ARCHITECTURE AND CIRCUIT DESIGN OPTIMIZATION FOR COMPUTE-IN-MEMORY

A Dissertation
Presented to
The Academic Faculty

by

Hongwu Jiang

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2022

© 2022 Hongwu Jiang

ARCHITECTURE AND CIRCUIT DESIGN OPTIMIZATION FOR COMPUTE-IN-MEMORY

Approved by:

Dr. Shimeng Yu, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Suman Datta
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Shaolan Li
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Hyesoon Kim
School of Computer Science
Georgia Institute of Technology

Dr. Asif Islam Khan
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: September 28, 2022

Dedicated to my beloved family:

To my parents, Dejun Jiang and Hong Su

and to my wife, Shanshi Huang

for their boundless love and support.

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest appreciation to my advisor, Dr. Shimeng Yu, for his persistent guidance, advice, and support throughout my Ph.D. study. He has been a fantastic mentor all the time. His patience, enthusiasm, and immense knowledge helped me overcome the challenges throughout my research. I will endeavor to emulate Dr. Yu's brilliance and positive attitude towards research and mentorship.

I would also like to extend my sincere gratitude to my committee professors, Dr. Shaolan Li, Dr. Asif Islam Khan, Dr. Suman Datta, Dr. Hyesoon Kim and Dr. Arijit Raychowdhury, for their valuable suggestions for this dissertation.

I am also grateful to Dr. Francky Catthoor, Dr. Stefan Cosemans, Dr. Meng-fan Chang, and Jian-Wei Su for our exciting collaborations. I am also thankful to Dr. Jan Rabaey and Amandeep Singh for mentoring me through my internship at IMEC.

Many thanks to all my labmates and my friends, Dr. Xiaoyu Sun, Dr. Xiaochen Peng, Dr. Panni Wang, Dr. Wonbo Shim, Dr. Jae Hur, Dr. Sola Woo, Wantong Li, Gihun Choe, Yandong Luo, Anni Lu, Yuan-Chun Luo, Chinsung Park, Po-Kai Hsu, Janak Sharda, Jungyoun Kwak, James Read, Vaidehi Garg, Omkar Phadke, Junmo Lee, who provided collaboration and assistance during my study.

Last but not the least, my deepest thanks go to my parents Dejun Jiang and Hong Su, as well as my wife Shanshi Huang for their love, understanding, and constant support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The main objective of this thesis is to optimize computing-in-memory (CIM) design for accelerating Deep Neural Network (DNN) algorithms. As compute peripheries such as analog-to-digital converters (ADCs) introduce significant overhead in CIM inference design, the first part of the research focuses on circuit optimizations for in-memory computing. In the first work, we comprehensively explore the tradeoffs involving different types of ADCs and investigate a new ADC design especially suited for the CIM, which performs the analog shift-add for multiple weight significance bits, improving the throughput and energy efficiency under similar area constraints. In the second work, we propose a resistive random access memory (RRAM) based ADC-free in-memory compute scheme validated with a prototype chip in TSMC 40nm process, which can significantly improve the hardware performance over the conventional CIM designs while achieving near-software classification accuracy on ImageNet and CIFAR-10/-100 dataset.

In the second part of the thesis, the research focuses on hardware support for CIM on-chip training. To maximize hardware reuse of CIM weight stationary dataflow, we propose the CIM training architectures with the transpose weight mapping strategy. The cell design and periphery circuitry are modified to support bi-directional computing efficiently. A novel solution of signed number multiplication is also proposed to handle the negative inputs in backpropagation. Based on the silicon measurement data on a two-way SRAM-based prototype chip in TSMC 28nm process, we comprehensively explore the hardware performance for the entire SRAM-based architecture for DNN on-chip training.

CHAPTER 1 INTRODUCTION

1.1 Motivations

In the recent decade, deep neural network (DNN) based machine learning (ML) algorithms have made remarkable achievements in artificial intelligence (AI) applications such as image/speech recognition and autonomous driving. The rapid growth of those AI applications is greatly facilitated by the continuing development of ever more powerful hardware. Traditionally, AI-related computations are handled by central processing units (CPUs) or graphic processing units (GPUs). However, those DNN-based algorithms are computationally expensive on conventional computing platforms as intensive vector-matrix multiplications (VMM) are involved. The traditional von Neumann architecture inherently limits the parallelism of DNN algorithms since massive data movement happens between the computing and storage units, resulting in low energy efficiency and speed.

Therefore, the success of DNN algorithms, in turn, boosted the development of hardware accelerators to be deployed from cloud to edge. A variety of application-specific integrated circuit (ASIC) designs based on silicon complementary metal-oxide-semiconductor (CMOS) technology (e.g., TPU [1], Eyeriss [2]) have been proposed to accelerate AI workloads. However, while showing superior performance compared to traditional general-purpose processors, the memory wall problem remains in such near-memory designs where the weights and intermediate data still require inefficient on-chip or off-chip memory access. To this end, compute-in-memory (CIM), where information can be processed and stored at the same locations, is emerging as an efficient paradigm to address the memory wall bottleneck. The crossbar-like memory array is usually employed

in CIM to store the values of the weight matrix, where the weights are mapped as the conductance of the memory cells [3]. The multiply-and-accumulate (MAC) operations can be performed in parallel: the input activates multiple rows, and the products between the inputs and the weights are summed up along the columns as MAC output.

Though in-memory computing is promising with enhanced parallelism, grand challenges exist in designing CIM accelerators. Firstly, the performance of CIM accelerators is still limited by the data conversion procedure. Analog-to-digital converters (ADCs) are typically employed in CIM to convert the partial sums to the digital signals for further processing steps such as activation function/pooling. If one implements enough ADCs to guarantee high throughput, the system's total power tends to be dominated by the ADCs rather than the memory array itself. On the contrary, if multiple columns share ADC in one array, the computing throughput will be restricted as a penalty. Consequently, ADCs have been identified as the primary bottleneck for power dissipation and area overhead in the CIM accelerators. Besides, ADC quantization loss may hamper the computational accuracy performance. Hence, at the circuit level, more economical peripheral circuits are preferred to unleash the benefits of CIM. Secondly, most of the CIM architectures or macros proposed so far could support the inference only. However, exploiting training functionality has become desired and essential. For edge devices, supporting on-chip training means adaptiveness to the local environment and fast response to new scenes. Besides, the privacy concerns of sharing personal data to the cloud could be eliminated with local training. Therefore, an adaptive and continuous learning mode is preferred for the edge device.

1.2 Thesis Overview

In this research, we analytically investigate different ADC designs for CIM arrays [4]. The experiment results show that 6-bit ADC precision is sufficient to guarantee no loss of accuracy for a large array (512×512). Compared to traditional ADC typologies such as Flash-ADC and SAR-ADC, the analog shift-add ADC achieves higher throughput and energy efficiency with less area overhead. To further reduce ADC overhead, we propose an *E*fficient *N*eural *N*etwork *A*ccelerator, namely ENNA, which eliminates ADCs at the array level, implementing inter-array data processing in an analog manner [5]. An ADC-free prototype chip is taped-out with TSMC's 40nm RRAM process [6]. The ADC-free chip features 1-8bit configurable precision. The impact of temporal/spatial variations is comprehensively evaluated to ensure accuracy performance. The simulation and measurement results show significant improvement in hardware performance both at macro and system levels.

This research comprehensively explores the hardware implementations of CIM on-chip training. We propose a transpose SRAM-based *CIM A*rchitecture for multi-bit precision DNN *T*raining, namely CIMAT, with two different bit-cell designs, and explore the corresponding weight mapping strategies, dataflow and pipeline design [7]. The experiment results reveal that CIM is a promising solution to implement on-chip DNN training, which can reduce off-chip talk significantly. To enable the use of compact-rule SRAM cells, we propose a two-way SRAM-array-based accelerator for DNN on-chip training [8]. A new signed number multiplication approach is proposed to perform backpropagation efficiently. The proposed two-way SRAM array macro is fabricated and

validated in TSMC 28nm process [9] (as a collaboration work with NTHU Dr. Meng-fan Chang's group). Based on the silicon measurement data on CIM macro, we comprehensively explore the hardware performance for the entire SRAM-based architecture, achieving ~3.2 TOPS/W on ImageNet dataset training.

The rest of the thesis is organized as follows: Chapter 2 introduces the principle of Compute-in-Memory and a comprehensive survey of the recent progress in the CIM design for DNN acceleration, focusing on macro-level and architecture-level demonstrations. Chapter 3 presents ADC design exploration for the CIM array. Chapter 4 presents the "ADC-free" CIM accelerator design for inference application, including an RRAM-based prototype chip and system-level evaluation. Chapter 5 presents the architecture-/circuit-level hardware implementations to support on-chip training in CIM. Chapter 6 presents a two-way SRAM-based accelerator design for DNN training. Finally, Chapter 7 summarizes the research presented in this dissertation and provides the potential directions for the future.

CHAPTER 2 BACKGROUND

2.1 CIM Basics

The idea of embedding computations into memory is not new. Compute-in-Memory (or Process-in-Memory) was initially proposed in the 1970s. The following works [10] [11] explored various methods to implement in-memory logic operations on standalone dynamic random access memory (DRAM). However, the incompatibility of the DRAM and logic manufacturing techniques made the concept of integrating logic with memory banks impracticable. In recent years, thanks to the CMOS technology scaling and innovations on emerging non-volatile memories (eNVMs), on-chip memory capacity is increasing rapidly (e.g., 256Mb SRAM [12], 8Mb RRAM [13]). Accordingly, researchers are developing CIM architectures with on-chip embedded memories.

In-memory computing has two potential advantages: 1) increasing the computing efficiency of a variety of functions, including Boolean logic operations (e.g., OR, AND), basic arithmetic operations (e.g., addition, multiplication), and linear algebra operations (e.g., dot products, matrix multiplication); 2) reducing the amount of data transfer to save both time and energy. CIM solutions have been demonstrated in various applications, ranging from scientific computing, image processing, hardware security, and spiking neural network (SNN) to deep learning inference/training. Among all, CIM's most representative application scenario is deep learning (DL) acceleration. The state-of-the-art DL algorithms require a large number of computational resources and memory storage as the size of deep neural networks (DNNs) increases dramatically (e.g., ResNet-152 for ImageNet has 60M parameters [14]). The most important operation of DNN processing is

the vector-matrix multiplication (VMM) between the input vector and the weight matrix, which could be executed as multiply-and-accumulate (MAC) operations. To this end, CIM designs for DL could benefit from both reduced memory access and the efficiency of parallel computing. This research mainly discusses CIM implementations for accelerating DNNs.

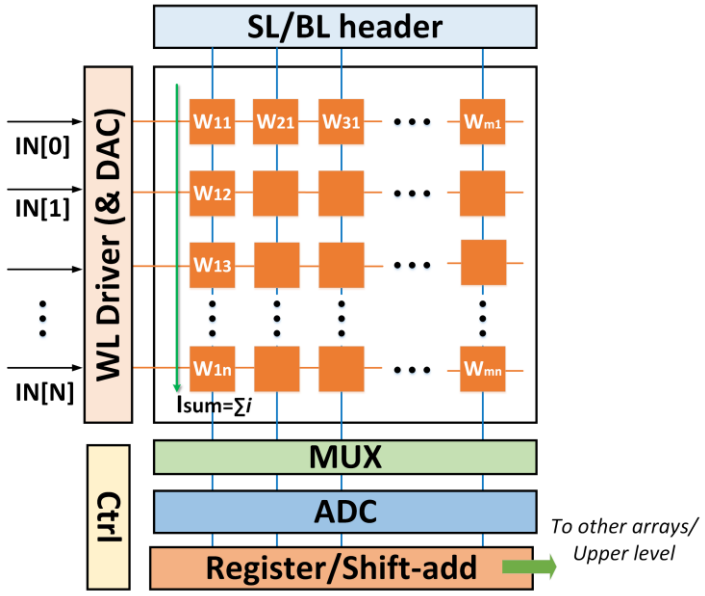


Figure 1: Generic CIM crossbar array structure

A generic structure of CIM crossbar array is shown in Figure 1, performing the multiply-and-accumulate (MAC) operation with perpendicular input rows and output columns. The memory cell is represented by the orange box, which could store binary or multi-bit weight according to device characteristics. The input vector is loaded in parallel to the rows and multiplied by weight to generate products. The current summation along columns represents the final MAC output in analog. Analog-to-digital conversion is normally required to quantize the analog MAC outputs to binary bits, providing scalability and flexibility for the mixed-signal communication between the sub-arrays and the upper

level. Digital shift-adds are usually equipped for high-precision operations. In practice, only a part of rows/columns could be synchronously turned on due to limited ADC resolution, device non-ideal effects, or the layout mismatch between the column pitch and the peripheries.

Table 1: Survey of recent chip-level demonstrations of SRAM-/RRAM-based CIM

Affiliation	Year	Memory technology	Technode (nm)	Cell structure	Energy efficiency (TOPS/W)	Dataset	Accuracy	Inference or training
Princeton [15]	2016	SRAM	130	6T	57.9	MNIST	91%	Inference
NTHU [16]	2018	SRAM	65	Split-6T	111.6	MNIST	97.5%	Inference
NTHU [20]	2019	SRAM	55	Twin-8T	734.8	CIFAR-10	90.42%	Inference
NTHU [9]	2020	SRAM	28	6T+Trans. cell	972	CIFAR-10	91.94%	Training
TSMC [21]	2021	SRAM	7	8T	5136	MINIST	98.6%	Inference
TSMC [22]	2021	SRAM	22	6T	1424	N/A	N/A	Inference
NTHU [26]	2018	RRAM	65	1T1R	38.4	MNIST	98%	Inference
NTHU [27]	2019	RRAM	55	1T1R	131.4	CIFAR-10	88.5%	Inference
ASU [29]	2018	RRAM	90	1T1R	24.1	CIFAR-10	83.5%	Inference
Gatech [30]	2021	RRAM	40	1T1R	56.67	ImageNet	54%	Inference

Note: the energy efficiencies are measured for 1-bit operations

2.2 Compute-in-memory Macro Designs

Compared to Von Neumann architecture, the development of CIM design demands more cross-layer optimizations from circuit- and device-level to architecture-level. This section mainly introduces state-of-art CIM circuit and device techniques. CIM architectural implementations will be discussed in the next section.

Table 1 surveys the recent macro-/chip-level demonstrations based on CIM. Theoretically, CIM could be implemented by any memory device technology, which can be roughly sorted into two categories: SRAM-based and eNVM-based. SRAM is considered as a mature candidate from the technology availability perspective. In 2016, as

a pioneering work of SRAM-based CIM, J. Zhang et al. [15] modified the standard 6T SRAM array to perform in-memory classification. In compute mode, the bitline pair (BL/BLB) is pre-charged first. The analog voltages on wordlines (WLs) are generated by DACs, representing the input vector. When multiple rows are turned on, currents from one column are summed together on BL/BLB. Finally, BL and BLB will decay from VDD with different rates, and a comparator is enabled, providing sign thresholding. In 2018, a split-6T cell design [16] was proposed to support XNOR-Net [17], which splits the wordline into two control signals: WL and WLB. As the inputs to WL and WLB are always complimentary, only one side of the SRAM column will be turned on, achieving 50% energy saving with nominal overhead for additional WL routing, compared to the conventional 6T cell design. However, 6T-based cell design suffers read disturbance when multiple rows are activated simultaneously since stored data could be flipped by large discharge current. Thus, the subsequent CIM designs try to decouple compute and write operations by modifying the cell design such as 8T cell [18] and 12T [19] cell. X. Si et al. [20] proposed an advanced 8T cell design, which combines two read-decoupled 8T cells (M8T, L8T) into a twin-cell. Such twin-8T cell could represent 2-bit since the transistor width of the read path in M8T is twice that in L8T, delivering 2× cell current. Furthermore, WL voltage is programmed with a 3-level magnitude to represent the 2-bit input. Consequently, this Twin-8T design enables multi-bit MAC operation in one compute cycle. A recent design implemented efficient training by designing a transposable SRAM array, which could support bidirectional MAC operations [9]. Each processing unit includes one transpose-multiply-cell (TMC) and 16 regular 6T SRAM cells. Each TMC consists of 10 transistors including two pass-gate transistors and two multiply branches. TMC has two

read modes to support bi-directional bitwise multiplication so that forward and backward propagation calculations can be implemented in the same array. Recently, TSMC implemented a CIM macro based on 8T cell design in 7nm tech node [21], showing the technology scalability of SRAM-based CIM. Instead of tuning the WL voltage, they encoded the number of pulses as the multi-bit input and hired capacitors to perform binary-weighted computation by charge sharing. In this design, a 4-bit Flash-ADC is shared by multiple columns for quantizing the outputs. An all-digital CIM design was recently presented in [22]. Other than the mixed-signal manner necessitating an ADC for digitization, this digital CIM design performs the accumulation in digital with extra digital accumulation circuitry inside the SRAM macro. Such a digital in-memory computing scheme eliminates ADCs while benefiting from CMOS scaling and lower Vdd operation.

Compared to SRAM, eNVMs, such as RRAM [23], PCM [24], and FeFET [25], are preferred for power-constrained edge devices due to their non-volatility, enabling instant-on computation. Among eNVMs, RRAM is widely used in many prototype chips due to its relatively large on/off ratio, small power consumption, and more foundry availability. In 2018, W. H. Chen et al. [26] fabricated a binary-input/ternary-weight RRAM-based CIM design in 65nm. Positive weights and negative weights are stored in separate macros. Digital quantization is implemented by 3-bit distance-racing current-mode sense amplifiers (CSAs), which could improve sensing margin and reduce input offset. To reduce the errors caused by ADC references, a reference array is hired to generate the input-aware references dynamically. As a continuing work, an advanced RRAM-based CIM design [27] was proposed in 2019, which could further suppress input offset by a novel triple-margin CSA design. In 2020, C. X. Xue et al. [28] further optimized the CSA design, which could yield

2b MAC output simultaneously. Besides, rather than a WL-in computing scheme, they clamped BL current to implement multi-bit input, shortening access time. One limitation of such current sensing is that only limited rows could be activated simultaneously due to large BL current. Instead, XNOR-RRAM [29] first converted the current output to voltage and then quantized it by voltage-mode Flash-ADC, which allows 64 rows to be opened simultaneously. Besides, this design hired two adjacent 1-transistor-1-resistor (1T1R) cells to represent a ternary weight rather than employing two macros to store negative and positive weight separately. To eliminate the sensing errors caused by the limited on/off ratio, J. H. Yoon et al. [30] proposed an active-feedback-based voltage-sensing approach to linearize readout BL voltage. In this design, an on-chip write verification approach is also proposed to tighten the RRAM resistance distribution.

Table 2: Survey of recent CIM Architectures and their characteristics

Name	Year	Memory technology	Data Precision	In-memory operation	ML algorithm	Support model
ISAAC [31]	2016	RRAM	16-bit fixed	Analog MAC	CNN Inf.	VGG
PRIME [32]	2016	RRAM	6-bit IN./8-bit W.	Analog MAC	CNN Inf.	MLP, VGG
Neural Cache [33]	2018	SRAM	8-bit	Add/Sub, Mul.	CNN Inf.	Inception V3
TIME [34]	2019	RRAM	8-bit	Analog MAC	CNN Inf. & Train	VGG
Pipelayer [35]	2017	RRAM	16-bit	Analog MAC	CNN Inf. & Train	VGG, AlexNet
CMP-CIM [37]	2018	SOT-MRAM	8-bit	Comparison	CNN Inf.	CMP-NET
X. Peng et al. [36]	2019	RRAM	8-bit	Analog MAC	CNN Inf.	VGG

2.3 Compute-in-memory Architecture Designs

This section presents an overview of the state-of-the-art research into DL-oriented CIM architecture designs, summarized in Table 2. Since DNN training process is more

complicated than inference, most early-stage CIM designs focus on supporting inference only. ISAAC [31] mapped the weights into crossbar arrays as the conductance of each memory cell, while the activations will be fed into each row as analog voltages encoded by the digital-to-analog converter (DAC). Weights are represented in 2's complement for both positive and negative values. Products in the form of current from each cell will be summed up along columns, and digitized by ADCs for the following processing. Given the nature of the crossbar array, the authors directly flatten each 3D kernel into a long column. Accordingly, all the kernels in one convolutional layer are converted to a large weight matrix. In this way, products along the same column will be summed up, representing the final output. Array partitioning is necessary considering the large size of the convolutional layers, thus ISAAC architecture is organized with multiple tiles assigned for different layers. Besides, an inter-layer pipeline is proposed to reduce the buffering requirement and improve system throughput. Compared with the prior near-memory-processing solutions, ISAAC achieved significant improvement of throughput, energy saving and computational density. Similar to the ISAAC, PRIME [32] is also a CIM architecture focusing on DNN inference. In PRIME design, RRAM is chosen as on-chip memory, and three types of memory array design are presented for different purposes: computing, storage and buffering. Instead of hiring traditional DACs and ADCs, periphery circuits such as sense amplifiers and write drivers are modified to serve storage and computing functions, reducing the area overhead. Unlike ISAAC, separate crossbars are dedicated to positive and negative weights in PRIME architecture. C. Eckert et al. [33] proposed an SRAM-based CIM engine, Neural Cache, for DNN inference. It can implement bitwise AND/NOR operations inside cache memory by simultaneously activating the operand rows. The

compute SRAM array is capable of bit-serial addition and multiplication. A transpose 8T SRAM bit-cell design enables data reorganization in bit-serial format. TIME [34] is a modified RRAM-based CIM architecture based on PRIME, supporting the training process. To fine-tune RRAM states efficiently, authors also presented several methods, including one-direction write scheme, smart refresh strategy, and variability tuning scheme, which could reduce RRAM non-ideal effects. TIME improved the energy efficiency by a factor of 126 compared to the GPU platform. Another RRAM-based design named PipeLayer [35] was presented for training support, in which the RRAM memory is separated into two types: morphable subarrays and memory arrays. The results of the feedforward process are stored in a memory array for future backward computation and a morphable array is used as both compute unit and storage. Their design exploits both intra- and inter-layer parallelism to implement pipelined training. Compared to GPU, their design could achieve 42.45× and 7.17× improvement in speed and energy efficiency, respectively. X. Peng et al. [36] proposed a novel weight mapping strategy to maximize the input data reuse in CIM architectures, which could be treated as kernel-splitting mapping. Unlike the kernel-flatten mapping method, the weights at different dimensional locations of each kernel are mapped into different processing units (PEs). Based on this kernel-splitting mapping method, which allocates the input data into different PEs, CIM architecture could efficiently reuse the input and weight data, yielding 2.1× speed-up and 17% energy saving compared with the prior mapping method. S. Angizi et al. [37] proposed a hardware-oriented CIM accelerator CMP-CIM that performs comparator-based DNN inference. Here the computationally-intensive convolution operations in Convolutional Neural Network (CNN) algorithm are replaced by a combination of comparison and addition operations.

CMP-CIM could achieve ~ 94× and 3× better energy efficiency than CNN and Local Binary CNN (LBCNN) tested on the SVHN dataset. It should be noted that the evaluation results of prior architectural CIM works are obtained mostly from simulations.

CHAPTER 3 ADC DESIGN EXPLORATION FOR CIM ACCELERATOR

3.1 Introduction

As aforementioned, in most of today's CIM designs, the memory array is equipped with an analog-to-digital converter (ADC) to convert analog MAC values to digital outputs. These outputs will be passed on to the peripheral circuitry for further processing steps such as activation function/pooling in the digital domain, and then sent to the next array as the input. This mixed-signal computing scheme offers scalability toward multiple-array designs via interconnect buses or network-on-chip, while introducing significant power dissipation and area overhead in the necessary data conversion at the array outputs. For illustration, the ISAAC architecture [31] reports that 58% of the total power is consumed by ADCs, occupying 31% of the total area. As reported in PUMA accelerator [38], high-precision ADCs consume ~ 60% of the total system energy and occupy ~80% of the chip area. W. He et al. [39] also indicated that ADC dominates the crossbar area and energy. To reduce the overhead of ADCs, there are two straightforward approaches. One is to limit the number of ADCs employed in one array, which means multiple columns share one ADC. As a penalty, the parallel computing throughput is reduced. This method could reduce the area overhead of ADCs while the total energy consumption will not be reduced. The other solution is to lower the ADC precision, which could benefit both energy and area overhead. However, the quantization loss of partial sum may hamper the inference accuracy performance. The acceptable precision reduction could differ from one design to another and must be explored accordingly. Furthermore, different ADC topologies differ in

hardware overhead under the same precision settings. Therefore, the choice of ADC topologies and configurations is critical to designing the CIM architecture under hardware constraints.

