row-by-row ADC," *2019 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, 2019, pp. 35–36.

[7] Z. Jiang, S. Yin, J.-S. Seo, and M. Seok, "C3SRAM: An in-memory-computing SRAM macro based on robust capacitive coupling computing mechanism," in *IEEE Journal of Solid-State Circuits*, vol. 55, no. 7, pp. 1888–1897, Jul. 2020.

[8] Q. Dong *et al.*, "15.3 A 351TOPS/W and 372.4GOPS compute-in-memory SRAM macro in 7nm FinFET CMOS for machine-learning applications," *2020 IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 242–244.

doi:10.6342/NTU202202019

[9] C. Yu, T. Yoo, T. T.-H. Kim, K. C. Tshun Chuan, and B. Kim, "A 16K current-based 8T SRAM compute-in-memory macro with decoupled read/write and 1-5bit column ADC," *2020 IEEE Custom Integrated Circuits Conference (CICC)*, 2020, pp. 1–4.

[10] Y.-T. Hsu, C.-Y. Yao, T.-Y. Wu, T.-D. Chiueh, and T.-T. Liu, "A high-throughput energy–area-efficient computing-in-memory SRAM using unified charge-processing network," in *IEEE Solid-State Circuits Letters*, vol. 4, pp. 146–149, 2021.

[11] S. Xie, C. Ni, A. Sayal, P. Jain, F. Hamzaoglu, and J. P. Kulkarni, "16.2 eDRAM-CIM: Compute-in-memory design with reconfigurable embedded-dynamic-memory array realizing adaptive data converters and charge-domain computing," *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, 2021, pp. 248–250.

[12] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *arXiv preprint arXiv:1510.00149*, 2015.

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

[14] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-tile 2.4-Mb in-memory-computing CNN accelerator employing charge-domain compute," in *IEEE Journal of Solid-State Circuits*, vol. 54, no. 6, pp. 1789–1799, Jun. 2019.

[15] S. Yin, Z. Jiang, J.-S. Seo, and M. Seok, "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks," in *IEEE Journal of Solid-State Circuits*, vol. 55, no. 6, pp. 1733–1743, 2020.

[16] X. Si *et al.*, "24.5 A twin-8T SRAM computation-in-memory macro for multiple-bit CNN-based machine learning," *2019 IEEE International Solid-State Circuits Conference (ISSCC)*, 2019, pp. 396–398.

[17] Y.-D. Chih *et al.*, "16.4 An 89TOPS/W and 16.3TOPS/mm^2 all-digital SRAM-based full-precision compute-in memory macro in 22nm for machine-learning edge appli-

doi:10.6342/NTU202202019

cations," *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, 2021, pp. 252–254.

[18] H. Kim, T. Yoo, T. T.-H. Kim, and B. Kim, "Colonnade: A reconfigurable SRAM-based digital bit-serial compute-in-memory macro for processing neural networks," in *IEEE Journal of Solid-State Circuits*, vol. 56, no. 7, pp. 2221–2233, Jul. 2021.

[19] A. Biswas and A. P. Chandrakasan, "Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications," *2018 IEEE International Solid-State Circuits Conference (ISSCC)*, 2018, pp. 488–490.

[20] S. K. Gonugondla, M. Kang, and N. Shanbhag, "A 42pJ/decision 3.12TOPS/W robust in-memory machine learning classifier with on-chip training," *2018 IEEE International Solid-State Circuits Conference (ISSCC)*, 2018, pp. 490–492.

[21] J.-W. Su *et al.*, "15.2 A 28nm 64Kb inference-training two-way transpose multibit 6T SRAM compute-in-memory macro for AI edge chips," *2020 IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 240–242.

[22] 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," *2022 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 1–3.

[23] J.-S. Park *et al.*, "A multi-mode 8k-mac hw-utilization-aware neural processing unit with a unified multi-precision datapath in 4nm flagship mobile soc," *2022 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 246–248.

[24] C. Baugh and B. Wooley, "A two's complement parallel array multiplication algorithm," in *IEEE Transactions on Computers*, vol. C-22, no. 12, pp. 1045–1047, 1973.

[25] V. Oklobdzija, D. Villeger, and S. Liu, "A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach," in *IEEE Transactions on Computers*, vol. 45, no. 3, pp. 294–306, 1996.

doi:10.6342/NTU202202019

[26] J. Yue *et al.*, "A 65nm 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," *2020 IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 234–236.

[27] J. Yue *et al.*, "15.2 A 2.75-to-75.9TOPS/W computing-in-memory NN processor supporting set-associate block-wise zero skipping and ping-pong CIM with simultaneous computation and weight updating," *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, 2021, pp. 238–240.

[28] S. Yin *et al.*, "PIMCA: A 3.4-Mb programmable in-memory computing accelerator in 28nm for on-chip DNN inference," *2021 Symposium on VLSI Circuits*, 2021, pp. 1–2.

[29] H. Jia, H. Valavi, Y. Tang, J. Zhang, and N. Verma, "A programmable heterogeneous microprocessor based on bit-scalable in-memory computing," in *IEEE Journal of Solid-State Circuits*, vol. 55, no. 9, pp. 2609–2621, Sep. 2020.

[30] J.-H. Kim, J. Lee, J. Lee, H.-J. Yoo, and J.-Y. Kim, "Z-PIM: An energy-efficient sparsity aware processing-in-memory architecture with fully-variable weight precision," *2020 IEEE Symposium on VLSI Circuits*, 2020, pp. 1–2.

[31] H. Jia *et al.*, "15.1 A programmable neural-network inference accelerator based on scalable in-memory computing," *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 236–238.

[32] 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," *2022 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 1–3.

doi:10.6342/NTU202202019