Figure 2 (a) demonstrates mapping a weight matrix to a conceptual resistor-based crossbar array for MAC operations. For the computation, the weight W_{ij} is encoded to the memory cell conductance G_{ij} with the neuron activations X_i converted to voltages V_{xi} and applied to the crossbar rows in parallel. The current from different rows will be summed through the column (I_i) and could be further converted to an analog voltage by a resistor divider or transimpedance amplifier (TIA). In this way, ADCs can quantize the analog outputs in voltage mode. Figure 2 (b) shows the conventional dataflow of multi-bit MAC operations in CIM. Binary input vectors with different significance are fed into the CIM array cycle by cycle. Multiple columns are combined to represent a fixed-point weight with different significant bits grouped into different columns separately. Therefore, the CIM array uses two shift-add processes to finish the MAC with multi-bit inputs and weights: one is to weigh and sum up partial sums across different significant input bits, while the other is across different significant weight bits. This work will not further discuss the input shift-add process since it is identical for different ADC topologies. On the contrary, we focus on improving the shift-add process for weight through ADC techniques. As shown in the diagram, we divide the shift-add process for weight into two phases: analog MAC and ADC in a single column (phase I) and digital shift-add between columns (phase II). In other words, Phase II accumulates the weighted sum from the least significant bit (LSB) and the most significant bit (MSB). This phase is usually realized by employing a digital shift-add module in the CIM periphery.

The rest of this chapter is organized as follows: Section 3.2 explores different types of ADCs and investigates a new ADC design especially suited for the CIM context. Section 3.3 presents the impact of quantization loss and comprehensive hardware evaluations for different ADC topologies. Section 3.4 summarizes the chapter.

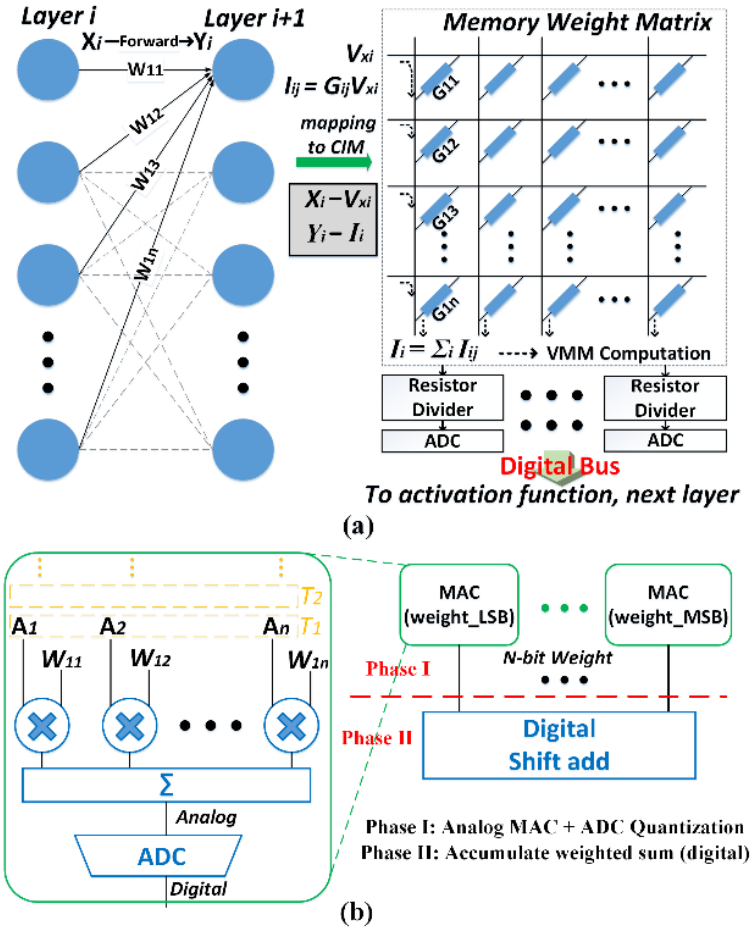


Figure 2: (a) Weight matrix mapping in a crossbar array. (b) Generic dataflow of multi-bit MAC operations.

3.2 ADC Topology Design Exploration

3.2.1 Conventional ADC choices for CIM application

It is important to have a compact ADC design in the CIM array considering the area/energy efficiency. Two ADC topologies are popular in prior CIM works, which are Flash-ADC [40] and successive-approximation-register (SAR)-ADC [26], because of their simplicity and suitability for low-to-medium precision (i.e., < 8 bit). Figure 3 presents the principle of Flash-ADC and SAR-ADC. For an N-bit converter, the Flash-ADC employs $2^N - 1$ cascading comparators to generate a thermometer from the analog signal. A encoder is needed to convert this thermometer code to the digital binary output. Due to the parallel sensing, Flash-ADC is the fastest ADC design in principle. However, its power consumption and area are exponentially proportional to its precision. Generally, Flash-ADC outperforms SAR-ADC for lower precision (3-bit or below), while SAR-ADC is preferred under higher precision. Flash-ADC of 3-bit precision has been demonstrated in the recent CIM macros [18] with small-scale arrays (64–128 rows). However, its scalability toward large-scale arrays remains unexplored.

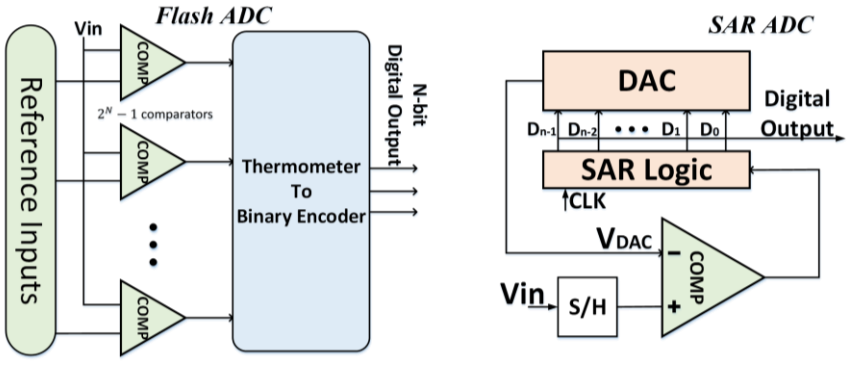


Figure 3: Principle of Flash-ADC and SAR-ADC.

On the contrary, only one comparator is employed in SAR-ADC to perform a one-bit comparison in each internal clock. Several comparisons will be made in serial based on the binary search algorithm. In other words, the output will be compared to a reference that

will be adjusted by the SAR logic (implemented with multistage shift registers) dynamically in a bit-by-bit fashion. After comparisons are made all the way from MSB to LSB, an N-bit digital output is available in the register. Typically, the voltage-mode SAR-ADC is adopted in the CIM design, which generates the analog reference voltage exploiting charge redistribution in a capacitive digital-to-analog converter (DAC) array. It is widely used in CIM [31] [38] [26] and reports lower area/energy overhead than Flash-ADC due to its simple structure.

In theory, the ideal ADC precision for VMM computation in a $N \times N$ CIM array with i input precision per compute cycle and j weight precision per cell is $(\log_2 N) + i + j$. However, thanks to data sparsity in DNNs, the CIM array can adopt lower-than-ideal ADC precision to lower ADC overhead and improve performance [29] [41].

3.2.2 Analog Shift-add ADC

As discussed, multiple columns have to share one ADC due to the tight area constraint for ADC in the CIM array. This time-multiplexing will reduce the throughput of the array consequently. Besides, it will also introduce additional multiplexer (MUX) and digital shift-add circuits. Alternatively, an analog shift-add approach was demonstrated in [20] to reduce the ADC overhead. Figure 4 (a) shows the principle of analog shift-add ADC, which modifies the shift-add process across weight bits. Here, phase II of the process is moved to the analog domain prior to the ADC. The analog MAC outputs from each column will be weighed and summed in multiple internal clock cycles. The preshifted and added MAC value, which already contains the weight significance, will be converted to the final digital output by a regular SAR-ADC. This way, the MUXs across columns and

digital shift-add module for multi-bit weight computation are eliminated. Therefore, throughput and energy efficiency will be improved under the same area constraint. On the other side, this pre-ADC accumulation will increase the full precision of the analog signal and thus has a potential precision impact on the following ADC. Both aspects will be discussed in more detail in the following section.

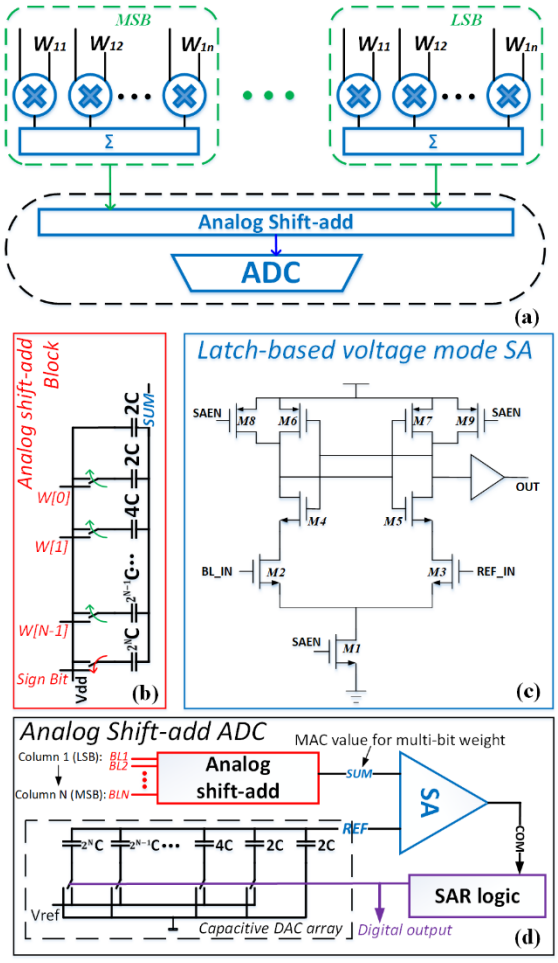


Figure 4: (a) Alternative CIM dataflow which performs weighted sum before ADC. (b) Schematic of analog shift-add block. (c) Schematic of latched-based sense amplifier. (d) Top-level structures of analog shift-add ADC.

The circuit schematic of the analog shift-add block is shown in Figure 4 (b). A capacitor array is adopted to perform the weighted accumulation in the analog domain,

exploiting its charge redistribution nature. This block is connected to multiple columns of weight bits with different significance from one fixed-point weight column. The 2's complement representation is adopted to encode the signed weights in DNN, where the significance is scaled by two, with the most significant bit being the sign bit. In the capacitor array, the capacitance of the capacitors in the block is increased exponentially from top to bottom, representing from LSB to MSB of the weight column. The switches are synchronously switched to enable the corresponding capacitor to be coupled to the power supply or the analog output from the corresponding column. Once the charge redistribution is stabilized, the analog shifted and added MAC value can be directly read out from the SUM port for the regular SAR-ADC process. A simple latch-based SA (Figure 4 (c)) is used as the comparator in SAR-ADC because of its low-power and high-speed advantages. In the illustrated SA, *M4–M7* transistors form a strong positive feedback amplifier and *M8–M9* transistors are used to pre-charge the output. *M2–M3* takes the inputs to the SA and can be treated as a common source differential amplifier. The sensing process will be enabled through the tail transistor *M1*. The top-level diagram of the analog shift-add ADC is shown in Figure 4 (d). To our best knowledge, while demonstrated in prior works, such analog shift-add ADC design was not thoroughly evaluated. Especially the impact on software accuracy performance and hardware overhead compared to conventional ADCs at the array level are not evaluated. Different ADC topologies may cause diversities on other periphery circuits and, thus, make the array-level hardware performance quite different from the ADC-only result where the rest of the array is ignored. This work comprehensively explores the ADC designs above for the CIM application, which will be shown in the next section.

3.3 Evaluation

3.3.1 ADC Quantization Impact

We first investigate the impact of ADC quantization loss on inference accuracy to decide the minimum ADC precision under acceptable software performance degradation. The ADC quantization is explored on a VGG-8 network on the CIFAR-10 classification. Generally, two algorithmic techniques can effectively reduce the ADC precision for partial sum quantization in the CIM operation: one is to cut the MSBs of the partial sum. Since the partial sum distribution is typically zero concentrated [18], appropriately reducing the range in the two tails, which could be done by cutting the MSBs, will not introduce significant accuracy loss. The other is the nonlinear quantization which adjusts the ADC references according to the partial sum's distribution. The first scheme is adopted in this work. Figure 5 shows the accuracy performance versus ADC precision for the conventional ADC designs (representing both flash and SAR) and the proposed analog shift-add ADC. Here, Flash-ADC and SAR-ADC make no difference in the conventional two-phase shift-add process for weight, as mentioned in section 3.1, so we group them together. Oppositely, the analog shift-add ADCs first weigh and sum up the 1-bit input with multi-bit weight MAC outputs in the analog domain. Then the final output containing the weight significance is quantized by a SAR-ADC. These two different quantization approaches are embedded in software simulation to evaluate the impact of ADC quantization loss on inference accuracy performance. The VGG8 network under investigation is trained with 4-bit weight, 8-bit activation, 8-bit gradient, and 8-bit error, following the WAGE (a framework that constrains weights (W), activations (A), gradients (G) and errors (E) of the

network to low-precision integers in both training and inference) method [42]. The software baseline accuracy for CIFAR-10 classification is 89%.

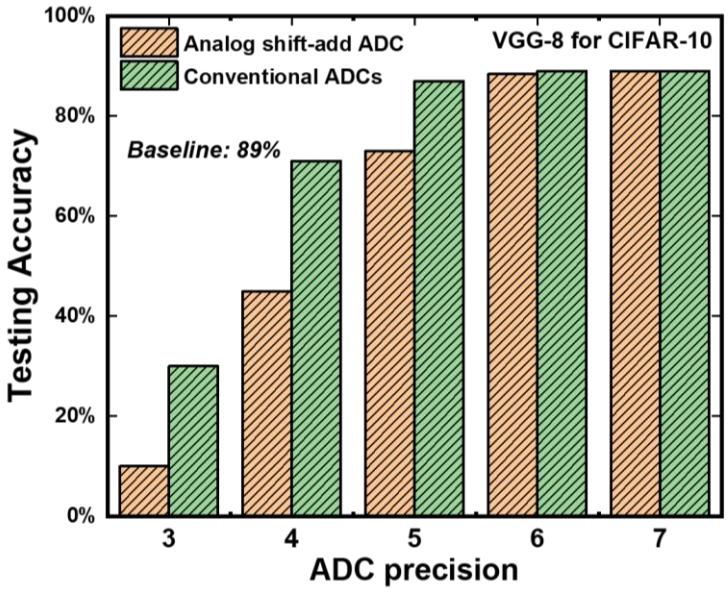


Figure 5: Accuracy performance vs. precision for different ADC designs.

A 512 × 512 memory array is assumed in this work, indicating a 9-bit full-precision partial sum will be generated from each column. From Figure 5, 6-bit precision is required to obtain the same accuracy as the baseline for the conventional weight shift-add process. In other words, 3-bit quantization loss is tolerable for the conventional case. Considering the analog shift-add ADC, the analog partial sum's full precision is increased to 13-bit (9-bit from column partial-sum and 4-bit from shift-add for weight precision). The results show that the accuracy degradation is negligible down to 6-bit analog shift-add ADC, tolerating 7-bit quantization loss. For the conventional process, partial sums from different significance are quantized first and then shift-added in the digital domain. Each partial sum suffers from quantization loss, and these quantization losses will be further accumulated. Oppositely, no quantization error will be introduced in partial sums during the shift-add in

the analog domain. The quantization loss happens in the final stage of the process. Thus the analog shift-add ADC could tolerate more ADC precision loss. In conclusion, the shift-add in analog could preserve more information and, thus, could bear higher ADC precision loss. In contrast, the digital shift-add after the ADC introduces residual information loss. As a result, it has a higher requirement in ADC precision to avoid big accumulated errors in the final output.

3.3.2 Hardware Evaluation

From our evaluation results, using ADCs with more than 6-bit resolution for both cases is an overkill, introducing hardware penalties for nothing. Meanwhile, the accuracy loss from 3-5 bits ADC is non-trivial under our configuration. However, 3-5 bits ADC is still viable by reducing the sub-array sizes (e.g., 128×128) or utilizing more aggressively quantized low-precision networks (e.g., XNOR-Net [17]). Thus, this work compared the ADC from 3-bit to 6-bit in our evaluation. The hardware performance of the three ADC topologies is evaluated based on a simulation program with integrated circuit emphasis (SPICE) simulation. As a case study, ferroelectric field-effect transistor (FeFET) arrays are adopted for CIM operations. FeFET is selected considering its elevated channel resistance (~500 kΩ) to potentially support the large array size (512 × 512) set in the evaluation. The FeFET bit cell size is assumed to be 4F by 4F, considering a relaxed channel length under high programming voltage. The simulation is done with a 40-nm low-power technode from a foundry process design kit (PDK). However, the presented ADC designs and evaluation results could be extended to other charge-based CIM schemes in other technology nodes and memory devices.

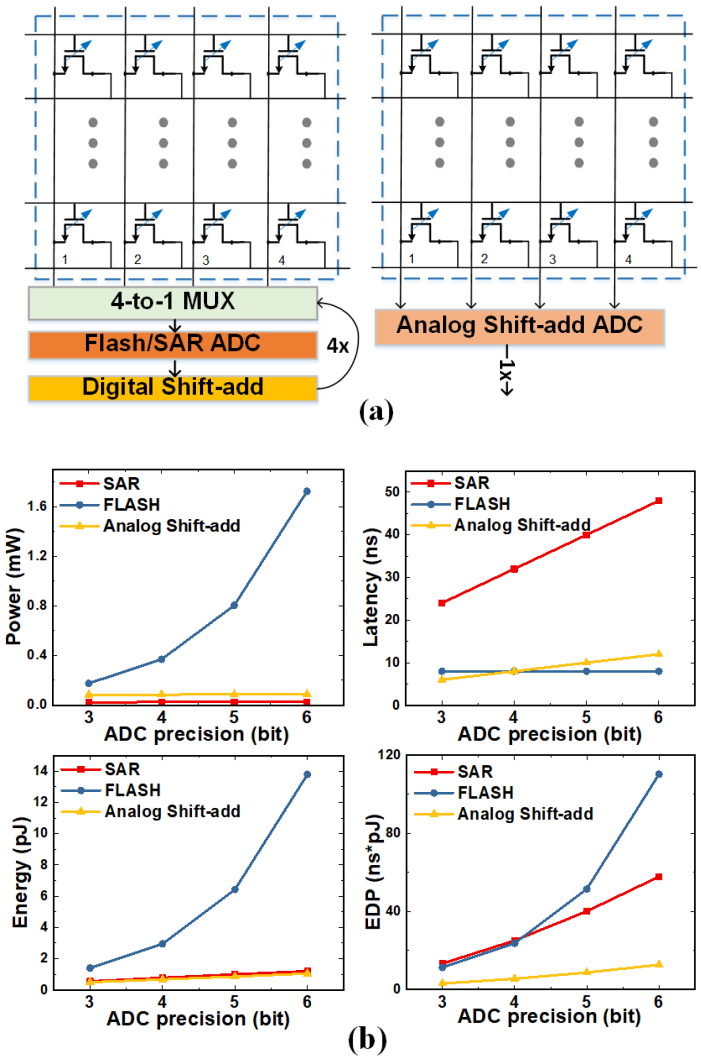


Figure 6: (a) Periphery configuration for different ADC designs. (b) ADC performance comparison.

First, the hardware performance of a single ADC is evaluated for different ADC topologies. For the one analog shift-add ADC, the multi-bit MAC results will be converted in one process step. On the contrary, in traditional dataflow, one ADC in CIM could only convert 1-bit input × 1-bit weight MAC. As a result, multiple ADCs are needed to work simultaneously to convert multi-bit MAC results in a single step. However, as discussed before, the ADC is shared among columns due to the area constraints. As a result, a MUX (Figure 6 (a)) is needed to switch from different significant weight bits, and the multi-bit

MAC results are generated in a multi-time ADC conversion process. Therefore, for a fair comparison, the hardware performance for the conventional process (Flash-ADC and SAR-ADC) should be evaluated with 4× single ADC conversion time (as 4-bit weight is used as case study). Figure 6 (b) shows the hardware performance comparison, including power, latency, energy, and energy-delay product (EDP) among the three ADC designs. Considering the power consumption, since the Flash-ADC has several SAs working in parallel, it consumes the most power. As the number of SAs increases exponentially with the ADC precision, so does the power consumption. On the contrary, since the number of SA works per cycle is always one, the power dissipation of SAR-ADC and analog shift-add ADC is nearly irrespective to ADC precision. As the ADC precision increases, the SAR-ADC and analog shift-add ADC need more cycles to complete the conversion, while Flash-ADC always adopts one cycle. Flash-ADC is obviously faster than SAR-ADC, and the difference increases with the ADC precision. While the analog shift-add ADC also adopts SAR-ADC for final conversion, one interesting observation is that its processing speed is comparable to Flash-ADC since it could complete multi-bit MAC in one conversion. The latency of this analog shift-add ADC will still linearly increase with ADC precision but with a much smaller slope, as shown in the latency plot. From the energy consumption plot, we can see that the growth of Flash-ADC is exponential, while the growth of SAR-ADC and analog shift-add ADC is linear. As Flash-ADC shows better throughput performance at the expense of energy consumption and the other two are opposite, a fair comparison among them is made with EDP, a well-defined metric that reflects the balance between energy consumption and throughput performance. In conclusion, the analog shift-add ADC has the best performance across 3–6 bits in this

module-level benchmarking for ADC comparison without considering CIM array-level constraints.

Table 3: ADC configuration under area constraints.

SAR ADC(bit)	ADC unit area(μm^2)	No. of columns share one ADC	ADC total area(μm^2)
3	99.52	16	4207.04
4	127.48	16	5357.12
5	163.60	16	5235.20
6	214.76	16	6872.32
FLASH ADC(bit)	**ADC unit area(μm^2)**	**No. of columns share one ADC**	**ADC total area(μm^2)**
3	144.12	8	9223.68
4	318.40	16	10188.80
5	668.96	32	10703.36
6	1375.19	64	11001.51
Analog Shift-add ADC(bit)	**ADC unit area(μm^2)**	**No. of columns share one ADC**	**ADC total area(μm^2)**
3	107.52	16	3440.64
4	143.48	16	4591.36
5	195.60	16	6259.20
6	278.76	16	8920.32

Area constraint: < 13421 μm^2

We further introduce area constraints at the array level to systematically compare ADC performance for CIM applications fairly. We optimize the number of columns sharing one ADC under a similar area constraint (i.e., 2× of memory array area, ~13,422 μm2 in our case study). The cost of digital shift-add is also included for multi-bit MAC operations for the traditional weight shift-add process. Table 3 shows the array-level ADC configuration under area constraint. The number of columns shared for 3-bit to 6-bit SAR-ADC and analog shift-add ADC maintains 16 since the area of a single ADC changes slightly as its precision increases. For Flash-ADC, we have to reduce the number of ADCs as ADC precision increases to satisfy the area constraint with exponentially increased unit ADC area. In other words, the column-sharing factor increases with ADC precision. Due

to the big area overhead of Flash-ADC, the total area of Flash-ADC is still much larger than that of SAR- and analog shift-add ADC, even with the less number of units employed.

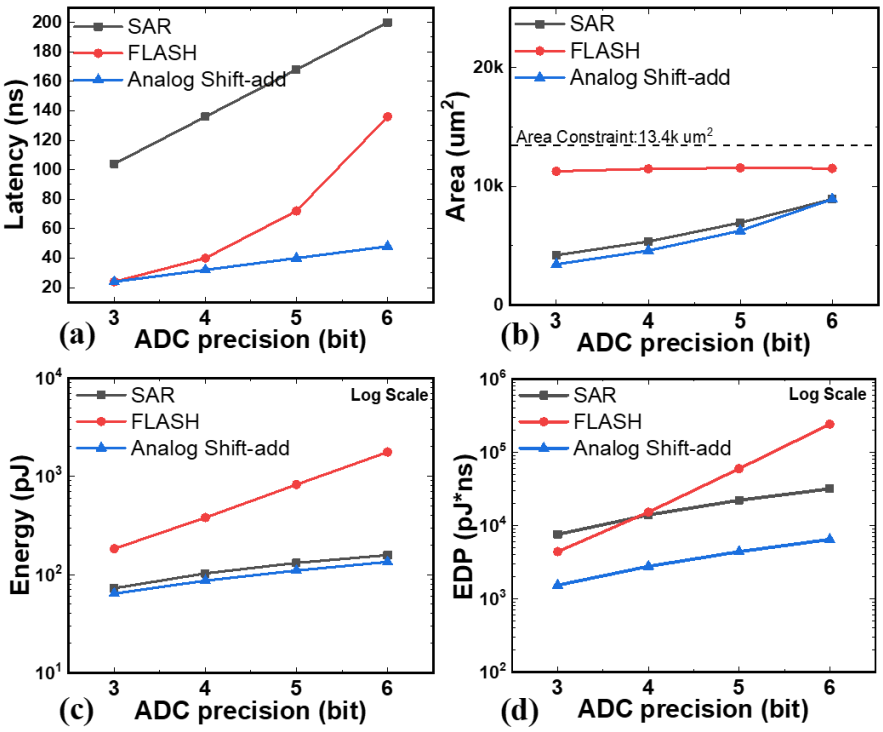


Figure 7: Array-level performance comparisons with possible ADC designs.

Figure 7 shows the array-level performance with ADC-related periphery included. As shown in the plot in Figure 7 (a), the analog shift-add ADC is always the fastest among the three topologies from 3-bit to 6-bit ADC precision. It outperforms Flash-ADC under high ADC precision since the multi-bit MAC operation is done in one cycle and the column-sharing factor is maintained. Unlike the ADC-only result, as the ADC precision goes high, the throughput difference between SAR-ADC and Flash-ADC decreases because of the reduced column parallelism of Flash-ADC under area constraint. Figure 7 (b) shows the flattened area overhead of Flash-ADC due to the increased column-sharing factor. The total area of the other two ADCs only slightly increases with ADC precision

with the constant column-sharing factor. Even at 6-bit, they consume a much smaller area than the Flash-ADC. The analog shift-add ADC still reports the lowest energy and EDP across the targeted precision range from 3-bit to 6-bit (Figure 7 (c) and (d)) at array-level under the area constraint. In conclusion, analog shift-add ADC is a promising solution for CIM applications thanks to the improved dataflow that performs multi-bit MAC operations before ADC.

3.4 Summary

This chapter explores different ADC typologies and investigated analog shift-add ADC design for CIM arrays. We first explores the effect of quantization loss on inference accuracy for different ADC designs and find that moving shift-add into the analog domain prior to ADC could improve the tolerance of ADC precision reduction. Then, the array-level performance with different ADC schemes is systematically evaluated, aiming to provide an understanding of the tradeoffs between hardware performance and area overhead. The VGG8 network for CIFAR-10 classification is evaluated for ADC quantization loss. According to the simulated results, 6-bit ADC precision is required for no accuracy degradation for a large array (512 × 512) for all ADC topologies. Under this 6-bit ADC precision, the analog shift-add ADC scheme achieves 37× and 4.9× higher EDP, compared to Flash-ADC, and SAR-ADC, respectively. The area footprint of analog shift-add ADC is comparable to that of SAR-ADC and only 0.77× that of Flash-ADC.

CHAPTER 4 ADC-FREE RRAM-BASED CIM ACCELERATOR DESIGN

4.1 Introduction

Due to the discrepancy between large DNN model sizes and the limited physical size of CIM sub-arrays, Input vectors and MAC outputs at the array level need to be encoded and processed through CIM-array peripheral circuits for communications. As shown in Figure 8, CIM array peripheries mainly comprise two parts: 1) input encoding; and 2) output processing.

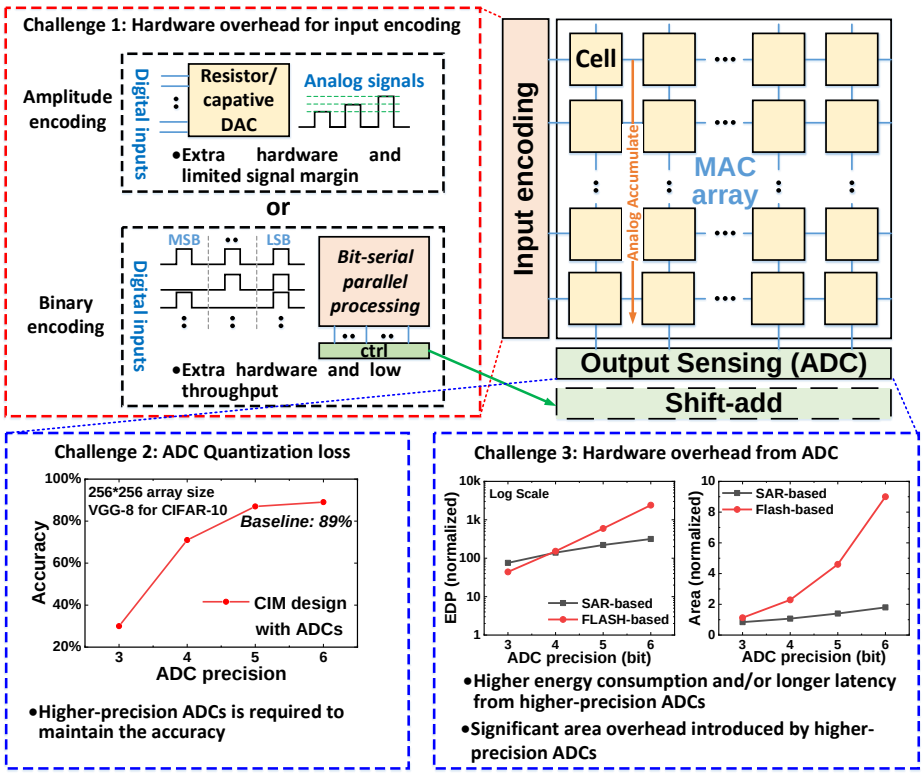


Figure 8: Hardware challenges in conventional CIM peripheries

The inputs of neural networks are usually mapped as voltage pulses in the CIM system. The popular input encoding methods in CIM designs are amplitude-based encoding and binary encoding. For amplitude encoding, digital-to-analog converters (DACs) are employed to encode the digital inputs as a voltage pulse of different amplitudes. Typical DAC designs, including capacitive circuits and resistive ladders, will introduce additional area overhead and power consumption. Besides, multi-level input voltage could also be implemented with multiple power supply sources to perform the multi-bit operation while increasing the load of the on-chip power unit. Although amplitude-based encoding theoretically could implement multi-bit operation in one cycle, the shortcoming is that the number of voltage levels is limited by the narrow voltage swing, which means the supported precision is low in practice. Besides, the current-to-voltage nonlinearity of the eNVM during input encoding could introduce errors in MAC operations, resulting in accuracy loss. For binary encoding, multi-bit inputs are sent sequentially to the CIM array and processed in a bit-serial parallel fashion. Additional circuits, such as registers and shift-adders, are required to combine the sequential data. Input DACs could be eliminated in binary encoding by hiring less-expensive digital circuits. Compared to amplitude-based encoding, binary encoding is more viable but offers lower throughput.

For output sensing, as illustrated in Chapter 2, the memory array is usually equipped with ADCs to convert analog MAC values to digital outputs for communicating with other units in the system. Hence, grand challenges are introduced by the mixed-signal compute scheme in CIM, including non-idealities from devices/circuits, hardware overhead by expensive DACs/ADCs, and limited performance while scaling to a large-scale system, as shown in Figure 8.

In this work, we propose an *E*fficient *N*eural *N*etwork *A*ccelerator, namely *ENNA*, based on "ADC-free" CIM subarray design. We develop a chip architecture that employs high-precision ADCs at higher level of hierarchy, which minimizes the data conversion energy and makes ADC no longer the bottleneck of the entire system. We evaluate the system-level hardware performance based on several large DNN models with silicon data measured from the macro chip implemented with TSMC 40nm RRAM process. To explore the potential of the proposed design, we project our design with advanced heterogeneous 3D (H3D) technology to unleash the parallelism of the computation.

4.2 Hardware Implementation

4.2.1 Design of ADC-free RRAM Subarray

Figure 9 presents the circuit diagram of the ADC-free CIM array. One single array involves a 256×256 1T1R RRAM memory array, capacitive digital-to-analog converters (DACs), comparators for input PWM-based encoding, multiplexing for row/column selection, current subtractors based on current-mirror structure, and edge capacitors. The conventional 1T1R design, where wordlines (WLs) are horizontal while both bitlines (BLs) and sourcelines (SLs) are vertical, is employed. Each 1T1R cell stores a binary value. We stored different significant bits in different subarrays to represent multi-bit weight. Many eNVMs, including RRAM, suffer from a limited on/off ratio. As shown in Figure 9, as a countermeasure, one dummy column is programmed to all off-states to neutralize the off-state current. The input is applied to WL with SL grounded. Rather than using amplitude-based or binary encoding, we employ a pulse-width-modulation (PWM)-based DAC to encode multi-bit inputs as varying pulse-widths. Compared to amplitude-based encoding,

PWM-based encoding is much less affected by the current-voltage nonlinearity, making it appealing to achieve more accurate results. Compared to binary encoding, PWM-based encoding could feed in multi-bit input in one cycle, significantly improving the throughput. The charge-based current-to-voltage stacking conversion is implemented to perform analog MAC operation.

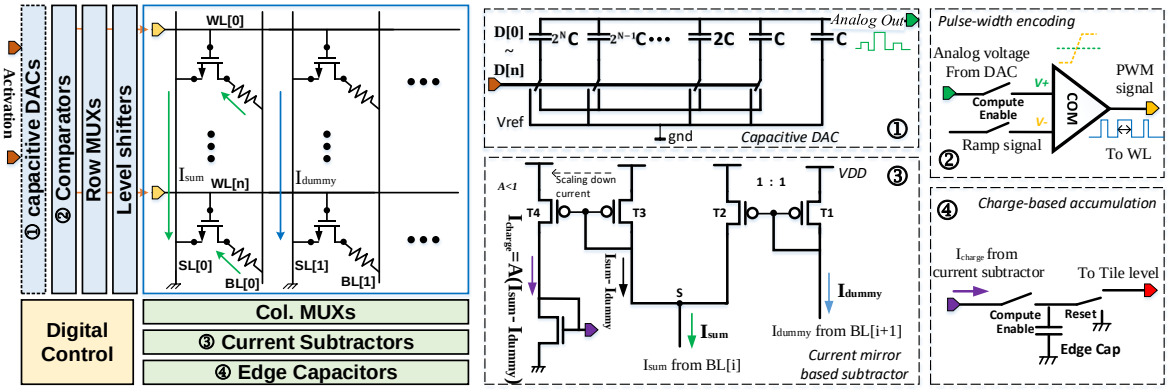


Figure 9: The ADC-free RRAM subarray design: ① The diagram of capacitive DAC; ② PWM input encoding method based on comparator; ③ the current-domain subtraction circuitry; ④ Edge capacitors with control signals

The operational waveform of intra-array analog computing is presented in Figure 10. the multi-bit digital input is first converted to an analog voltage signal by a capacitive DAC (see Figure 9 ①) in the first phase (PH1), and is then transformed to a PWM signal by a comparison operation in the second phase (PH2). The two inputs to the comparator are the analog voltage input and a uniform ramp signal (see Figure 9 ②). As shown in the waveform of PH2, while the ramp signal (*V*-) passes over the analog voltage (*V*+), the comparator produces a PWM output whose pulse duration is proportional to the amplitude of the analog voltage. Simultaneously, PWM input pulses that reach access transistors' gates sink current from BLs to grounded SLs. I_dummy from the dummy column is copied from transistor *T1* to transistor *T2* and subtracted from I_sum of the weight columns,

eliminating the contribution from non-ideal off-state (see Figure 9 ③). Another pair of the current mirror (*T3-T4* pair) scales down the subtracted current. The scale-down current (*I_charge*) is immediately converted to an analog voltage by charging the edge capacitor (see Figure 9 ④) rather than digitizing the partial sum using an ADC-like conventional approach. The charged analog voltage on the edge capacitor represents the MAC outputs. Through the third phase (PH3), the capacitor will temporarily retain the voltage until the computation of the entire column completes. Due to hardware constraints, only limited rows (i.e., 8 rows) are activated simultaneously, but the actual compute of one column (@100 MHz with edge capacitors ~500fF) is executed within several hundreds of nanoseconds. Thus, there is no concern about the voltage decay on the capacitor since the typical retention time is on the order of microseconds, similar to eDRAM. Then PH1 to PH3 will repeat, and the analog MAC outputs are stacked on the edge capacitor until all rows are computed.

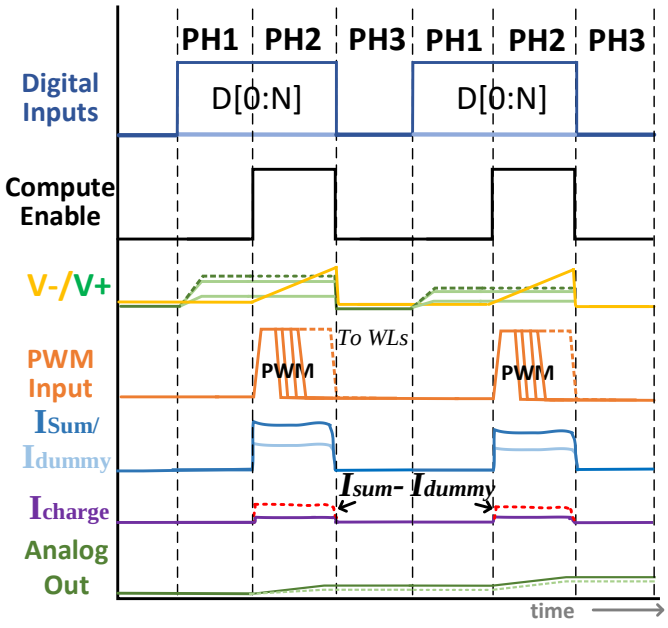


Figure 10: Operational waveform of the proposed array-level ADC-free compute scheme.

4.2.2 Hierarchical Architecture

Figure 11 exhibits the tile-level architecture and the inter-array dataflow. Given the limited sub-array size and the large filter size of one layer in DNNs, which can range from several hundred up to thousands, array partitioning is necessary. One tile is formed by nine subarrays (corresponding to 3×3 filters). The tile-level design also includes input/output buffers, a shared ramp generator, and high-precision. Despite the fact that we eliminate ADC at array-level, we still need to digitize the analog outputs at tile-level, offering scalability for a hierarchical large-scale system design. As the dataflow shows in Figure 11, activations are firstly fed into local buffers for each RRAM subarray through tile-level interconnects. Local buffers could reduce expensive data transfer through interconnect since array-level input vectors could be reused across columns as multiple columns share one output sensing block. In our design, 16 columns share one current subtractor. Secondly, CIM arrays perform MAC operation as Figure 10 illustrates, and then the final array-level outputs are stored on edge capacitors of nine subarrays. Thirdly, for tile-level analog accumulation, all switches (SWs) are turned on, and nine edge capacitors with sampled charge are connected to average the voltage by charge redistribution. Finally, the accumulated MAC results (V_{ac}) are sent to the tile-level SAR-ADC for 7bit digital output, which are then sent to the output buffer for further data conveying and processing. To summarize, local analog communication is conducted in the PWM format within the tile, and ADC quantization occurs only between tiles which is at a higher level of the hierarchy for global digital communication. However, due to limited space on the prototype chip, we only fabricate a two-array macro to demonstrate the proposed PWM-based input encoding design and charge-based ADC-free voltage sensing scheme. In this case, we build the CIM

system with NeuroSim simulator [43] and integrate the array-level performance (measured from the chip) into the hierarchy above the subarray level. The system setup and evaluation results will be presented in next section.

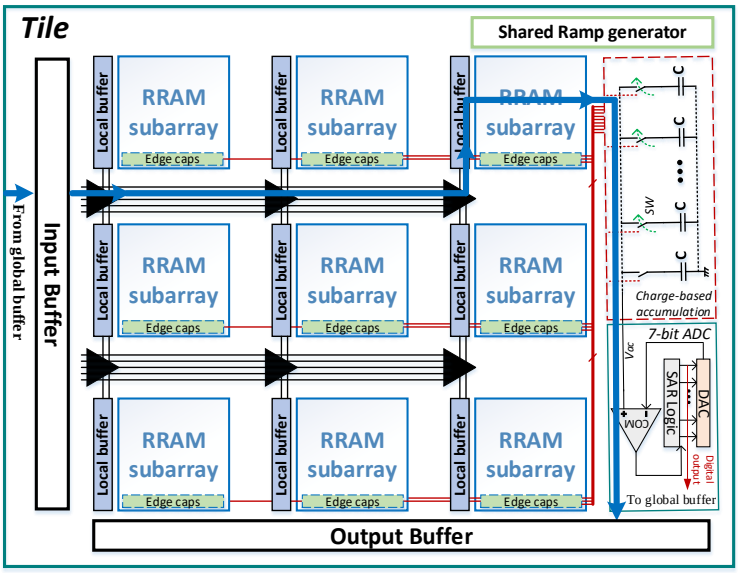


Figure 11: Tile-level architecture and dataflow between subarrays

As discussed in Chapter 3, compared to Flash-ADC, SAR-ADC shows benefits at higher precision (e.g., > 4-bit) due to their linearly increased area/energy cost. Due to the tile-level outputs being acquired by summing up all kernels, which demands more precision than the array-level MAC results, we choose SAR-ADC for our proposed design. More crucially, different ADC precision will directly affect the accuracy performance and the hardware cost in the CIM-based design. As shown in Figure 12, we demonstrate the impact of ADC quantization loss on accuracy performance to verify the required tile-level ADC precision. Ideally, a high-resolution ADC can be employed to avoid any quantization loss. However, ADC's area/power or even latency cost will increase radically as ADC precision increases. Our ADC-free CIM array fetches 4-bit inputs and sums up 256 rows in total,

which means the ideal full precision of each column MAC is 12-bit. Then this analog partial sum for subarray needs to be accumulated across nine subarrays, which means the final theoretical full precision before tile-level ADC is 16-bit. By incorporating the proposed compute scheme and real-traced input/weight data into software simulation, we explore the impact of ADC resolution from 4-bit to 9-bit on different networks. The plots in Figure 12 demonstrate that the 7-bit ADC could maintain the same accuracy as the baseline, which means the 9-bit quantization loss is bearable. Compared to prior CIM designs with array-level ADCs tolerating limited quantization loss, our proposed scheme could tolerate more quantization loss because quantization loss only happens at the tile-level stage. On the other hand, quantization loss happens on each column's partial sum from array-level ADCs. Then, as these quantized partial sums are accumulated at the upper level, errors will also be accumulated, causing a bigger loss in accuracy.

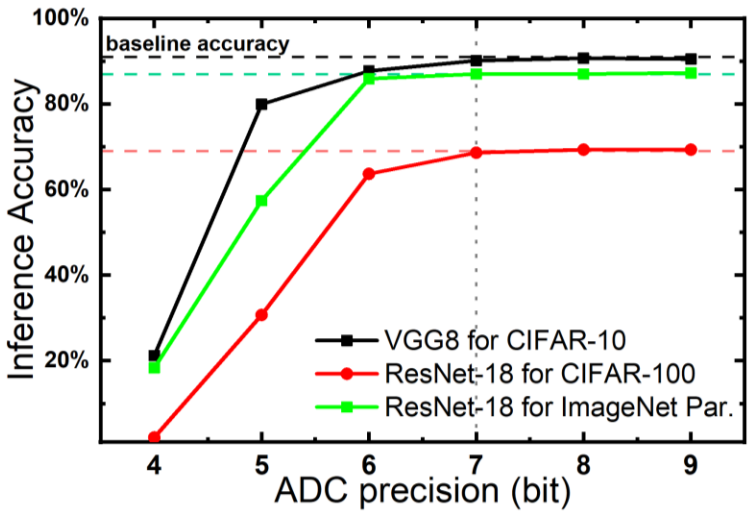


Figure 12: Accuracy performance versus tile-level ADC precision.

Figure 13 (a) presents the key components of our ENNA accelerator at the chip level: multiple tiles assigned for different layers, digital computational units, neural functional

units such as pooling/activation, and global buffer. H-tree routing is employed to estimate the overhead of interconnection. The H-tree is built up by multiple stages from wide to narrow ones, corresponding to the main bus to subarray/tile connection. The actual size of the filters determines the number of tiles assigned for each layer. The sub-array size determines the local buffer size, while the global buffer size depends on the model's input feature map for pipelining. Accordingly, we allocate 1.8Mb, 2.4Mb, and 24Mb global buffers for VGG-8, CIFAR-100, and ImageNet, respectively.

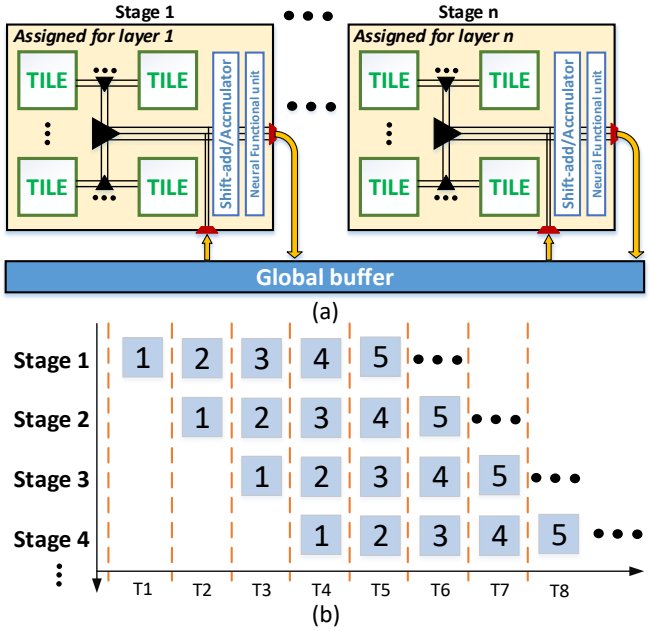


Figure 13: (a) The top-level ENNA architecture. (b) Dataflow diagram of layer-wise pipeline

The multi-bit weights are deposited in different tiles across the layer, and shift-adders are used to accumulate the tile outputs. The accumulated results are then further processed by neural functional units to generate the activations for the next layer, which are sent to the global buffer for storage. Figure 13 (b) illustrates the layer-wise pipeline according to the proposed top-level architecture. Each pipeline stage is assigned for one single layer,

and forward propagation is executed stage-by-stage. Each stage could perform different tasks (i.e., images) in parallel to improve the system throughput. The speed of the slowest stage, which is usually the 1st or 2nd layer in neural networks, determines the pipelining efficiency.

4.2.3 Heterogeneous 3D Integration

Except for the hardware overhead caused by ADCs, there are still other remaining limitations, such as CMOS technology scaling challenges and chip area constraints, to be faced. In this case, heterogeneous 3D (H3D) integration enabled by through silicon via (TSV) technology and hybrid bonding become a competitive solution. Some advanced TSV technologies, such as nano-TSV, which could reach sub-micron pitches [44], have been proposed to reduce the die-to-die interconnection overhead. For CIM design, by partitioning the circuit modules in hybrid technology nodes across different tiers (e.g., the memory tiers at legacy node and logic tiers at leading edge node), the challenges of ADC/DAC overhead and scaling limitations caused by high write voltage in eNVM could be addressed [45]. 3D CIM accelerator could achieve higher throughput and larger capacity at the array level to unleash the system's performance.

We redesign the ENNA architecture as a 2-tier design to investigate the potential of CIM design with 3D stacking technology. To embrace logic scaling, the digital blocks, buffer, and low-voltage peripherals are relocated to the bottom tier at 7 nm, while the RRAM array and high-voltage peripherals remain on the top tier at 40 nm as TSMC is currently offering. Figure 14 shows the schematic of the proposed 2-tier partition. It should be noted that the level shifters and switch matrixes have to stay on the top tier as RRAM

cells require high write voltage (>3V) for programming, which is not applicable at 7 nm. The nano-TSV technology pioneered by IMEC [44], which could achieve sub-500 nm pitch interconnects, is chosen in our 3D design due to its trivial area cost. Additionally, with such a small size, TSVs' parasitic capacitance and resistance become insignificant. Thanks to the decreased size of ADCs/DACs at 7 nm, this 3D-ENNA architecture can support in-memory computing in a highly parallel read-out manner: 128 rows are activated simultaneously with PWM inputs, and the ADC-free analog sensing simultaneously reads out 128 columns of MAC outputs. To match the array-level bandwidth, the same number of SAR-ADCs and corresponding digital compute blocks are employed. In this case, the area-hungry CIM periphery is no longer the bottleneck of parallelism. The next section will present the evaluation of the 3D-ENNA architecture.

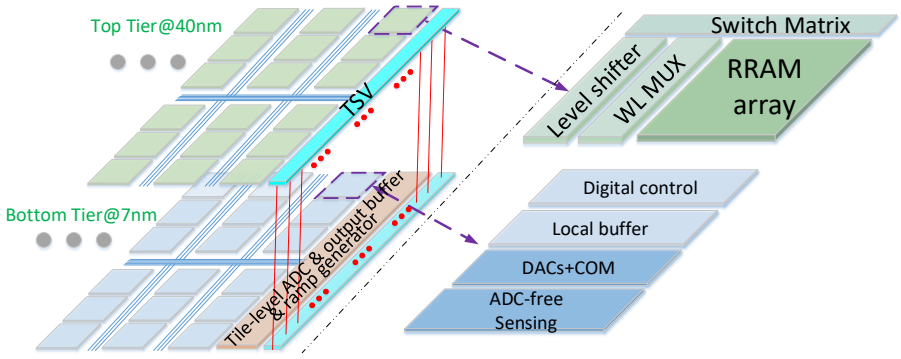


Figure 14: Floorplan of H3D 2-tier ENNA accelerator

4.3 Evaluation

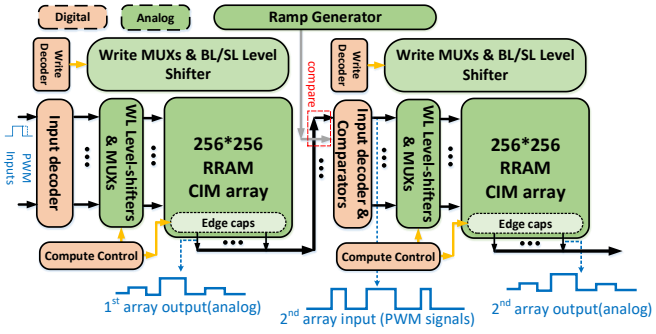
4.3.1 ADC-free RRAM Prototype Chip

RRAM is a two-terminal device with a metal-oxide-metal structure that could be integrated at the drain via a silicon transistor. The data is stored as a high resistance state

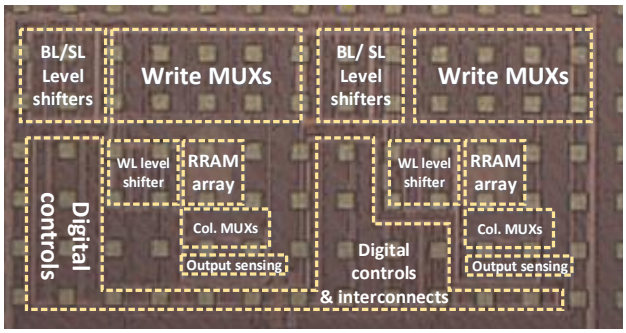
(off-state) or low resistance state (on-state) for binary operations, and multilevel operations are also possible by varying the resistance. Among eNVM-based CIM, RRAM is a promising candidate due to its silicon CMOS fabrication compatibility and low integration cost [46]. Compared to PCM, RRAM has lower write energy. The industry has invested in RRAM technology in the past decade, and 16Gb [47] to 32Gb [48] RRAM prototype chips have been demonstrated. Recently, TSMC has offered the embedded RRAM process in the 40 nm process [49], and Intel offered the embedded RRAM process in 22nm FinFET platform [50].

A 128 kb ADC-free CIM macro is prototyped in TSMC RRAM 40nm process. Figure 15 (a) shows the overall chip architecture, highlighting the intra-/inter-array compute dataflow. This macro includes two identical RRAM CIM arrays, digital control for CIM computing, and essential peripheries for device programming (e.g., write MUXs and level-shifters). During in-memory computing, the input vector is encoded as a group of word line pulses to the rows of the memory array, where the pulse-width represents the analog input value. As opposed to digitizing the partial sum using an ADC-like conventional approach, the accumulated current is directly transformed to an analog voltage, which is temporally stored onto the edge capacitor. When the computation of the next array commences, the analog voltage is compared to a ramp signal to generate PWM signals representing the inputs of the second array. Then the second array will perform the MAC operations in the same computing manner as the first array. Figure 15 (b) presents a die photo with labels indicating the sub-blocks. A summary of chip measurement results is shown in Figure 15 (c). We measure analog MAC operating at 100 MHz with 0.9 V supply voltage. Here one operation is regarded as 1-bit weight × analog input. With eight rows

operating simultaneously, this ADC-free RRAM CIM macro could achieve 13.93 GOPS throughput and 26.97 TOPS/W energy efficiency.



(a)



(b)

Technology	TSMC 40nm CMOS w/ RRAM
Array size	256*256
Macro size (mm^2)	0.619
Rows turned on simultaneously	8
IN/W/OUT precision	Analog/1b/Analog
Clock frequency (MHz)	100
Supply voltage (V)	0.9
Energy efficiency (TOPS/W)	26.97
Throughput (GOPS)	13.93

note: 1 OPS = 1b W × analog IN

(c)

Figure 15: (a) Two-array macro architecture of prototype chip. (b) Die photo of ADC-free RRAM macro. (c) Chip summary of silicon measurement data

Figure 16 (a) presents the demonstration setup, which combines the proposed ADC-free RRAM-CIM test chip with an NI-PXIe system to deliver control signals/inputs and measure the outputs. Correct CIM operations are measured as shown in Figure 16 (b). The

activation inputs and code for setting the hardware configuration are first scanned into the chip. Following receipt of the start signal, the CIM array performs in-memory computing. Varying PWM outputs with different pulse-width correspond to different MAC results. From left to right, the waveform shows that the pulse-width gets narrow, corresponding to decreasing MAC output.

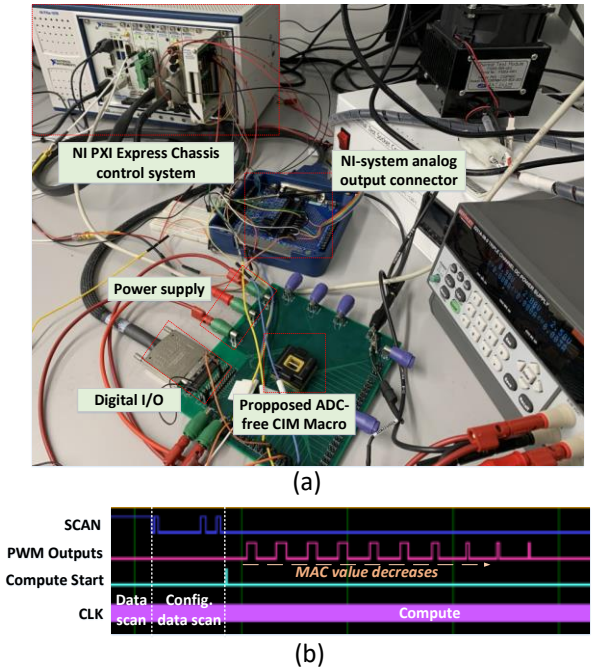
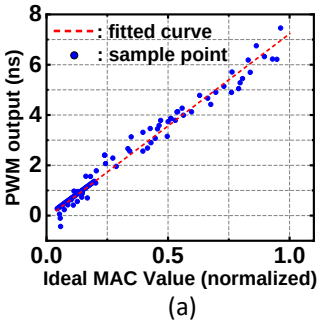


Figure 16: (a) The experimental platform based on NI-PXIe Chassis. (b) Measurement results on demonstrated transient response.

4.3.2 Accuracy Performance

Undeniably, CIM analog computing suffers more from process variations/noises than traditional digital computing. Furthermore, even when the off-state current is cancelled out for more accurate MAC outputs, other non-idealities of RRAM devices might still affect the accuracy. The impact of process variations or noises on accuracy performance needs to be verified. We first extracted weights and input vectors from software simulations that

match the proposed in-memory compute scheme (i.e., open eight rows simultaneously with pulse-modulated inputs). Then the acquired weights are programmed to RRAM arrays, and the proposed PWM-based method encodes the corresponding input vectors.



	σ in terms of percentage
Simulated temporal variation	2.9%
Measured temporal variation	3.9%
Measured temporal & spatial variation	4.1%

(b)

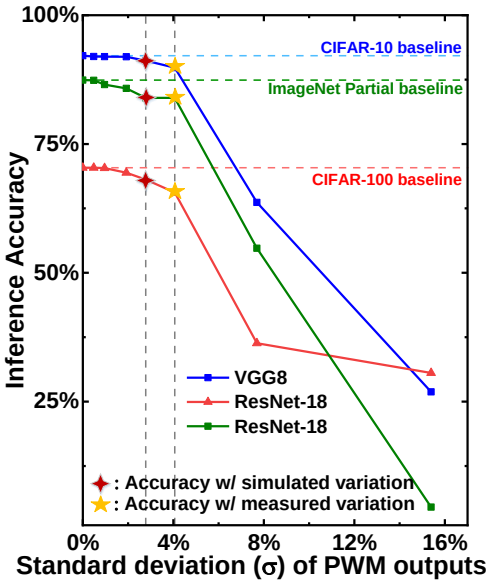


Figure 17: (a) Real MAC outputs versus ideal MAC values. (b) Results of simulated/measured variations. Standard deviation (σ) is measured in terms of percentage. (c) Accuracy performance with process variations tested on various DNN models.

Figure 17 (a) illustrates real-traced MAC outputs versus ideal MAC values based on 200 samples of different values and multiple reads, showing a linear-like relationship. As shown in Figure 17 (b), the simulated standard deviation (σ) of the temporal variation (related to various input & weight patterns) on PWM outputs is 2.9%, while the measured value is 3.8% in terms of percentage. After including spatial variation (read-out across different columns), the measured total standard deviation is 4.1%. Here we take 4-bit input & 2-bit weight configuration as a case study. By integrating the measured output variation in the software simulations, we evaluate the inference accuracy on the VGG-8 and ResNet-

18 networks trained for the CIFAR-10, CIFAR-100, and ImageNet-partial datasets. As shown in Figure 17 (c), our ADC-free CIM design achieves 91% accuracy for CIFAR-10 classification (software baseline: 92%), 68% accuracy for CIFAR-100 classification (software baseline: 70.4%), and 84% accuracy for ImageNet-partial classification (software baseline: 87.4%).

Table 4: Performance comparison across various weight configurations

	VGG-8 for CIFAR-10			
Input precision	4	4	4	4
Weight precision	2	4	6	8
Throughput (GOPS)	282.3	277.1	274.5	273.8
Energy efficiency (TOPS/W)	10.8	3.7	2.3	1.6
Accuracy (%)	89.2	91.3	91.5	91.6

4.3.3 Benchmark Results

The system-level hardware and accuracy performance across eight input-weight configurations for inference are summarized in Table 4. On the VGG-8 model for CIFAR-10 classifications, the 2D ENNA accelerator achieves 1.6~10.8 TOPS/W and 273.8~282.3 GOPS while maintaining 89~91% accuracy. As observed in Table 4, increasing weight precision could slightly improve accuracy but degrade the hardware performance.

1) Comparison among 2D-/3D-ENNA and baseline design

To have an apple-to-apple comparison, we build a baseline accelerator with explicit ADCs based on a fabricated CIM prototype chip that was fabricated with the same RRAM process [51]. The hierarchy architecture of the baseline accelerator is shown in Figure 18.

3-bit ADCs are hired next to the memory array for MAC output quantization at the sub-array level, which enables traditional in-memory computing. The macro-level parameters for baseline are obtained from the experiment results as well. The baseline CIM accelerators have three primary hierarchies: chip level, tile level, and sub-array level. Peripheries such as buffers, neural function units, and digital computational blocks (e.g., adder trees and shift-adds) are included inside different levels.

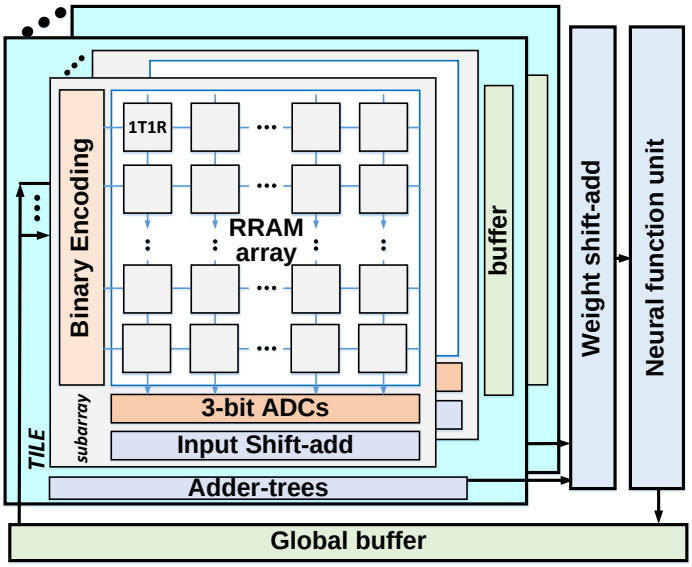


Figure 18: Top-level architecture of baseline design (with subarray level ADCs) for one convolution layer

Figure 19 shows the system-level performance comparison between the proposed ENNA accelerator (2D + 3D) and the baseline for various networks. As illustrated in Figure 19 (a), since expensive ADCs are moved from the sub-array level to the upper tile level with significantly fewer workloads, 2D-ENNA could reduce chip area by 30%~40% (observable across various networks). 3D integration could further reduce the chip area by ~50% compared to the 2D ENNA accelerator by re-organizing the design into two tiers. Compared to the binary encoding method in baseline design, the proposed multi-bit PWM

encoding improves the throughput by 2×~2.8×, and pipelined execution further improves the throughput by 2.3×~8.9× for different tasks. In total, as shown in Figure 19 (b), our proposed 2D-ENNA accelerator could perform computations 4.7×~25.7× faster than conventional CIM designs, mainly benefitted from the multi-bit PWM-based input encoding scheme and pipelined datapath. ResNet-18 has a higher speed-up as pipelining shows more advantages on deeper networks.

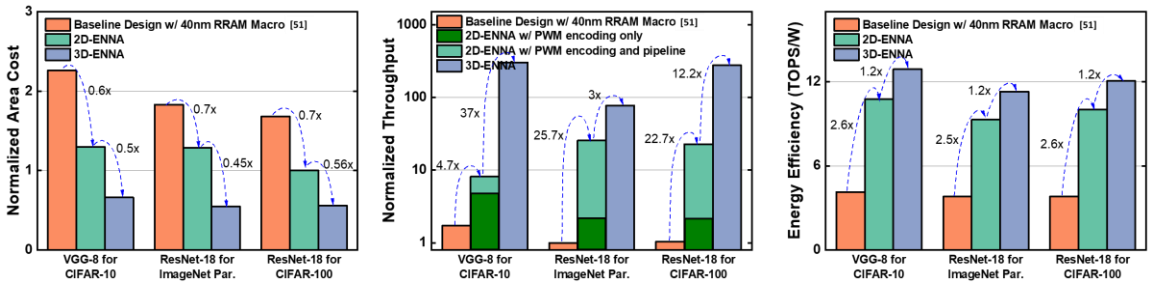


Figure 19: Performance benchmark on different DNN models (2-bit weight/4-bit input configuration): (a) Area cost comparison. (b) Throughput comparison. (c) Energy efficiency comparison.

Considering the area cost, 2D-ENNA only opens 8 rows simultaneously. If 128 rows are activated at the same time, the chip area will increase by 5×, which will be over 1000 mm^2. On the contrary, by applying 3D integration and advanced technode on the bottom tier, such a large area overhead for more opening rows could be hidden. Thus, by fully using CIM parallelism (i.e., activating more rows/columns in H3D architecture), 3D-ENNA could further improve the throughput by 3×~37× with even lower area cost. Compared to VGG-8 with the 2nd-layer's channel size of 128×128 (determining pipelining speed), ResNet-18 for CIFAR-100 shows less improvement on throughput as 3D's high-parallelism benefit (opening 128 rows/columns simultaneously) is limited by the smaller channel size (i.e., 64×64) of ResNet's first several layers (determining pipelining speed).

Besides, since the input image size (i.e., 224×224) from ImageNet dataset is much larger than that (i.e., 32×32) from CIFAR dataset and speed-up from 3D stacking is limited on layer 1, processing the 1st layer in ResNet-18 for ImageNet takes the most portion of total latency. As a result, throughput improvement by 3D stacking on ImageNet classification is less. To summarize, the throughput benefit of 3D stacking on CIM system varies according to network structures and the complexity of the tasks. The results of energy efficiency are compared in Figure 19 (c). Our 2D-ENNA accelerator achieves 9.2~10.8 TOPS/W demonstrated on different networks, on average ~2.5× of the baseline design. There is a slight improvement in energy efficiency (~1.2×) from 3D integration mainly because the energy consumption of the in-memory computing from the top-tier stays at legacy 40 nm node, thus not benefiting from the technology scaling.

Figure 20 shows the breakdown reports for the inference task of the proposed 2D-ENNA design. Figure 20 (a) displays the energy breakdown of CIM ADC-free subarrays, tile-level cost (including ADCs, digital units, buffer, and interconnect), and chip-level components (including global buffer, interconnect, and digital computational blocks). We can see that CIM subarrays contribute most of the total energy consumption at the system level, demonstrating the scalability of CIM-based systems. Tile-level peripheries contribute a small portion of total energy cost, and tile-level ADCs are even negligible, proving the efficacy of our ADC-free subarray design. The latency breakdown of key components is shown in Figure 20 (b), indicating that data transfer above the tile level through interconnect and the global buffer is not time-consuming. In 2D ENNA architecture, the time consumption of in-memory computing stands out, diminishing the system throughput.

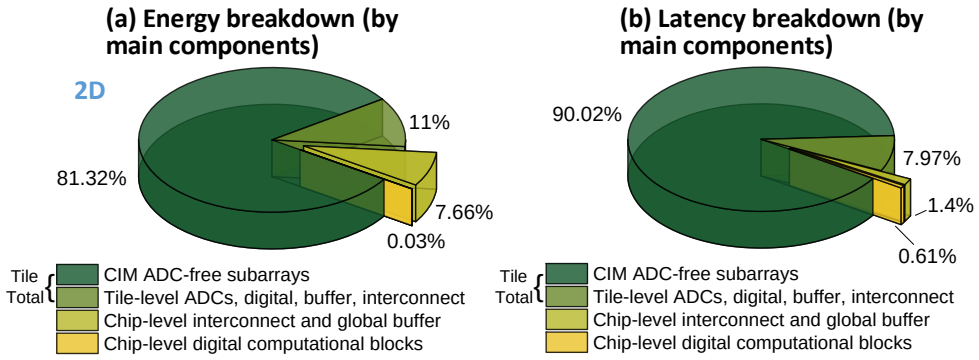


Figure 20: (a) Energy and (b) Latency breakdown of 2D-ENNA design by main components (tested on VGG-8 for CIFAR-10).

Figure 21 shows the energy and latency breakdowns of the proposed design with 3D integration. While comparing the energy breakdown results between Figure 20 (a) and Figure 21 (a), we find that CIM subarrays take more portion (from 80% to 90%) in 3D design since the energy cost from other parts is reduced by technology scaling. The majority of the total energy cost in both 2D and 3D architectures is contributed by CIM subarrays. This observation also offers support for the limited improvement in energy efficiency by 3D integration. While comparing the latency breakdown shown in Figure 20 (b) and Figure 21 (b), we could observe that data transfer inside the tile through interconnect, and local buffer becomes dominant in 3D architecture. The reason is that interconnects will get more complex, leading to a larger overhead accompanied by wider bandwidth of CIM subarrays in 3D architecture.

In summary, thanks to the PWM input encoding and reduced ADC workloads, our ENNA design could greatly outperform the traditional CIM approach in terms of hardware performance. In addition, the 3D integration of CIM design has the potential to release parallel processing expressively.

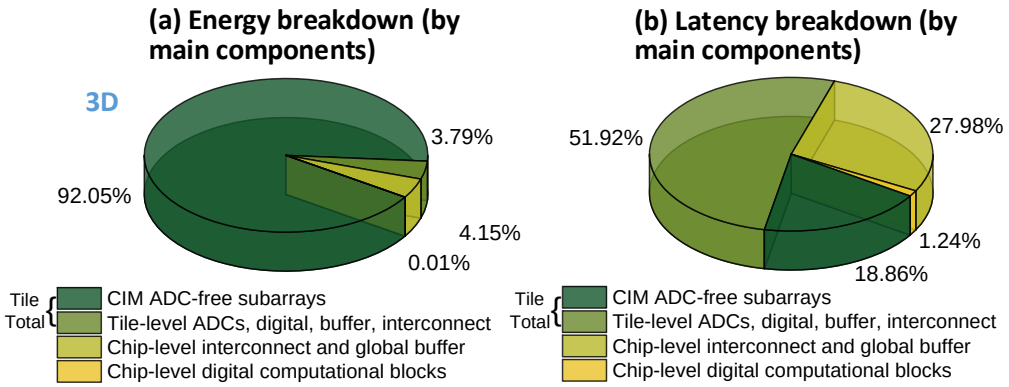


Figure 21: (a) Energy and (b) Latency breakdown of 3D-ENNA design by main components (tested on VGG-8 for CIFAR-10).

Table 5: Comparison table with state-of-the-art CIM macro designs using SRAM and RRAM technologies

	ISSCC'18 [26]	ISSCC'19 [52]	JSSC'22 [53]	ISSCC'21 [28]	ISSCC'21 [30]	CICC'21 [51]	This work
Technology	65nm	28nm	14nm	22nm	40nm	40nm	40nm
Cell Type	RRAM	8T-SRAM	PCM	RRAM	RRAM	RRAM	RRAM
Power supply	1	0.6	0.8	0.8	0.9	0.9	0.9
Input/Weight precision	1b/2b	8b/1b	8b/analog	8b/8b	8b/8b	1b/8b	Analog/2b
Sensing scheme	3b ADC	PWM Conversion	CCO-based ADC	2-phase 4b-ADC	4b ADC	3b ADC	PWM Conversion
Normalized throughput (GOPS)	-	<1200*	8000 (peak)*	835.92*	-	20.96*	222.88*
Reported energy efficiency (TOPS/W)	19.1	46.7	10.5	11.91	56.67	36.39	26.97
Normalized energy efficiency (TOPS/W)	38.4*	373.6*	84*	391.4*	56.67*	36.39*	431.52*
Normalized compute Efficiency (GOPS/mm2)	-	-	12720*	139.32*	-	83.52*	360*

*: normalized performance for 1b IN×1b/analog W ops

2) Comparison with Prior Works

To have a fair comparison, we benchmark our design with macro-only CIM works and system-level accelerator design, respectively. Table 5 compares the proposed ADC-free RRAM-CIM with state-of-the-art CIM macro designs at similar tech nodes [52] [53]. Here the definition of one operation is normalized to be 1-bit by 1-bit MAC to evaluate

different designs. Our ADC-free macro achieves the highest normalized energy efficiency (~431.52 TOPS/W) and compute efficiency (360 GOPS/mm^2) compared with other RRAM macro-only designs.

Table 6: Comparison table with recent CIM accelerators at system-level

	DAC'19 [54]	ISSCC'21 [55]	ISSCC'21 [56]	This work	
Technology	45nm	65nm	16nm	40nm (2D-ENNA)	40nm+7nm (3D-ENNA)
Device	RRAM	SRAM	SRAM	RRAM	RRAM
Array size	256×256	128×128	1152×256	256×256	256×256
Array-level interfacing	2-bit DAC/ 6b SAR-ADC	Binary/ 4-b ADC	Binary/ 8b ADC	4-bit DAC /ADC-free	4-bit DAC /ADC-free
Bit Precision	2b IN/8b W	2-8b IN/1-8b W	1-8b IN/W	1-8b IN/ 2-8b W	1-8b IN/ 2-8b W
Application	CNN	CNN	CNN	CNN	CNN
Supported networks	CIFAR-10	CIFAR-10 ImageNet	CIFAR-10 ImageNet	CIFAR-10/-100 ImageNet	CIFAR-10/-100 ImageNet
Supply Voltage	/	1V/0.62V	0.8	0.9	0.9
System Throughput (TOPS)	/	0.1-3.16	3	0.282-0.880	2.6-13.4
System Energy Efficiency (TOPS/W)	3.44	2.75	121	9.2-10.8	11.3-12.9
Normalized TOPS/W*	55.04	75.9	121	73.6-86.4	90.4-103.2

*: Scale with number of bits used for input activation and weight

Table 6 summarizes and compares the ENNA accelerator with state-of-the-art CIM accelerator designs at scale [54] [55] [56]. The proposed ENNA accelerator supports operations with programmable 1-8 bit inputs/weights. Thanks to the improved hierarchy architecture and ADC-free data computation between sub-arrays, this ENNA accelerator achieves improvements in normalized energy efficiency by 1.56× and 1.14× compared to an RRAM CIM design [54] and an SRAM CIM design [53] at comparable technology node, respectively. The more advanced 16 nm SRAM CIM design [55] shows benefits introduced by the technology scaling. If we normalize the performance to the same

technology node, our proposed design could potentially show more energy efficiency and throughput benefits. In addition, 3D integration dramatically improves ENNA's system throughput from 0.3~0.9 TOPS to 2.6~13.4 TOPS, signifying the promises of CIM design with 3D technologies.

4.4 Summary

In this work, we proposed a novel CIM subarray without explicit ADCs. A high-throughput input encoding scheme based on pulse-width modulation (PWM) is proposed to perform MAC operation for multi-bit input in one cycle. The MAC outputs could be temporally stored on edge capacitors through the charge-based compute, improving the area and energy efficiency over the prior CIM subarray design that equips with expensive ADCs. We develop a chip architecture that employs explicit ADCs at a higher level of the hierarchy, which minimizes the data conversion energy and makes ADC no longer the bottleneck of the entire system. The impact of tile-level ADC resolution on accuracy performance is explored. At the chip level, a lay-wise pipeline dataflow is applied to speed up the computation. To explore the potential of the proposed design, we project our design with advanced heterogeneous 3D technology to unleash the parallelism of the computation. The proposed ADC-free RRAM array macro is fabricated in 40 nm TSMC process. The impact of temporal/spatial variations is comprehensively evaluated to ensure accuracy performance. According to experimental results, our accelerator could achieve 9.2~10.8 TOPS/W and 282~880 GOPS for inference tasks with 4-bit input and 2-bit weight. Compared to 2D architecture, 3D stacking, which halves the area cost, could further improve the system throughput by 3×~37×.

CHAPTER 5 HARDWARE SUPPORT FOR CIM TRAINING

5.1 Introduction

In the most common application scenario, edge devices are assumed to support the inference of DNN only with well-trained networks from the cloud. While this scenario is friendly to edge devices, considering the limited hardware resources and power budget, some limitations are introduced to its applications. On one side, the networks on the cloud are usually pre-trained with global datasets. Downloading it to edge devices for local use may suffer from performance degradation due to poor adaptation. On the other side, the pre-trained model could not deal with the new scenes captured in the field that are not included in the global datasets. In the inference-only-on-edge scenario, the naïve solution to overcome these limitations is to send the local data back to the cloud for model adaption. However, data communications (e.g., over the wireless network) will introduce additional power consumption and response time. Besides, uploading personal data will cause privacy and security concerns both in transmission and on the cloud. Therefore, it is not only desired but also essential to exploit on-device learning (or on-chip training). Most prior works are focused on inference for the in-memory architecture, leaving the dataflow and mapping strategy for training largely unexplored. Moreover, although low-precision MAC is necessary for CIM, the scalability of most low-precision techniques training is bad. Only moderate-sized networks could maintain similar training accuracy with the quantized parameters, while more advanced networks like ResNet [14] and DenseNet [50] suffer significant performance degradation. In this research, we propose a transpose SRAM-based *CIM A*rchitecture for multi-bit precision DNN *T*raining, namely CIMAT, with essential

circuit implementations and explore the corresponding weight mapping strategies, data flow, and pipeline design.

The rest of this chapter is organized as follows: Section 5.2 introduces the principles of DNN training process. Section 5.3 presents the transpose weight mapping strategy. Section 5.4 presents the circuit implementation for on-chip training. Section 5.5 presents the evaluation results. Section 5.6 summarizes the chapter.

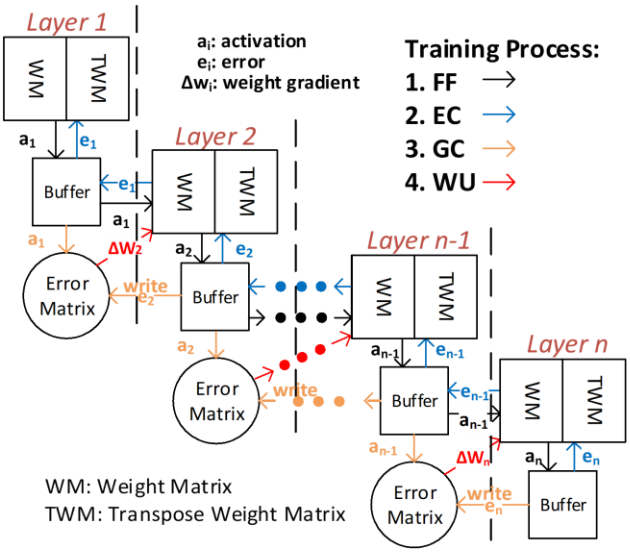


Figure 22: Generic diagram of DNN training.

5.2 DNN Training Basics

DNNs are networks of interconnected nodes (neurons), inspired by the principle of the human brain. A neuron in one layer is connected to neurons in another layer. The strength of these connections between neurons is generally weighed by learnable parameters called weights in DNNs. The process of adjusting weights to achieve a certain purpose from data is called training of the network. There are a lot of different training

methods proposed by the software community. This work focuses on stochastic gradient descent (SGD), one of the most widely used training methods for DNN today.

A simplified workflow of the SGD training process is shown in Figure 22. We divide the process into four steps when it is mapped to CIM architecture: 1) feed-forward (FF), 2) error calculation (EC), 3) gradient (ΔW) calculation (GC) and 4) weight update (WU). As the SGD is batch based, each input in the batch will go through steps 1) to 3) to get the corresponding gradient. Then, step 4) occurs at the end of the batch to update the weight with accumulated gradient. This batch process will be repeated for iterations/epochs with resampled inputs to obtain a well-trained model.

The FF process is exactly the same as the inference. Input data is fed into the network and processed layer by layer from the 1st to the last layer. In detail, the CONV/FC layer will take the input or activations from the previous layer (or layers) and perform VMM on the weight matrix. Some other functions (e.g., ReLU, BN) may process the outputs of VMM before they are fed into the next layer. For inference, these intermediate activations could be discarded after calculation. However, in training, they must be stored for later usage in step 3). Eq. 1 shows the FF operation for a given layer *i*.

$$a_i = f(w_i \cdot a_{i-1} + b_i) \qquad (1)$$

In this equation, a_{i-1} denotes the activations from the previous layer. It is (matrix) multiplied with w_i, which is the weight of the current layer and then shifted by a bias b_n. $f(\cdot)$ denotes the combination of all the other functions, including activation functions, pooling, BN and so on.

After the FF process is done, the gradient of inputs will be calculated with respect to the loss ($\delta L/\delta a_i$). Here, the loss is a metric used to measure the difference between the target and the network's output, which should be minimized for good performance. The input gradient is also called the error (e_i). Thus, this process is called EC in this work. Based on the chain rule, the e_i of each layer is calculated from the last layer to the 1st. In other words, e_i of a given layer i is calculated from the the error e_{i+1} and weight matrix w_{i+1} from the layer after it. As shown in Eq. 2, the main operation of EC is the VMM between the error e_{i+1} and the "transposed" weight matrix w_{i+1}^T. As a result, the same weight matrix will be used for both FF and EC but are transposed in the VMM.

$$e_i = \frac{\delta L}{\delta a_i} = \frac{\delta L}{\delta a_{i+1}} \cdot \frac{\delta a_{i+1}}{\delta a_i} = e_{i+1} \cdot w_{i+1}^T \qquad (2)$$

For each layer i, once the error e_i is obtained, its weight gradient g_i could be obtained through the VMM between e_i and the activation a_{i-1} (Eq. 3). Finally, the weights of the network are updated by this value scaled with the learning rate, as shown in Eq. 4. In the batch mode, the ΔW_i will be the accumulation of Δw_i from different inputs.

$$\Delta w_i = g_i = \frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial a_i} \cdot \frac{\partial a_i}{\partial w_i} = e_i \cdot a_{i-1} \qquad (3)$$

$$W_i^t = W_i^{t-1} - \eta \times \Delta W_i \qquad (4)$$

5.3 Transpose Mapping Strategy

Figure 23 (a) shows the mapping of the FF process of the CONV layer to the CIM. Assuming the weight size of the CONV layer is $K_1 \times K_2 \times C \times M$, it could be viewed as M filters of size $K_1 \times K_2 \times C$. Each filter will generated a channel of output feature map of

size E × F, which consists of an output feature map (OFM) of depth equal to M. C is decided by the depth of the input feature map (IFM), whose plane height/width is denoted by H/W. The mapping of the CONV layer to the CIM array is a process to unroll the 4D weight matrix to 2D and partition it to the proper size to store it in the memory array. In this work, the $K_1 \times K_2 \times C \times M$ kernel is first divded to $K_1 \times K_2$ of 2D matrix with size $C \times M$. In this case, a vector of size $1 \times C$ from the IFM could be applied to the row in parallel, and the accumulated result from different columns will consist an OFM vector of size $1 \times M$. This kernel-split mapping weight mapping strategy [51] is proposed to maximize the input data reuse. Each 2D matrix out of the $K_1 \times K_2$ ones is assumed to be processed in a processing unit (PE) consisting of CIM arrays. The partial sums from all the PEs need to be further accumulated by the digital circuits outside PEs. Figure 23 (a) illustrates that only one PE is demonstrated for calculating the first IFM vector (a0 element). If a CONV layer has a filter size of 3×3, 9 PEs must work in parallel to process different input vectors (a0~a2, b0~b2, c0~c2).

Figure 23 (b) shows how the VMM of the transposed weights in the EC process is mapped to transposed CIM operation. As illustrated by a 3×3 filter, the PE corresponding to the first channel group (a0 element) will hold the weight W_{i+1} for forward calculation. For backward calculation, the same PE will be used for error calculation but transposed W_{i+1}^T is used for calcualtion. In other words, for the FF process, a input sliding window of size $3 \times 3 \times C$ will be multiplied with M number of $3 \times 3 \times C$ filters and summed to generate one output independently. Inside each PE, the accumulation has a depth of C, which means all the dot products in the same column of the PE are summed up. For the EC process, the transposed weight can be viewed as C number of $3 \times 3 \times M$ filters with the

re-group process demonstrated in Figure 23 (b). In this case, multiplication and accumulation in side PE are done on the same row. Thus, we process the transposed version of the weight matrix at the PE level in the EC. In summary, the dataflow of EC (Eq. 2) for one PE (a0 element) is shown in Figure 23 (c). With such a weight mapping strategy and a transposable array, the FF and EC can be performed within the same CIM array. In this way, no additional memory access (reloading the weight for transpotion version) or extra hardware (a copy of CIM arrays with transposed weight) is needed in EC, improving the area and energy efficiency.

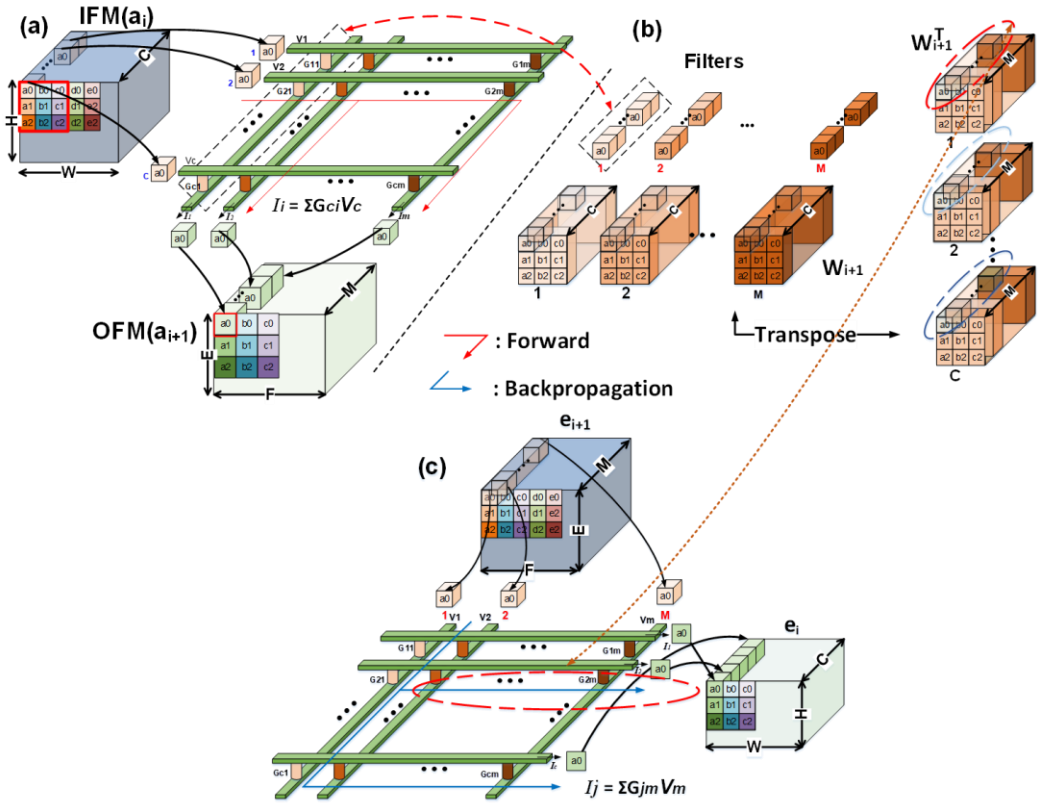


Figure 23: Weight mapping scheme for convolution operations in CIM training.

After FF and EC processes are done, the weight gradient matrix ΔW_i could also be calculated with CIM arrays as its operation is essentially VMM (Eq. (3)). This work

proposes to perform the outer dot product multiplication between error matrix ($\partial L/\partial a_i$) and related input activation matrix (a_{i-1}) using additional 6T non-transpose CIM SRAM arrays.

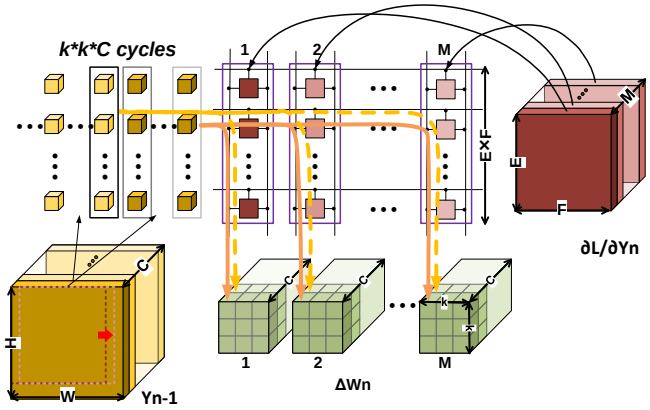


Figure 24: Weight mapping of CIM approach for weight gradient calculation

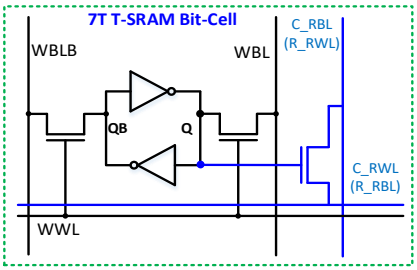
Figure 24 shows the mapping method and dataflow for this GC process implemented in CIM. In detail, the previously calculated error matrix $\partial L/\partial a_i$ from EC is viewed as weight and written in 6T CIM SRAM array. As the error matrix is 3D, it needs to be stretched to 2D for CIM mapping. Each plain of $\partial L/\partial a_i$ is stretched into a $1 \times (E \times F)$ vector and mapped into a long column of the array. Then, the error matrix will become a 2D array with a size $(E \times F) \times M$, which means M columns are needed in total. Then, the activation is loaded (from off-chip DRAM to on-chip buffer) as inputs for VMM calculation. The outer product is done in a sliding window with a window size equal to $E \times F$. In this case, each slding window of $E \times F \times C$ on the input, a_{i-1}, is also stretched into C number of $1 \times (E \times F)$ vectors. These activation vectors, with the same length as the weight matrix height, are fed into the array in different cycles for VMM operations. The number of sliding windows will be $K_1 \times K_2$. One column on each cycle will generate

one point of the weight gradient matrix, and outputs from different columns belong to different filters defined in FF process. After $C \times K_1 \times K_2$ cycles, the entire gradient matrix Δw_i of one layer's filter will be generated. In the batch training mode of SGD, Δw_i of each image need to be stored to off-chip DRAM and loaded back for accumulation at the end of each batch to get the final value for weight update ΔW_i.

5.4 Hardware Implementation for In-memory Training

5.4.1 SRAM Cell Modification

To support the bi-directional CIM operation for FF and EC processes, a 7T transpose SRAM bit-cell is proposed, as shown in Figure 25 (a). The viability of this 7T transpose SRAM array on silicon chips has been validated for another purpose [58]. The area overhead of such 7T design is small while providing disturb-free bi-directional read access. It consists of a regular 6T SRAM cell, which stores the weight data, and an additional transistor (in blue color) for bi-directional read access. This 7T transpose SRAM design has one write mode and two read modes to support the transposable CIM operations. The write mode is the same as the traditional 6T SRAM controlled by the WWL. The two read modes are designed to support the forward and backward process separately, as shown in Figure 25 (b). In forward mode, the neuron input is applied on C_RWL, and the current is summed through C_RBL. These two lines work oppositely for backward mode: input is applied on R_RWL while output is read out from R_RBL. C_RBL/R_RWL wire and C_RWL/R_RBL wire are connected to two sets of WL writers and ADCs separately. Based on the read mode, the current along C_RBL or R_RBL will be proportional to the MAC results and will be converted to the digital signal by the ADCs at the end of the wire.



(a)

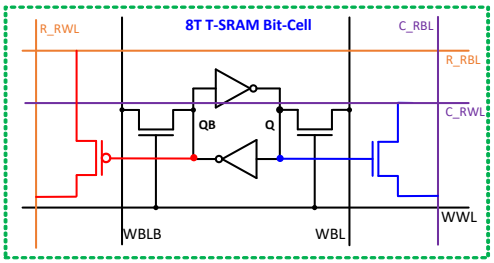
Mode	Column Read Wordline(C_RWL)/ Row Read Bitline(R_RBL)	Column Read Bitline(C_RBL)/ Row Read Wordline(R_RWL)
Forward	Activation input	Readout by Column
Backward	Readout by Row	Error input

(b)

Figure 25: (a) 7T transpose SRAM bit cell. (b) Operation modes of 7T SRAM bit cell

The inference engine usually applies pipelines for acceleration, which can also be applied to the FF and EC processes. Although the EC of a single image could be started right after its FF is done, the FF of other images occupies the array because of the pipeline. While this 7T transpose SRAM design could perform bi-direction read access, both FF and EC calculations are conducted through the additional transistor. Thus, the FF and EC can not be performed simultaneously. As a result, this 7T SRAM-based CIM has to finish the FF of all the images in a batch before entering the EC process. Considering the limited data dependence of FF and EC, the processing efficiency could be improved by executing the FF and EC of a batch in parallel. For this purpose, this work further proposes an 8T T-SRAM bit-cell structure, as shown in Figure 26 (a). Compared with 7T, an additional PMOS transistor is added to the other side of the storage node. Similar to the additional NMOS, this newly added PMOS could also support read access from two sides of bit-cell while its gate is connected to QB. This 8T SRAM design uses different ports to support the two read modes for forward and backward processes (Figure 26 (b)). The additional NMOS

transistor (blue) is activated in the forward process. C_RWL and C_RBL work the same way as the FF process in the 7T bit-cell. Instead, the backward mode utilized the additional PMOS transistor (red) for CIM operation. In other words, the error is applied on the vertical R_RWL with the R_RBL partial sum current read out from the horizontal R_RBL. Thanks to these separate read transistors and WLs/BLs, this 8T cell could perform bi-direction simultaneously. Thus, FF and EC could be executed and pipelined in parallel in this 8T design to speed up the training process.



(a)

Mode	Column Read Wordline(C_RWL)	Column Read Bitline(C_RBL)
Forward	Activation input	Readout by Column

Mode	Row Read Wordline(R_RWL)	Row Read Bitline(R_RBL)
Backward	Error input	Readout by Row

(b)

Figure 26: (a) 8T transpose SRAM bit cell; (b) Operation mode of 8T SRAM bit cell

The storage part of 7T/8T design is exactly the conventional 6T SRAM cell. Normal read/write to the cell could be applied through the WWL and WBL/WBLB. Thus, its design could follow the compact foundry design rule for SRAM without any modification. The additional transistors for bi-directional in-memory computation can use the logic design rule, considering they are not part of a standard memory cell and usually no optimized design rule is provided.

5.4.2 Overall Architecture

The periphery circuits and their connection to the transpose 7T/8T SRAM array are presented in Figure 27 (a). The WL decoder and pre-charge circuit are connected to the 6T storage cell for weight writing. The word line (WL) writers are adopted for both column and row access in the CIM mode. In addition, Flash-ADCs (i.e., multilevel sense amplifiers with different references) are employed on both ends of columns and rows to convert the analog partial sum. After the digitalization, the bit-wise partial sums are accumulated to shift-adders to generate multi-bit input activation with multiple input cycles from the least significant bit (LSB) to the most significant bit (MSB) of input. An extra row of 6T SRAM is added to transpose SRAM CIM array for the weights update. This row shares the BL/BLB with the main array, which could support in memory addition of two rows for weight update as inspired by [58].

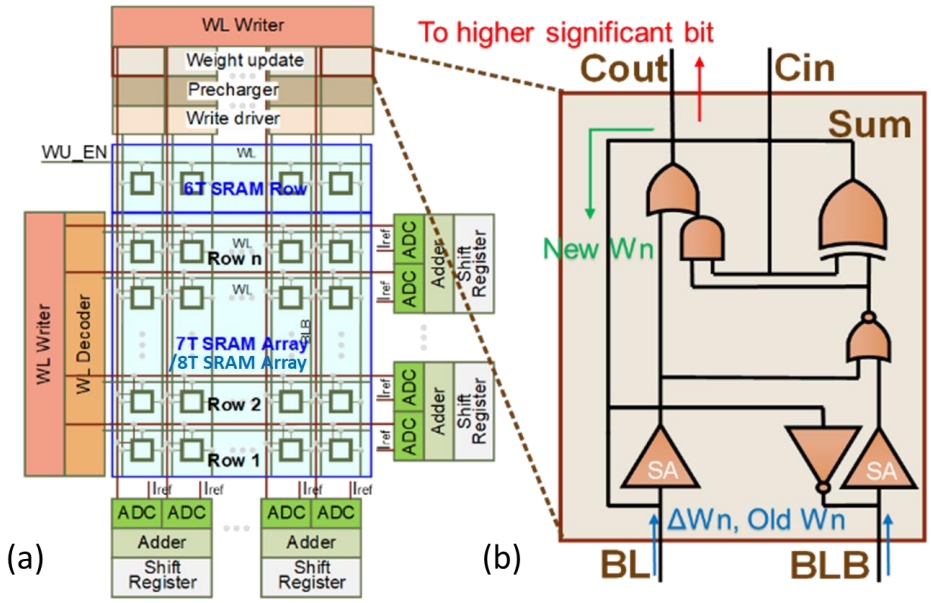


Figure 27: (a) Block diagram of transpose SRAM sub-array and periphery circuity. (b) The structure of weight update modules.

The detailed structure of the weight update module is also shown in Figure 27 (b). During the FF, BP and GC processes, the additional 6T SRAM row is idle. In the WU mode, the weights in the storage nodes of the CIM array need to be updated with the scaled accumulated weight gradients from one batch. By limiting the learning rate (LR) to a power of 2, the multiplication between gradients and LR could be realized by shift. Then, the weight could be updated (Eq. 4) row-by-row in a read-modify-write scheme. First of all, when updating a certain row of weights W_n, the corresponding ΔW_n is written into the additional 6T row. Then, the to-be-updated row and the 6T row are activated simultaneously to generate the AND output of two rows. This output will be further used to calculate the sum of ΔW_n and W_n with the carry in (C_{in}) from lower significance in the weight update module. The sum will be the new W_n that could be written back to the storage row while the carry-out (C_{out}) of the module will be forwarded to the higher significant bit as C_{in}. This row-by-row update in one PE could be accelerated by pipelining. The same significant bits from different weights could be updated in parallel, while the different significant bits of the same weights need to be updated from LSB to MSB considering the C_{in}.

For the on-chip training, all the weight, activation, gradient and error are assumed to be 8-bit fixed-point values in this work according to the recent progress in algorithms. Considering the bi-directional CIM operations, weight bits with the same significance are grouped into a tile. Thus, 8 tiles are needed for 8-bit weight, and the output of each tile will be the CONV result of 1-bit weight and 8-bit input. These outputs will be further combined to full precision-weight and 1-bit input results with digital shift-add modules. We evaluate our CIMAT architecture with training on the ResNet-18 network for

ImageNet dataset classification. The subarray size is assumed to be 128×128 as a practical SRAM array. Thus, some network filters need to be partitioned into subarrays. To ensure the largest CONV layer could fit into a signle tile, 16 subarrays are assumed to form one PE with 9 PEs (corresponding to 3×3 filters) on one tile.

A top-level CIMAT architecture is shown in Figure 28. A tile contains multiple PEs connected with adder trees and an L1 buffer for activations/errors buffering. Every eight tiles are grouped for one CONV layer for 8-bit weights and connected with shift-add units for full precision VMM results. The full-precision digital outputs will go through activation function units for further processing. An L2 buffer is assumed for each tile group for one layer and connected to the global buffer.

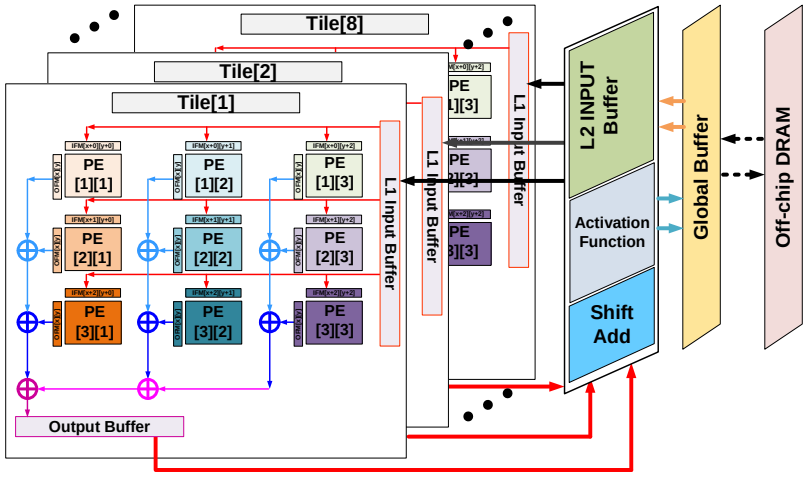


Figure 28: Top-level CIMAT architecture for one convolution layer.

The dataflow on the top level is like this: eight cycles in subarray are needed in both FF and EC processes to accomplish the CIM operation of multi-bit inputs with 1-bit weights inside the PE. The outputs are held and accumulated by the shift-add module at the edge of the subarray. Then the outputs from different subarrays will be added together

through adder trees based on the filter size of the layer. Results from different PEs will be further added together through adder trees to finish MAC for the entire filter with 1-bit weight and 8-bit input. Finally, the shift-add among tiles will generate the eventual full-precision OFM.

5.4.3 Pipeline Design

As mentioned before, this work utilizes pipeline dataflow to accelerate the FF and processes EC of training. As shown in Figure 29 (a), with 7T SRAM bit-cell, the FF and EC are pipelined independently. For the example of ResNet-18, both the FF and EC processes are divided into seven stages. Generally, in the DNN, as the CONV/FC layer goes deeper, the $W \times H$ of the feature map becomes smaller while its channel depth goes bigger. Based on the mapping methods assumed in this work, $W \times H$ input verctors will be applied to the VMM operation in CIM. In this case, $W \times H$ cycles is needed to finish the certain layer if no duplication or folding of the weight is applied. According to the tile architecture assumed in this work, even the biggest filter of the CONV layer could fit into a tile group, which means no folding is applied to any layer. On the contrary, weight is duplicated in the PE for the shallow layers with smaller filter sizes so that one PE is occupied by one layer. As a result, the 1st to 5th layers' latency is similar, equaling two times of the latency of the 6th to 17th convolution. Therefore, each of the first five layers is treated as a pipeline stage. The 6th to 17th layers are grouped as one stage (stage 6). The final stage 7 contains the FC layer and other activation function circuits.

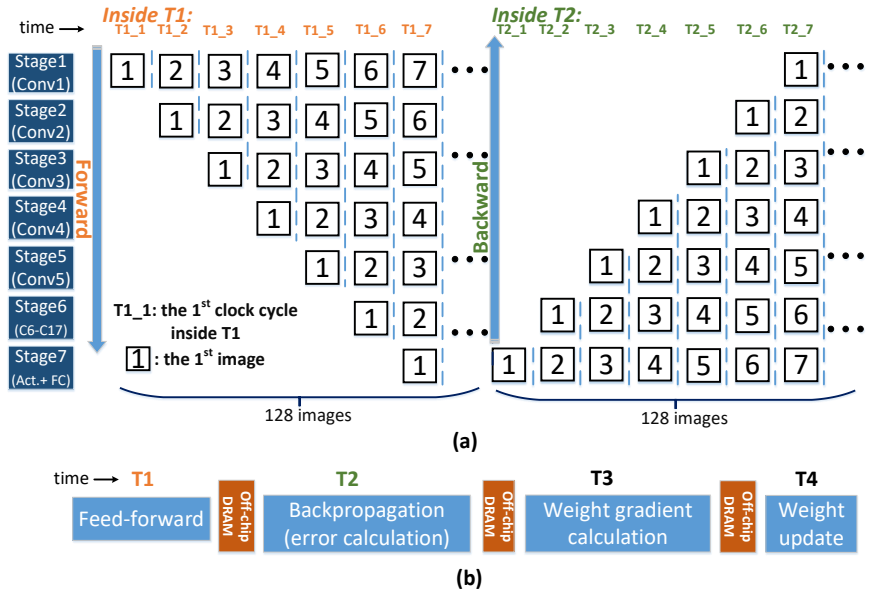


Figure 29: (a) Intra-pipeline design inside FF and EC. (b) The training process in timeline of 7T-based architecture.

The timeline of the 4-stage training process of the 7T design is illustrated in Figure 29. In the first stage (T1), a batch of images is fed into the network for the FF process. After the FF of the whole batch is done, the EC calculation is stated as the second stage (T2). The intermediate feature maps generated in the first two stages must be saved off-chip for later use. After the EC is done for all images from the batch, the feature maps will be loaded back for GC (T3). As each image will generate a group of weight gradients, batch-size groups of gradients will be generated in total. Finally, these groups of weight gradients are accumulated and used to update the weights in the WU stage (T4).

The data flow will be different if the CIM accelerator is implemented with the 8T SRAM bit-cell, as described in section 5.4.1. Since the 8T-based subarray can support parallel bi-directional VMM calculation, the pipeline of FF and EC could be further optimized, as shown in Figure 30 (a). The pipeline stage configurations of the FF and EC

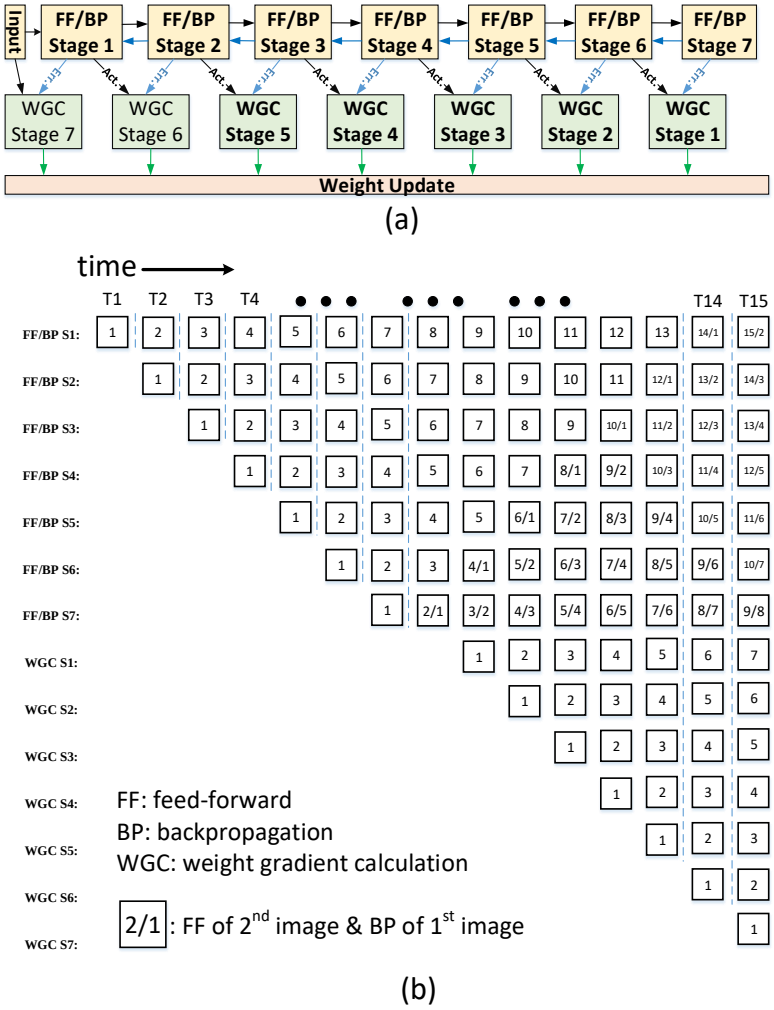


Figure 30: (a) Pipeline structure of 8T SRAM-based CIM training. (b) The state of each stage as a function of time. 15 super clock cycles are illustrated.

are the same as the 7T design. However, these two processes are interleaved together, increasing the throughput significantly. In addition, since a certain layer's GC could be stated as long as its activations and errors are ready, the T3 could also be folded with T1/T2 to improve the calculation efficiency. The GC process could be pipelined with FF/BP processes if they have comparable latency. In this way, the off-chip error movement is avoided. The latency of the GC process of each layer could be simply adjusted by duplicating CIM arrays to satisfy the pipeline requirement. An example of the pipelined

timeline is shown in Figure 30 (b). At the 15th cycle, all the pipeline stages are filled. FF/BP stage 1 performs the FF of image 15 and the EC of image 2 at the same time. Meanwhile, the weight gradient of image 1 is calculated in the WGC stage 7 because its activations and errors are both ready after 14th cycle.

In summary, in the 7T-based architecture, the pipeline is adopted only inside each stage. Each stage is executed one by one. The 8T design introduces an inter-pipeline between different training processes in addition to the intra-pipeline of the 7T-based architecture, further improving the training throughput. Besides, the optimized pipeline also improves energy efficiency by reducing off-chip memory. However, the improved parallelism of 8T-based training architecture requires a larger buffer to hold increased on-chip data brought by the inter-pipeline.

5.5 Evaluation Results

For the baseline, we store weights and do near-memory computation with the regular 6T SRAM arrays (i.e., row-by-row read-out with digital adders at the edge of the array to accumulate the partial sum). As non-transpose SRAM is utilized in the baseline, for EC process, we obtain all the elements for one transpose filter each time we read out a row. The products of the filter elements and the inputs are then obtained using a set of 8-bit multipliers. To get the final sum of errors (illustrated in Eq. 2), these products are then accumulated by adders. For weight gradient calculation (illustrated in Eq. 3), the FF activations of each layer are first acquired from off-chip DRAM and then fed into multipliers and adder trees together with the errors, performing bitwise matrix-to-matrix multiplication and accumulation. To perform weight update, all the calculated gradients

will be stored off-chip first. In the end, ΔW_i for batch inputs are accumulated by adders to update the weights by digital circuits. The new weights are then written back to memory arrays. This near-memory computation solution is treated as a baseline to evaluate the advantages of the proposed CIM approaches for on-chip training.

Table 7: 7T CIMAT parameters

Block	Spec.		Area(um^2)	Energy
Technode: 7nm HP		***Input Precision: 8-bit***		
Subarray level (7T)				
7T SRAM Array	Size	128*128	143.41	0.22pJ/op
	Precision	1-bit		
ADC (MLSA)	Number	32	278.95	2.25pJ/op
	Precision	4-bit		
Shift-Add &Register	Number	32	174.34	8.35pJ/op
	Precision	11-bit		
Weight Update	Number	128	52.88	5.76pJ/op
Drivers, MUX and others			200.62	14.95pJ/op
Subarray Total			797.33	25.75pJ/op
PE level				
Subarray	Number	4*4	1.27E04	418.44pJ/op
Adder Tree	Number	128	2,865.37	6.51pJ/op
	Precision	13-bit		
L1 Buffer	Size	128*128	2,066.30	0.01pJ/bit
Output Buffer	Size	32*54	216.30	0.003pJ/bit
Tile level(357 tiles on chip)			***ResNet-18***	
PE	Number	3*3	1.62E05	3,795pJ/op
Adder Tree	Number	128	25,634	29.26pJ/bit
	Precision	17-bit		
L2 Buffer	Size	256*512	16,435	0.01pJ/bit
Output Buffer	Size	32*68	284.09	0.003pJ/bit
Global Buffer(7T)	Size	8MB	8.41E06	0.05pJ/bit

5.5.1 Circuit-level Parameter

We build the CIMAT architecture using a modified version of NeuroSim, taking into account the on-chip buffer and off-chip DRAM access, and simulate the CIMAT design using the most recent 7 nm high performance (HP) CMOS library. Table 7 lists the

Table 8: 8T CIMAT parameters

Block	Spec.		Area(um^2)	Energy
Technode: 7nm HP		***Input Precision: 8-bit***		
Subarray level (8T)				
8T SRAM Array	Size	128*128	163.90	0.45pJ/op
	Precision	1-bit		
ADC (MLSA)	Number	32	278.95	2.25pJ/op
	Precision	4-bit		
Shift-Add &Register	Number	32	174.34	8.35pJ/op
	Precision	11-bit		
Weight Update	Number	128	52.88	5.76pJ/op
Drivers, MUX and others			200.62	14.95pJ/op
Subarray Total			817.82	25.98pJ/op
PE level				
Subarray	Number	4*4	1.30E04	422.12pJ/op
Adder Tree	Number	128	2,865.37	6.51pJ/op
	Precision	13-bit		
L1 Buffer	Size	256*128	4132.60	0.01pJ/bit
Output Buffer	Size	64*54	432.60	0.003pJ/bit
Tile level(357 tiles on chip)			***ResNet-18***	
PE	Number	3*3	1.68E05	3,828pJ/op
Adder Tree	Number	128	25,634	29.26pJ/bit
	Precision	17-bit		
L2 Buffer	Size	512*512	32,870	0.01pJ/bit
Output Buffer	Size	64*68	568.18	0.003pJ/bit
Global Buffer(8T)	Size	20MB	2.1E07	0.05pJ/bit

hardware configuration, accuracy, size, and energy consumption for significant circuit modules along with other circuit-level metrics based on the 7T SRAM architecture. The energy is provided in terms of energy per operation (or bit). For instance, the total energy of a single sub-array for performing a single vector-matrix multiplication is 25.75 pJ/op, while the energy of a single L1 buffer for writing a single bit of data is 0.006 pJ/bit.. The estimated energy cost of off-chip DRAM access is 4.2pJ/bit from prior work [59], assuming 3D high-bandwidth memory is used. According to section 4.4.2, eight cycles are used to send the multi-bit activations to the weight arrays from LSB to MSB, and shift-add is used to sum up the outputs. The calculated MAC value of 4-bit LSBs after ADC is first stored

in the register. Then, the MAC value of 4-bit MSBs will be shifted and added with the stored value in the register. The adder is designed to be 11-bit, which allows for the storage of carry-bit information for each 8-bit shift-add operation.

8MB global buffer is assigned for 7T-SRAM cell based CIMAT design. However, the massive intermediate data generated during the training process is hard to fully store in the on-chip SRAM buffer. To store massive intermediate data for training purposes, activation outputs of each layer a_i (generated in FF) and calculated errors (generated in EC) of each layer for batch input are both sent to off-chip DRAM for reuse in weight gradient calculation. During the GC process, the weight gradient ΔW_n for each image is also stored off-chip DRAM for weight update. The energy efficiency of CIMAT architecture could reach ~20 TOPS/W if the energy cost of off-chip data transfer is excluded. As indicated in Table 7, energy efficiency drops to just 6 TOPS/W when DRAM access is considered. Thus, we propose 8T-SRAM cell based CIMAT design to implement inter-/intra-pipeline, further improving energy efficiency and throughput. Table 8 displays the circuit-level parameters of 8T transpose SRAM-based architecture. Due to the area overhead of additional transistors, the 8T memory array is larger than 7T. Since FF and EC calculation can perform simultaneously in 8T-based CIMAT architecture, the L1, L2 and output buffer in each tile are expanded to support concurrent bi-directional computing. Additionally, in order to handle inter-pipeline among FF, error calculation and gradient calculation, the global buffer size needs to be enlarged in the 8T design, which is 20MB compared to 8MB of 7T design. The pipeline processing in the 8T-based CIMAT is able to reduce DRAM access during feedforward and error calculation, but off-chip data transfer is still necessary during weight gradient calculation. The results of our analysis, as

displayed in Table 9, can support this statement. Compared to the 7T design, 8T could improve energy efficiency by 1.6× (~10 TOPS/W). It is also observable that total energy efficiency is still constrained by significant off-chip memory access, while ~55 TOPS/W can be achieved without considering DRAM access.

Table 9: Benchmark results

Batch input(128 images)			ImageNet
Architecture	**TOPs/W**	**FPS**	**Area(mm^2)**
7T SRAM			
CIMAT_FF	28.11	50,585	67.83
CIMAT_BP	17.23	4,376	5.41
CIMAT_weight update	95.36	2.11E06	negligible
CIMAT Subtotal without DRAM Access	19.84	4,020	81.80
CIMAT_Total (7T)	**6.02**	**4,020**	**81.80**
8T SRAM			
CIMAT_FF	55.53	49,469	78.23
CIMAT_BP			43.28
CIMAT_weight update	95.36	2.11E06	negligible
CIMAT Subtotal without DRAM Access	55.37	48,335	121.51
CIMAT_Total (8T)	**10.79**	**48,335**	**121.51**
Baseline			
Baseline_FF	1.98	10,430	64.15
Baseline_BP	2.38	1,018	96.54
Baseline_weight update	197.25	9.60E07	1.79
Baseline Subtotal without DRAM Access	2.23	927	164.02
Baseline_Total	**1.78**	**927**	**164.02**

5.5.2 Benchmark Evaluation

As described, the baseline architecture is a near-memory computation solution that digital computation units are located at the memory's edge, while the SRAM memory array only serves as weight storage unit. Table 9 compares the performance of the baseline and 7T/8T SRAM-based CIMAT architecture. The batch input for training contains 128 ImageNet images. The energy efficiency (in terms of TOPS/W), the throughput (in terms

of FPS), and the chip area are evaluated for FF , BP and WU processes, respectively. Here, Both error and gradient calculations are included in this BP performance.

As shown in Table 9, the energy efficiency of CIMAT for both FF and EC process is much better than the baseline, supporting our hypothesis that CIM architecture can reduce memory access for convolution computation. Besides, the hardware cost for the BP process is significantly less with the transpose SRAM architecture than with the baseline due to the shared CIM arrays and the removal of extra digital circuits for error calculations. For the EC process, the hardware of gradient calculation of 8T CIMAT has an 8× area size increase over the 7T design, which speeds up gradient calculation by duplicating CIM arrays to implement inter-pipeline in training. In summary, compared to the near-memory computation baseline, 7T SRAM-based CIMAT training architecture can achieve an overall ~4.34× speedup and ~3.38× improvement in energy efficiency with ~0.5× chip area. Meanwhile, the 8T SRAM-based CIMAT can boost energy efficiency and throughput by ~6.06× and ~52.14×, respectively, with 0.74× chip area.

Figure 31 shows the performance comparison among the proposed CIMAT architecture, the GPU-based implementation, the near-memory computation (baseline design), and Neural Cache [33], which is state-of-the-art hardware accelerating DNN inference using SRAM-based in-memory computing architecture. For the GPU platform, the performance are measure using Pytorch running on NVidia Titan RTX, with the NVidia-SMI tool for GPU power measurement. As stated in the reference paper [33] we use 28 TOPS and 52.92 W average power at 22nm technology to evaluate Neural Cache's performance. While comparing to the GPU-based solution, Our evaluation results indicates that CIMAT 7T design can achieve an average 7.4× speedup and 100× energy efficiency

for training tasks. While Comparing to Neural Cache, CIMAT not only could support training instead of inference only, but also provides 78.7× speedup and 53× higher energy efficiency for inference tasks. The performance of CIMAT training over inference with 7T and 8T structures is also presented. As shown in Figure 31 (a), 8T CIMAT design could provide almost the same speed for both inference and training, while the speed of training on 7T CIMAT architecture is limited to only 1/10 of inference. The higher throughput of 8T design comes from the inter-pipeline between training processes, as illustrated in section 5.4.3. As shown in Figure 31 (b), for both 7T and 8T CIMAT designs, training's energy efficiency is only 0.25× of the inference. The reason is that massive off-chip intermediate data access for GC calculation exists during training.

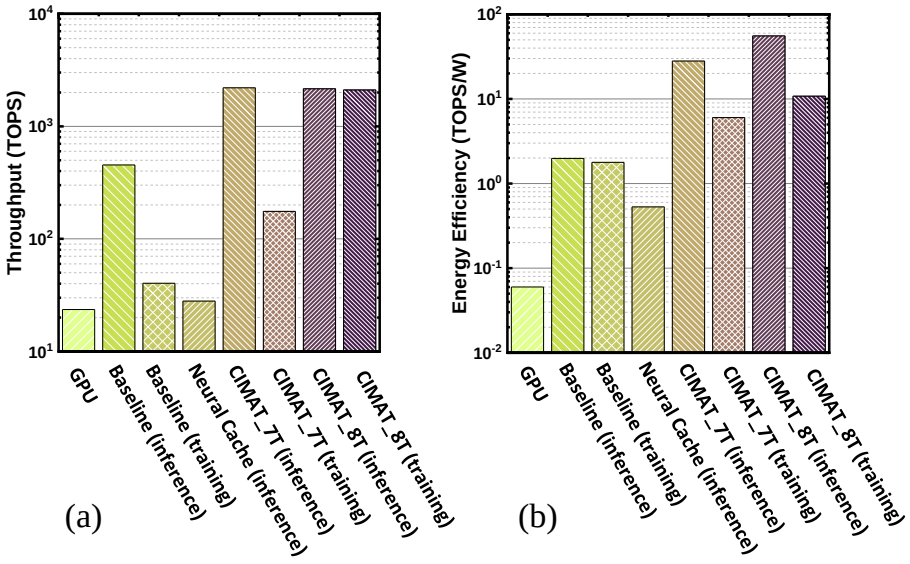


Figure 31: Performance benchmark: (a) Throughput comparison. (b) Energy efficiency comparison

5.6 Summary

In this research, we propose CIMAT, a CIM Architecture for Training. At the bit-cell level, we design two 7T- and 8T-based transpose SRAM cell structures to implement bi-directional computing that is needed for FF and EC processes. Additionally, we design the periphery circuitry, mapping strategy, data flow for backpropagation, and weight update to support the on-chip training based on CIM. To further improve training performance, we explore the pipeline optimization of the proposed architecture. We focus on ResNet-18 implementation, but the methodologies we proposed could be applied to other DNN models. The experimental results demonstrate that 7T-based design can achieve 3.38× higher energy efficiency (6.02 TOPS/W), 4.34× frame rate (4,020 fps) and only 50% chip size compared to the baseline architecture. With a more advanced 8T bit cell and optimized pipeline design, 8T SRAM-based CIMAT can further achieve more energy saving (~10.79 TOPS/W) and aggressively more than 10× throughput (48,335 FPS) with tolerable area overhead compared to 7T-based CIMAT architecture. Our results reveal that CIM is a promising solution to implement on-chip DNN training, which can reduce off-chip talk significantly. Limited on-chip buffer will be a constraint for pipeline implementation of training deeper neural networks on chip.

CHAPTER 6 TWO-WAY SRAM-BASED CIM ACCELERATOR FOR ON-CHIP TRAINING

6.1 Introduction

Despite there have been some efforts to implement on-device learning on CIM systems, several challenges exist in designing a practical accelerator that supports both training and inference. First, One critical problem is how to implement signed number multiplication during backpropagation as negative inputs are involved. This issue deserves to be deeply explored since CIM performs mixed-signal computation instead of pure digital. Second, most of the CIM training solutions are still based on architecture-level design without chip-level validation. The feasibility of the CIM system for training needs to be proven in silicon. Furthermore, the impact of analog-to-digital converter (ADC) resolution for CIM is rarely studied on training performance. For training tasks, drawbacks of eNVMs, including asymmetric conductance tuning, low endurance and large write energy/latency, limit the on-chip training accuracy. On the other hand, SRAM is mature in manufacturing and faster/less energy in write operations (than eNVMs), which benefits the write-intensive on-chip training. In this research, we propose an SRAM-based CIM accelerator for DNN training and evaluate the hardware performance based on several large DNN models with experimental data.

The rest of this chapter is organized as follows: Section 6.2 proposed a signed number multiplication based on 2's complement for CIM. Section 6.3 presents the design of the two-way SRAM CIM array. Section 6.4 presents the overall architecture design based on the proposed SRAM-CIM array. Section 6.5 presents the hardware configurations for the

two-way SRAM-based CIM accelerator. Section 6.6 presents the prototype chip and the comprehensive evaluation results at the system level. Section 6.7 summarizes the chapter.

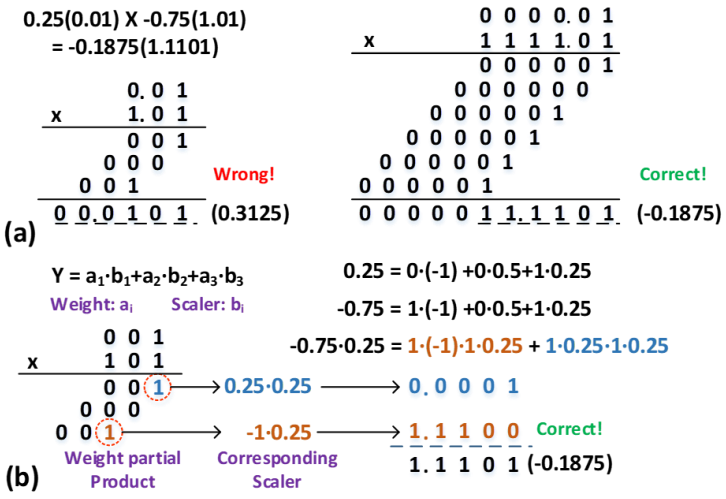


Figure 32: Signed number multiplication (a) Signed extension method (b) Proposed new method

6.2 Signed Number Multiplication

Data is typically represented by 2's complement code for digital calculation for the mathematical calculation in traditional computer systems. However, In CIM systems, dot products are summed up in analog current (which is always a positive value from power supply to ground). Thus, only when input and weight are both positive can the multiplication be claimed to be true. Given that the FF usually only involves positive input data (as ReLU is often used for activations), signed number multiplication may not be a concern for inference engine design. However, for the typical training process, the errors in the backpropagation could be either positive or negative. When a positive value is multiplied by a negative value, the trouble with 2's complement multiplication is that both the multiplicand and the multiplier have to be sign-extended. As illustrated in Figure 32

(a), sign extension is essential to obtain the correct result of the signed number multiplication to calculate 0.25×−0.75=−0.1875 in the binary fashion. For convolution computations, more input cycles could realize sign extension of input with a penalty of latency. Meanwhile, sign extension of weights requires extra arrays, introducing additional area costs. For instance, PRIME [32] stores negative and positive weights separately on different arrays, which thus doubles the memory size.

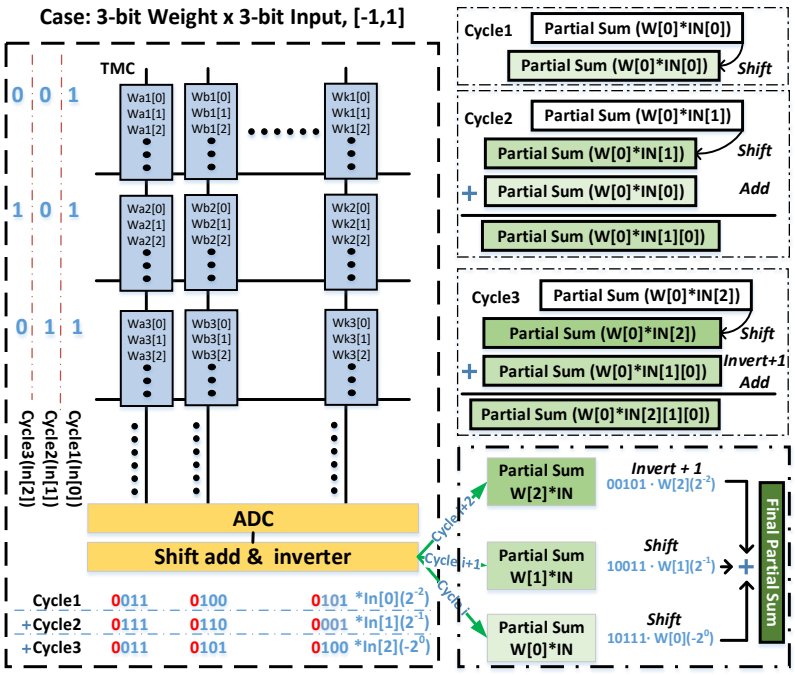


Figure 33: In-memory dataflow of signed VMM

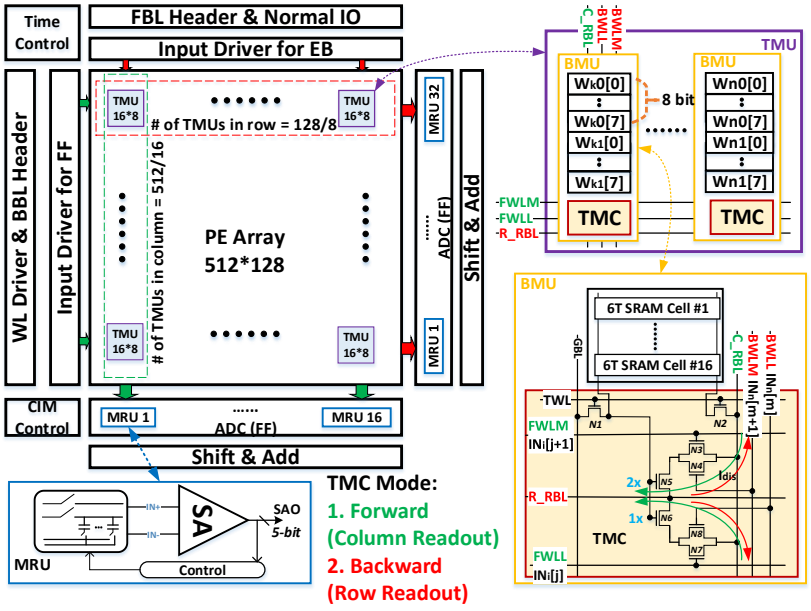
As shown in Figure 32 (b), we instead condcut the unsigned VMM in a single memory array and then get use of the periphery to shift-add or invert the partial sums according to the scale. This method converts a 2's complement value back to decimal. Despite the fact that the 2's complement value contains sign information by itself, we ignore the sign information at first in the weighted sum operation. In the same example as

shown in Figure 32, 001×101 will generate two 1's (in the red circle), the first "1" has a scale of 0.25×0.25, thus we need to shift it by four times to obtain 0.0001; the second "1" has a scale of -0.25, thus we need to shift it by twice and then perform 2's complement conversion (i.e., invert and add 1) to obtain 1.1100. Finally, we add these two values together $0.0001 + 1.1100 = 1.1101$, which is the correct 2's complement code for -0.1875.

There is no need to worry about sign information as they are all positive in our approach. The binary input/weight sequence is treated as the unsigned weighted sum in the crossbar array. The scale and sign of the partial sums are then multiplied back to obtain the 2's complement representation. The architecture and corresponding dataflow is presented in Figure 33. different rows are used to store different significances of the multi-bit weight, e.g., *W[0], W[1],* and *W[2],* and multiple cycles are employed to represent the multi-bit input data. For the same significant bit of weights in the same column, the MAC outputs of each cycle are multiplied by the input scales and summed up by shift-add. For example, if input precision is 3-bit, the output of *Cycle 1* will be shifted first. The output of *Cycle 2* will be shifted and then added together with the output of *Cycle 1*. Next, the MAC output of *Cycle* 3 will be inverted and added 1 (to represent the negative sign) and then added with the first two cycles' partial sum.

Partial sums are further weighted and summed by the weight scales according to the weight significance. For example, *W[0]*'s partial sum will be first shifted, and *W[1]*'s partial sum will be shifted and added together with *W[0]*'s. Next, *W[2]*'s partial sum will be inverted and added 1 and then added with the other two bits' partial sums, resulting in the final sum. In general, the range of weights and inputs is between [-1,1], and the

quantization scales will always be -2^0, 2^{-1} to $2^{-(n-1)}$. Therefore, we can operate the scaling back using a shift-adder, except for the highest bit. For the highest bit corresponding to -2^0(-1 sign), we directly invert all the bits and add 1 to obtain the 2's complement value.



	Forward/Backward	Binary value							
Input	FWLM/BWLM	1	1	0	0	1	1	1	1
	FWLL/BWLL	1	0	1	0	1	0	1	0
Weight	LBL	1				0			
Product (In * W)	R_RBL/C_RBL	Sum value							
		$3I_{dis}$	$2I_{dis}$	I_{dis}	0	0	0	0	0
		$-3\Delta V$	$-2\Delta V$	$-\Delta V$	0	0	0	0	0

(b)

Figure 34: (a) Overall structure of two-way SRAM array design. (b) Truth table and operation mode for TMC.

6.3 Design of Two-way SRAM Array

In practice, there is room to modify the bit-cell structure and reroute the in-array interconnection in old technology nodes (e.g., 65nm), while foundries typically do not allow any customizations on memory design in advanced technology nodes (e.g., 28nm or beyond). Another approach to implement in-memory computing is grouping the normal

SRAM cells into memory sub-banks and embedding compute cells between sub-banks. The circuit diagram of the two-way CIM SRAM arrays is presented in Figure 34 (a). One single array comprises 16×32 transpose-multiply-units (TMUs), 16 multibit-readout-units (MRUs) for FF process, 32 MRUs for EC process, input drivers for FF and EC, and other peripheral circuits. Each TMU contains 8 bitwise-multiply-units (BMUs) across 8 columns, in which two 8-bit weights are stored in the 16 SRAM cells on the same column. In total, 16 standard 6T SRAM cells as well as 1 transpose-multiply cell (TMC) are present in each BMU. Each TMC unit comprises 10 transistors, including 2 pass-gate transistors (*N1, N2*) and 2 multiply branches (*N3-N5* and *N6-N8*). In standby mode, all *N1/N2* are on and the global BLs charge local BLs/BLBs to a precharge voltage. For computation, TMC has two modes, forward mode and backward mode, to support bi-directional bitwise multiplication, as shown in Figure 34 (a). In forward mode, row-read bitlines (R_RBLs) are grounded. A 2-bit input is applied to the gates of *N3* (FWLM) and *N7* (FWLL), corresponding to $IN_i[j+1]$ and $IN_i[j]$, respectively, in each input cycle,. When a wordline (WL) is activated, the weight stored in the selected 6T SRAM is read out, multiplying 2-bit input through 2 multiply branches at local BL. The more-significant-bit (MSB) multiplication and less-significant-bit (LSB) multiplication of input are calculated through the *N3-N5* pair and *N6-N8* pair, respectively. The transistor width of *N3-N5* is designed to be twice that of *N6-N8*, enabling the *N3-N5* pair to produce twice as much discharge current as the *N6-N8* branch. When the two currents of both *N3-N5* and *N6-N7* pairs are summed up, the voltage swing (ΔV) contributed by the TMC cell to column-read-bitline (C_RBL) is proportional to the result of multiplying 2-bit input and 1-bit weight. In backward mode, different significant inputs, $IN_n[m+1]$ and $IN_n[m]$, are fetched into BWLM and BWLL

and C_RBL is set at 0V. The N4-N5 pair in TMC performs the bitwise multiplication of $IN_n[m+1] \times W$, while the N6-N8 pair performs $IN_n[m] \times W$. Then the two discharge currents from *N4-N5* and *N6-N8* branches are combined, the voltage swing produced by TWC cell to R_RBL is proportional to the multiplication result of column input and stored weight. All the TMUs in the same column share the same C_RBL while all the TMUs in the same row share the same R_RBL.

The truth table in Figure 34 (b) lists possible combination patterns of 2-bit input×1-bit weight and the corresponding multiplication results. The difference of ΔV value on R_RBL/C_RBL reflects the product of 2-bit input and binary weight effectively. The forward mode is activated for FF calculation: 16 selected vertical TMCs from the same column are grouped, and all the discharge currents from the same group are totaled, performing a MAC operation. The partial sum of 2-bit input multiplication with 1-bit weight is presented by the generated analog voltage V_{C_RBL},which is then quantized to digital output by MRU. The final sum of multi-bit input multiplication with multibit weight can be obtained by shift-adding the partial sum of different significant bits of weight and input successively. For EC process, 16 horizontal TMCs from the same row are chosen in backward mode. Each R_RBL accumulates products from TMUs in the same row and generates an analog sum voltage V_{R_RBL}, which is then read out by MRU. Figure 34 illustrates the MRU basic structure built with a low-offset sense amplifier (SA), control block, and capacitor-sharing unit. MRUs repeatedly perform SA operations with five reference voltages through 5 cycles to generate a 5-bit digitalized output, effectively acting as 5-bit ADC.

6.4 Overall Architecture

Figure 35 presents the overall architecture of our two-way SRAM-based CIM accelerator. The accelerator is hierarchically composed of subarrays, tiles, digital blocks, and on-chip buffers from the bottom to the top level. One tile consists of 32 proposed subarrays and digital blocks, including accumulators and adder trees. Each tile has a weight capacity of 256kB. Figure 35 also depicts the training dataflow according to the proposed mapping strategy [7]. Performing the perpendicular MAC operations, our bi-directional SRAM array design could eliminate the additional hardware for error calculation.

First, the green line and arrow indicate that activations are fed into two-way SRAM arrays to perform forward MAC, resulting in partial sums. By accumulating these partial sums, the final activation output of each layer is acquired, which will be then sent to the off-chip buffer for further reuse. Second, after FF, as shown in the red lines and arrows, subarrays perform backward MAC to implement error backpropagation. For GC computation, prior works usually need additional hardware to compute the gradient. Inspired by our earlier work [7], we execute gradient computation inside the SRAM memory: we reuse the same hardware for FF and EC process to calculate the gradient, which can reduce both chip area and total SRAM leakage. The blue line and arrows illustrate that calculated errors will be first loaded into SRAM array during GC, and then each layer's activations are fed in as the array’s inputs to perform VMM calculation to obtain weight gradient. The to-be-updated weights are calculated inside the digital blocks. Finally, as shown in the purple line, new weights will be sent back to each subarray to update existing weights via standard SRAM’s write operation.

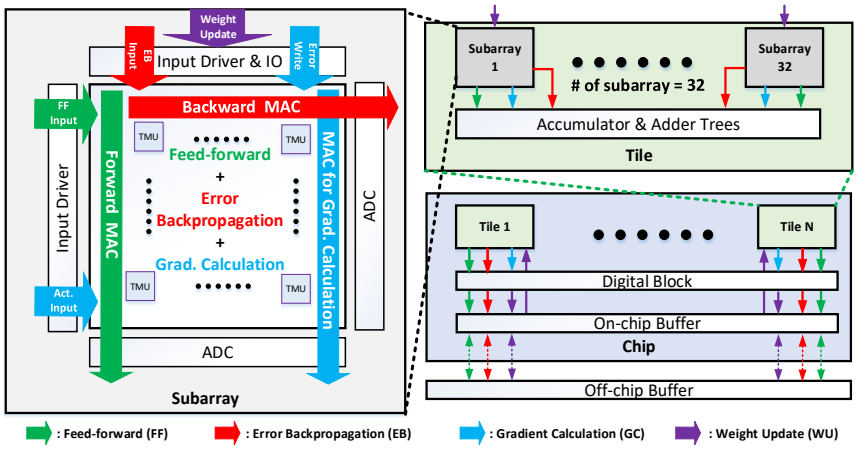


Figure 35: The overall architecture and DNN training dataflow on two-way SRAM design.

6.5 Hardware Configuration

Different configurations of the SRAM array will directly affect the accuracy performance and the hardware cost in the CIM accelerator design. The array size and ADC precision are the two primary design parameters of CIM. First, we need to consider the neural network's precision while the array size. Traditional processors, such as GPUs and TPUs widely used in servers for DNN online training, usually support 64-bit floating-point or 16-bit fixed-point operations. However, area overhead and energy costs will be significant for edge devices if such high precision is used. Therefore, there are many efforts to lower the computational cost by quantizing the parameter precision. Several approaches have been proposed to realize low-bit precision training. The effectiveness of training with 1-bit weight in FF has been demonstrated by BNN [60] and XNOR-Net [17], but all other parameters are still floating-point. DoReFa-Net [61] further lowers the gradient precision to reduce the high-precision copy needed for gradient, showing that 1b-weight, 2b-activation, and 6b-gradient is adequate for training. Although low precision gradient is claimed in DoReFa-Net, its errors are still floating-point. Moreover, WAGE [42]

introduced quantization in all parameters, including errors, to realize low-precision training with 2b-weight, 8b-activation, 8b-gradient, and 8b-error. While these methods perform excellently on small to medium-sized networks for MNIST and CIFAR-10 datasets, they all show a visible accuracy drop when applied to large networks for ImageNet dataset. Most of the results reported so far are for the simple AlexNet and VGG-16, which are large in terms of parameters but relatively low in accuracy. R. Banner et al. [62] present that 16b-gradient could maintain high accuracy on ResNet networks. Recently, an advanced version of WAGE, namely WAGEBU [63], also indicates that a high-precision setting for the gradient is essential for scalability. Consequently, based on these studies, we select a low-precision quanzitzation with 8b-weight, 8b-activation, 8b-error, and 16b-gradient as a safety option in our training accelerator design.

To support different applications, as presented in section 6.3, we assign 16 SRAM cells to share one TMC, which means we can theoretically support 1-bit to 16-bit weight precision. The array size influences the ideal precision of partial sum as well as hardware performance. Theoretically, larger array sizes lead to higher throughput while ADC offset becomes worse than smaller array sizes. Besides, the most efficient size also depends on different network filter sizes. Combining all the above considerations, we choose 512×128 as our SRAM array size, which can store 64×128 8-bit weights. This array size is appropriate for a variety of moderate networks, ranging from VGG-8 to large networks such as ResNet-18/34 and DenseNet-121/169 with 7 ~ 22 million parameters.

To determine ADC resolution, we need to examine the impact of ADC quantization on accuracy performance. However, prior studies only explored the ADC impact on inference. Ideally, the easiest way is to hire a high-resolution ADC to prevent any

quantization loss. However, when ADC precision goes higher, the area and power of ADC will increase dramatically. Our two-way SRAM array fetches 2-bit input and accumulates 16 rows at once, which means the ideal no-loss precision of each MAC operation is 6-bit. Accordingly, we build the simulation testbench on PyTorch platform to explore the impact of ADC resolution (from 4-bit to 6-bit) on the training performance. We test our design with VGG-8 network on CIFAR-10 dataset. The curves in Figure 36 present that 5-bit ADC design could maintain the same accuracy as 6-bit full precision, which means 1-bit quantization loss is tolerable. Meanwhile, 4-bit ADC causes ~1% accuracy drop compared to the baseline accuracy. Based on our experiments, we decide to use 5-bit ADC in our SRAM array, which could achieve the same accuracy but with a smaller area cost than the full precision design (i.e., 6-bit).

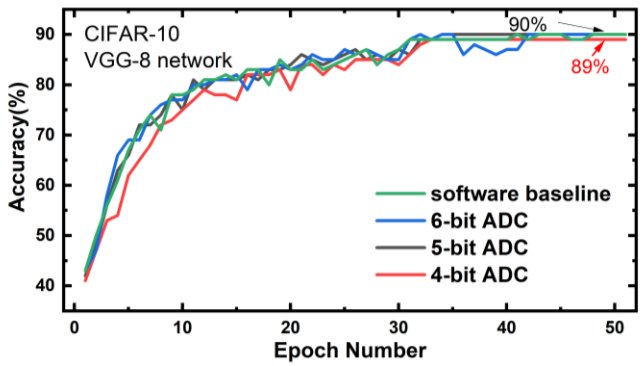


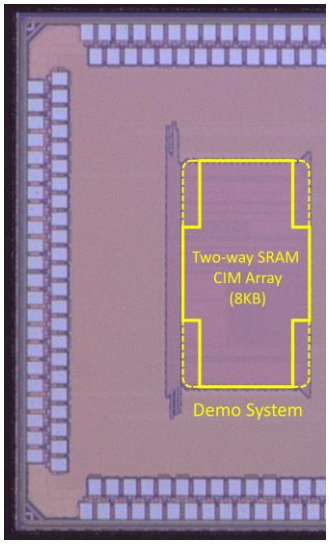
Figure 36: The impact of ADC resolution on training performance

6.6 Evaluation

In this section, the fabricated two-way SRAM CIM macro is presented. The proposed architecture designs are systematically evaluated. Finally, we build the benchmarks for comparison across various DNN models. In our benchmarks for ImageNet, we employ four DNN models: ResNet-18. ResNet-34, DenseNet-121 and DenseNet-169. The system

performance is evaluated in a hybrid approach: the parameters of SRAM-based CIM macro are obtained from silicon measurements; the tile-level peripheral digital blocks and chip-level modules are estimated with NeuroSim framework.

(a)



(b)

Technology	TSMC 28nm			
Supply voltage(V)	0.85 – 1			
SRAM array size	8KB (512 row × 128 column)			
Mode	Forward		Backward	
Input precision (bit)	8		8	
Weight precision (bit)	4	8	4	8
Output precision (bit)	16	20	16	20
Process time (ns)	15.2-17.2	30.4-34.4	18.5-21.0	37.0-42.0
Energy (pJ)	67.48-73.10	67.51-73.20	136.02-142.02	136.25-142.23
memory area (no periphery)	1.68×			
Array area (total)	1.49×	2.98×	1.49×	2.98×

Figure 37: (a) Die photo of Two-way SRAM macro. (b) Summary table of silicon measurement data.

6.6.1 Prototype Chip

The die photo and measurement results of the fabricated prototype chip are presented in Figure 37. The 64kb SRAM array has 512 rows and 128 columns. The subarray's partial sum precision is 20-bit for 8-bit input and weight. Compared to regular SRAM cells, BMU's area overhead is only 1.68×. As shown in Figure 37 (b), EC's energy cost is about twice that of the FF process since forward MAC is processed in 16 columns simultaneously while backward MAC is processed in 32 rows in parallel. Compared to the conventional SRAM array for cache with the same array size, the area overhead of the proposed two-way SRAM array is 2.98× owing to transpose-multiplication-cells and ADCs.

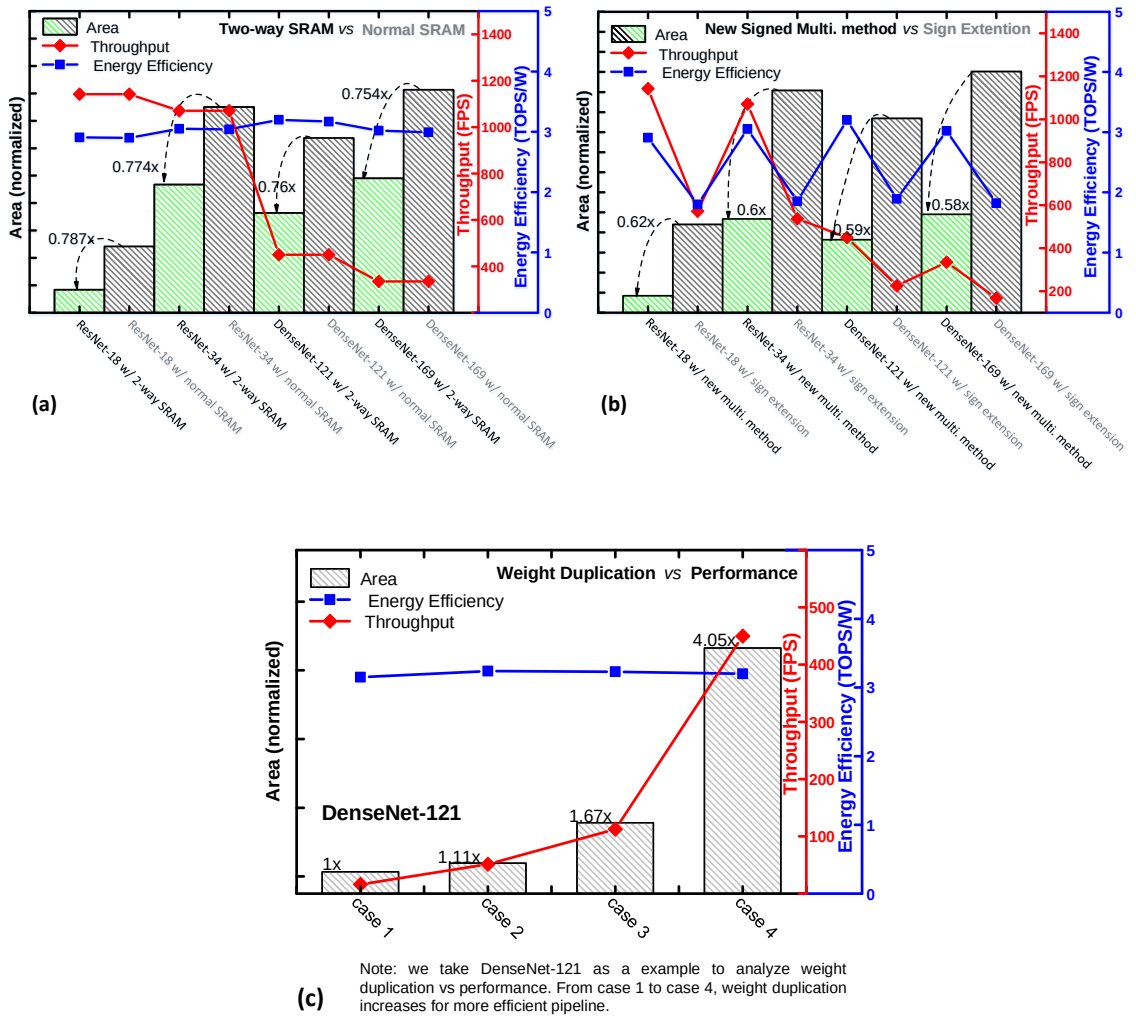


Figure 38: (a) Performance improvement with two-way SRAM design. (b) Performance improvement with proposed signed number multiplication. (c) Performance comparison when increasing the weight duplication on DenseNet-121

6.6.2 Benchmark Results and Discussion

Figure 38 (a) illustrates the system-level performance comparison between our two-way SRAM-based CIM design and conventional SRAM-CIM architecture. Across different networks, it is observable that the proposed two-way SRAM design could reduce chip area by ~25% while maintaining the throughput or energy efficiency. Figure 38 (b) shows the evaluation of the proposed signed number multiplication compared to the

traditional approach, which directly performs sign extension multiplication. The new multiplication method dramatically boosts hardware performance in terms of space, throughput, and energy efficiency. As an illustration, our proposed design could achieve 2× throughput and 1.6× energy efficiency with only 0.6× area cost tested on ResNet-18. While comparing the performance of ResNet and DenseNet, we can find that our design shows larger throughput on ResNet than DenseNet. The reason is that the ResNet model needs fewer computations with fewer layers. Meanwhile, weight duplication for slow layers in a pipeline design allows further speed-up. Taking DenseNet-121 as an example, we explore the trade-off between weight duplication and performance presented in Figure 38 (c). We can see that the throughput and hardware area cost is inversely correlated.

Figure 39 (a) shows the energy breakdown among training steps. Gradient calculation takes more energy than FF and EC due to less reuse of the error matrix and frequent write to reload error matrix. Figure 39 (b) shows the energy breakdown of CIM computing, SRAM leakage, and off-chip DRAM access. SRAM leakage is almost eliminated since we keep reusing the SRAM array during the entire training process to shorten standby time. DRAM access is still a significant part of the total energy since GC and WU process needs to access DRAM frequently due to limited on-chip buffer. Table 10 compares this work to state-of-the-art accelerator designs from digital ASIC to CIM approaches. It is seen that our proposed design achieves the highest energy efficiency and throughput, which could support multi-bit training on large datasets, while most prior works only support low-bit inference on small to moderate datasets. Our SRAM-based CIM accelerator also holds advantages (~3.2 TOPS/W) compared to GPU (~0.1 TOPS/W) or TPU (~0.45 TOPS/W for the training version).

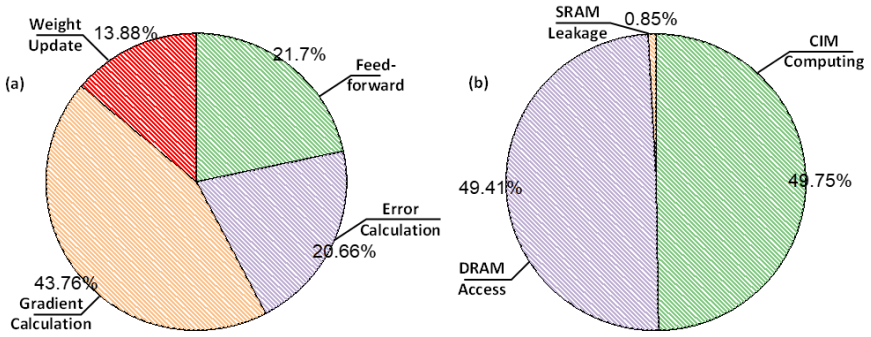


Figure 39: Two-way SRAM architecture-level benchmark result: (a) Energy breakdown of training steps (b) Energy breakdown of operations

Table 10: Comparison with recent ASIC and CIM designs

	JSSCC'17[1]	ISSCC'18[18]	DAC'18[16]	JSSCC'19[20]	ISSCC'19[19]	This work
Technology	65nm	65nm	65nm	65nm	55nm	28nm
Algorithm	AlexNet	SVM	BNN, XNOR-Net	CNN	CNN	VGG,ResNet DenseNet
Dataset	ImageNet	MIT-CBCL	CIFAR10	MNIST, CIFAR10	MNIST, CIFAR10	CIFAR10, ImageNet
Training support	No.	Yes	No.	No.	No.	Yes
Architecture	Digital	SRAM-based CIM	SRAM-based CIM	SRAM-based CIM	SRAM-based CIM	SRAM-based CIM
Precision	16-bit	8-bit	1-bit	1-bit	4-bit	8-bit
MAC TOPS/W	-	3.125	50	-	18.37	17.2
Total TOPs/W	0.023	-	-	1.25	-	3.15-3.2
TOPS	0.12	0.004	-	0.0432	0.09	6.1-23.2

6.7 Summary

This work proposes an efficient SRAM-based CIM accelerator aimed at DNN training. Compared to prior works, we make the following contributions. Firstly, we propose an efficient solution to implement in-memory signed multiplication, which performs unsigned in-memory VMM and then multiple rounds of shift-add or invert operations in the periphery. Secondly, we design a two-way SRAM array to support both

row-wise and column-wise computation. With bi-directional access, forward and backward propagation calculations can be implemented in the same array, improving the area efficiency over the prior design. Thirdly, we build our accelerator and optimize the workflow for DNN in-memory training, which can perform feed-forward calculation, error calculation, and gradient calculation within the same proposed SRAM array design. The impact of ADC quantization loss is also explored for DNN training. Finally, Based on the silicon measurement data on the prototype chip, the hardware performance for the entire architecture for DNN on-chip training is systematically explored. The experimental data shows that the proposed accelerator can achieve energy efficiency of ~3.2 TOPS/W, >1000 FPS, and >300 FPS for ResNet and DenseNet training on ImageNet, respectively.

CHAPTER 7 CONCLUSION AND OUTLOOK

7.1 Summary of Presented Works

This dissertation investigates the prospects and challenges of the CIM accelerators for both DNN inference and training, and explores the architecture/circuit design optimization for CIM accelerators. This thesis presents four major works, of which the conclusions and contributions are summarized as follows:

We comprehensively explore conventional ADC designs (Flash vs. SAR) for CIM array. Then we investigate a new data conversion scheme that performs the analog shift-add for multiple weight significance bits, namely analog shift-add ADC. The impact of the ADC precision on inference accuracy performance is thoroughly analyzed and illustrated for the representative CIFAR-10 dataset based on a multi-bit VGG-8 network. We benchmark the hardware performance of CIM arrays with various ADC designs at given similar area constraints. The analog shift-add ADC achieves 37× and 4.9× higher EDP performance, compared to Flash-ADC, and SAR-ADC, respectively. Meanwhile, the area cost of analog shift-add ADC for CIM design is only 77% of Flash-ADC.

After that, we propose ENNA, a novel CIM architecture based on an ADC-free sub-array design, implementing inter-array data processing in an analog manner. A lightweight input encoding scheme based on pulse-width modulation (PWM) is proposed to improve the throughput. We tape out a prototype macro and validate the proposed ADC-free RRAM array design in TSMC 40nm process. Based on the measured silicon data, we explore the system-level performance with a partition between analog and digital processing at a level higher than the sub-array. The evaluation results show that the proposed accelerator can

achieve 73.6~86.4 TOPS/W energy efficiency and 2.3~7 TOPS throughput (normalized to binary operation) tested on various DNN models. Furthermore, we project the proposed design with heterogeneous 3D integration technology to explore the potential of CIM design with 3D stacking technology, showing a 3×~37× throughput improvement depending on different tasks and ~50% reduced area overhead compared to 2D architecture.

For on-chip training exploration, we first present CIMAT, a CIM Architecture for Training. We propose a novel transpose weight mapping strategy. We design the periphery circuitry and the dataflow for the BP process and weight update to support the on-chip training based on CIM. To further improve training performance, we explore the pipeline optimization of the proposed architecture. We utilize the mature and advanced CMOS technology at 7 nm to design the CIMAT architecture with 7T-/8T-based transpose SRAM array that supports a bi-directional parallel read. We explore the 8-bit training performance of ImageNet on ResNet-18, showing that 7T-based design can achieve 3.38× higher energy efficiency (6.02 TOPS/W), 4.34× speedup and only 50% chip size compared to the baseline architecture with a conventional 6T SRAM array that supports row-by-row read only. The even better performance is obtained with 8T-based architecture, which can reach 10.79 TOPS/W and more than 40× speedup with 74% chip area compared to the baseline.

Lastly, A two-way SRAM-based CIM accelerator that could perform bi-directional in-memory VMM with minimum hardware overhead is proposed for on-chip training support. A novel solution of signed number multiplication is proposed to handle the negative input in backpropagation. Moreover, we explore ADC quantization's impact on training performance. The proposed two-way SRAM array design is validated with a tape-

out in TSMC 28 nm process. The architecture-level performance for DNN on-chip training is evaluated from the measured silicon data of CIM macro, which shows ~3.2 TOPS/W energy efficiency, with over 1000 FPS on ResNet-18/34 and over 300 FPS on DenseNet-121/169.

7.2 Future Works

Firstly, more economical peripheral circuits are always preferred at the circuit level. Regarding the ADC challenge, more efforts should be made to alleviate or remove the ADCs (and DACs) in CIM designs. ADCs/DACs are vital in conventional CIMs because we perform analog computing inside the CIM arrays while requiring digital data storage and transfer beyond CIM arrays. Hence, there are two future directions we can think about: 1) Performing inter-/intra-array computing all in the digital domain. As CMOS technology scales down with more non-idealities and less design space, digital-based CIM tends to show more advantages with higher reconfigurability, robustness, and scalability than mixed-signal CIM. Besides, scaling supply voltage to lower the energy cost should be more friendly for all-digital approaches. 2) Performing all operations in the analog domain without any data conversion. Charge-/Pulsewidth-based analog computing will be a good try. Meanwhile, potential challenges, such as PVT variations and analog data storage, must be faced.

Secondly, for training complex datasets such as ImageNet, the system performance of CIM architectures is still bounded by off-chip data transfer due to limited on-chip capacity. 3D integration could be a promising approach to stack massive amount of embedded memories that are required in state-of-the-art AI accelerators. Besides, in CIM

designs, by partitioning the circuit modules in hybrid technology nodes across different tiers, the challenges of ADC/DAC overhead and scaling limitations caused by high write voltage in eNVM could be addressed. 3D CIM accelerator could potentially achieve higher bandwidth and larger capacity, which enables the on-chip training ability for edge devices.

REFERENCES

[1] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," in *ACM International Symposium on Computer Architecture (ISCA)*, 2017.

[2] Y.H. Chen, T. Krishna, J.S. Emer, V. Sze, "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127-138, 2016.

[3] S. Yu, P.-Y. Chen, "Emerging memory technologies: recent trends and prospects," *IEEE Solid State Circuits Magazine*, vol. 8, no. 2, pp. 43-56, 2016.

[4] H. Jiang, W. Li, S. Huang, S. Cosemans, F. Catthoor, S. Yu, "Analog-to-Digital Converter Design Exploration for Compute-in-Memory Accelerators," *IEEE Design & Test*, vol. 39, no. 2, pp. 48-55, 2022.

[5] H. Jiang, S. Huang, W. Li, S. Yu, "ENNA: An Efficient Neural Network Accelerator Design Based on ADC-Free Compute-In-Memory Subarrays," *IEEE Transactions on Circuits and Systems I: Regular Papers*, early access, 2022.

[6] H. Jiang, W. Li, S. Huang, S. Yu, "A 40nm Analog-Input ADC-Free Compute-in-Memory RRAM Macro with Pulse-Width Modulation between Sub-arrays," in *IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits*, 2022.

[7] H. Jiang, X. Peng, S. Huang, S. Yu, "CIMAT: A Compute-In-Memory Architecture for On-chip Training Based on Transpose SRAM Arrays," *IEEE Transactions on Computers*, vol. 69, no. 7, pp. 944-954, 2020.

[8] H. Jiang et al., "A Two-way SRAM Array based Accelerator for Deep Neural Network On-chip Training," in *ACM/IEEE Design Automation Conference (DAC)*, 2020.

[9] J.-W. Su, et al., "a 28nm 64kb inference-training two-way transpose multibit 6t sram compute-in-memory macro for AI edge chips," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020.

[10] D. Patterson, K. Asanovic, A. Brown; R. Fromm, J. Golbus, B. Gribstad, K. Keeton, C. Kozyrakis, D. Martin, S. Perissakis, R. Thomas, N. Treuhaft, K. Yelick, "Intelligent RAM (IRAM): the industrial setting, applications, and architectures," in *International Conference on Computer Design VLSI in Computers and Processors*, 1997.

[11] D.G. Elliott, M. Stumm, W.M. Snelgrove, C. Cojocaru, R. Mckenzie, "Computational RAM: implementing processors in memory," *IEEE Design & Test of Computers*, vol. 16, no. 1, pp. 32 - 41, 1999.

[12] T. Song, J. Jung, W. Rim, H. Kim, Y. Kim, C. Park, J. Do, S. Park, S. Cho, H. Jung, B. Kwon, H.-S. Choi, J. Choi, J. S. Yoon, "A 7nm FinFET SRAM using EUV lithography with dual write-driver-assist circuitry for low-voltage applications," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2018.

[13] A. Kawahara, R. Azuma, Y. Ikeda, K. Kawai, Y. Katoh, K. Tanabe, T. Nakamura, Y. Sumimoto, N. Yamada, N. Nakai, S. Sakamoto, Y. Hayakawa, K. Tsuji, S. Yoneda, A. Himeno, K. Origasa, K. Shimakawa, T. Takagi, T. Mikawa, K. Aono, "An 8Mb multi-layered cross-point ReRAM macro with 443MB/s write throughput," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2012.

[14] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition," in *IEEE conference on computer vision and pattern recognition*, 2016.

[15] J. Zhang, Z. Wang, N. Verma, "A machine-learning classifier implemented in a standard 6T SRAM array," in *IEEE Symposium on VLSI Circuits*, 2016 .

[16] W.-S. Khwa, J.J. Chen, J.F. Li, X. Si, E.Y. Yang, X. Sun, R. Liu, P.Y. Chen, Q. Li, S. Yu, M.F. Chang, "A 65nm 4Kb algorithm-dependent computing-in-memory

SRAM unit-macro with 2.3ns and 55.8TOPS/W fully parallel product-sum operation for binary DNN edge processors," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2018.

[17] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," in *14th European Conference on Computer Vision (ECCV)*, 2016.

[18] R. Liu, X. Peng, X. Sun, W.S. Khwa, X. Si, J.J. Chen, J.F. Li, M.F. Chang, S. Yu, "Parallelizing SRAM arrays with customized bit-cell for binary neural networks," in *ACM/IEEE Design Automation Conference (DAC)*, 2018.

[19] Z. Jiang, S. Yin, M. Seok, J.S. Seo, "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks," in *IEEE Symposium on VLSI Circuits*, 2018.

[20] X. Si, J.-J. Chen, Y.-N. Tu, W.-H. Huang, J.-H. Wang, W.-C. Wei, S.-Y. Wu, X. Sun, R. Liu, S. Yu, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, Q. Li, M.-F. Chang, "A twin-8T SRAM computation-in-memory macro for multiple-bit CNN-based machine learning," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2019.

[21] Q. Dong et al, "A 351TOPS/W and 372.4GOPS Compute-in-Memory SRAM Macro in 7nm FinFET CMOS for Machine-Learning Applications," in *IEEE International Solid- State Circuits Conference (ISSCC)*, 2021.

[22] Y.D. Chih, P.H. Lee, H. Fujiwara, Y.C. Shih, C.F. Lee, R. Naous, Y.L. Chen, C.P. Lo, C.H. Lu, H. Mori, W.C. Zhao, "An 89TOPS/W and 16.3 TOPS/mm 2 All-Digital SRAM-Based Full-Precision Compute-In Memory Macro in 22nm for Machine-Learning Edge Applications," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2021.

[23] Y. Chen, "ReRAM: History, Status, and Future," *IEEE Transactions on Electron Devices* , vol. 67, no. 4, pp. 1420-1433, 2020.

[24] T. Kim, S. Lee, "Evolution of phase-change memory for the storage-class memory and beyond," *IEEE Transactions on Electron Devices*, vol. 67, no. 4, pp. 1394-1406, 2020.

[25] T. Mikolajick, U. Schroeder, S. Slesazeck, "The Past, the Present, and the Future of Ferroelectric Memories," *IEEE Transactions on Electron Devices,* vol. 67, no. 4, pp. 1434-1443, 2020.

[26] W.H. Chen, K.X. Li, W.Y. Lin, K.H. Hsu, P.Y. Li, C.H. Yang, C.X. Xue, E.Y. Yang, Y.K. Chen, Y.S. Chang, T.H. Hsu, "A 65nm 1Mb nonvolatile computing-in-memory

ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2018.

[27] C.X. Xue et al, "A 1Mb multibit ReRAM computing-in-memory macro with 14.6ns parallel MAC computing time for CNN-based AI edge processors," in *IEEE International Solid- State Circuits Conference (ISSCC)*, 2019.

[28] C.X. Xue, et al, "A 22nm 2Mb ReRAM Compute-in-Memory Macro with 121-28TOPS/W for Multibit MAC Computing for Tiny AI Edge Devices," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020.

[29] X. Sun, S. Yin, X. Peng, R. Liu, J.-S. Seo, S. Yu, "XNOR-RRAM: A scalable and parallel resistive synaptic architecture for binary neural networks," in *IEEE/ACM Design, Automation & Test in Europe (DATE)*, 2018.

[30] J. -H. Yoon, M. Chang, W. -S. Khwa, Y. -D. Chih, M. -F. Chang, A. Raychowdhury, "A 40nm 64Kb 56.67TOPS/W Read-Disturb-Tolerant Compute-in-Memory/Digital RRAM Macro with Active-Feedback-Based Read and In-Situ Write Verification," in *EEE International Solid- State Circuits Conference (ISSCC)*, 2021.

[31] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, V. Srikumar, "ISAAC: A convolutional neural network accelerator

with in-situ analog arithmetic in crossbars," in *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2016.

[32] P. Chi, S. Li, Z. Qi, P. Gu, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, Y. Xie, "PRIME: A novel processing-In-memory architecture for neural network computation in ReRAM-based main memory," in *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2016.

[33] C. Eckert, X. Wang, J. Wang, A. Subramaniyan, R. Iyer, D. Sylvester, D. Blaaauw and R. Das,, "Neural cache: Bit-serial in-cache acceleration of deep neural networks," in *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2018.

[34] M. Cheng, L. Xia, Z. Zhu, Y. Cai, Y. Xie, Y. Wang, H. Yang, "TIME: A training-in-memory architecture for RRAM-based deep neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 5, pp. 834-847, 2019.

[35] L. Song, X. Qian, H. Li, Y. Chen, "PipeLayer: A pipelined RRAM-based accelerator for deep learning," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2017.

[36] X. Peng, R. Liu, S. Yu, "Optimizing weight mapping and data flow for convolutional neural networks on processing-in-memory architectures," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 4, pp. 1333-1343, 2020.

[37] S. Angizi, Z. He, A. S. Rakin, D. Fan, "CMP-PIM: An Energy-Efficient Comparator-based Processing-In-Memory Neural Network Accelerator," in *ACM/IEEE Design Automation Conference (DAC)*, San Francisco, 2018.

[38] A. Ankit, et al., "PUMA: A programmable ultra-efficient memristor-based accelerator for machine learning inference," in *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, New York, 2019.

[39] W. He, S. Yin, Y. Kim, X. Sun, J.J. Kim, S. Yu and J.S. Seo, "2-Bit-per-Cell RRAM based In-Memory Computing for Area-/Energy-Efficient Deep Learning," *IEEE Solid-State Circuits Letters*, vol. 3, pp. 194-197, 2020.

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

[41] X. Peng, S. Huang, Y. Luo, X. Sun, S. Yu, "DNN+NeuroSim: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators with Versatile

Device Technologies," in *IEEE International Electron Devices Meeting (IEDM)*, San Fancisco, 2019.

[42] S. Wu, G. Li, F. Chen, L. Shi, "Training and Inference with Integers in Deep Neural Networks," in *6th International Conference on Learning Representations (ICLR)*, 2018.

[43] P.-Y. Chen, X. Peng, S. Yu, "NeuroSim+: An integrated device-to-algorithm framework for benchmarking synaptic devices and array architectures," in *IEEE International Electron Devices Meeting (IEDM)*, 2017.

[44] A. Jourdain et al., "Extreme Wafer Thinning and nano-TSV processing for 3D Heterogeneous Integration," in *IEEE Electronic Components and Technology Conference (ECTC)*, 2020.

[45] X. Peng, W. Chakraborty, A. Kaul, W. Shim, M. S Bakir, S. Datta, S. Yu, "Benchmarking Monolithic 3D Integration for Compute-in-Memory Accelerators: Overcoming ADC Bottlenecks and Maintaining Scalability to 7nm or Beyond," in *IEEE International Electron Devices Meeting (IEDM)*, 2020.

[46] H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, M.-J. Tsai, "Metal–oxide RRAM," *Proceedings of the IEEE*, vol. 100, no. 6, p. 1951–1970, 2012.

[47] R. Fackenthal, M. Kitagawa, W. Otsuka, K. Prall, D. Mills, K. Tsutsui, J. Javanifard, K. Tedrow, T. Tsushima, Y. Shibahara, G. Hush, "A 16Gb ReRAM with 200MB/s write and 1GB/s read in 27nm technology," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2014.

[48] T.-Y. Liu, T. H. Yan, R. Scheuerlein, Y. Chen, J. K. Lee, G. Balakrishnan, G. Yee, H. Zhang, A. Yap, J. Ouyang, T. Sasaki, S. Addepalli, A. Al-Shamma, C.-Y. Chen, M. Gupta, G. Hilton, S. Joshi, A. Kathuria, V. Lai, D. Masiwal, M. Matsumoto, et al. , "A 130.7mm2 2-layer 32Gb ReRAM memory device in 24nm technology," in *IEEE International Solid-State Circuits Conference*, 2013.

[49] C.C. Chou, Z.J. Lin, C.A. Lai, C.I. Su, P.L. Tseng, W.C. Chen, W.C. Tsai, W.T. Chu, T.C. Ong, H. Chuang, Y.D. Chih, T.Y. J. Chang, "A 22nm 96KX144 RRAM macro with a self-tracking reference and a low ripple charge pump to achieve a configurable read window and a wide operating voltage range," in *IEEE Symposium on VLSI Circuits*, 2020.

[50] P. Jain, U. Arslan, M. Sekhar, B.C. Lin, L. Wei, T. Sahu, J. Alzate-vinasco, A. Vangapaty, M. Meterelliyoz, N. Strutt, A.B. Chen, "A 3.6 Mb 10.1 Mb/mm 2 Embedded Non-Volatile ReRAM Macro in 22nm FinFET Technology with Adaptive Forming/Set/Reset Schemes Yielding Down to 0.5 V with Sensing Time of 5ns at 0.7 V," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2019.

[51] W. Li,, X. Sun, H. Jiang, S. Huang, S. Yu, "A 40nm RRAM Compute-in-Memory Macro Featuring On-Chip Write-Verify and Offset-Cancelling ADC References," in *IEEE European Solid State Circuits Conference (ESSCIRC)*, 2021.

[52] J. Yang, Y. Kong, Z. Wang, Y. Liu, B. Wang, S. Yin, L. Shi, "Sandwich-RAM: An Energy-Efficient In-Memory BWN Architecture with Pulse-Width Modulation," in *IEEE International Solid- State Circuits Conference (ISSCC)*, 2019.

[53] R. Khaddam-Aljameh, M. Stanisavljevic, J.F. Mas,, G. Karunaratne, M. Brändli, F. Liu, A. Singh, S.M. Müller, U. Egger, A. Petropoulos, T. Antonakopoulos, "HERMES-core—a 1.59-TOPS/mm 2 PCM on 14-nm CMOS in-memory compute core using 300-ps/LSB linearized CCO-based ADCs," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 4, pp. 1027-1038, 2022.

[54] Z. Zhu, H. Sun, Y. Lin, G. Dai, L. Xia, S. Han, Y. Wang, H. Yang,, "A configurable multi-precision CNN computing framework based on single bit RRAM," in *ACM/IEEE Design Automation Conference (DAC)*, 2019.

[55] J. Yue, X. Feng, Y. He, Y. Huang, Y. Wang, Z. Yuan, M. Zhan, J. Liu, J.W. Su, Y.L. Chung, P.C. Wu, "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 International Solid-State Circuits Conference (ISSCC)*, 2021.

[56] H. Jia, M. Ozata, Y. Tang, H. Valavi, R. Pathak, J. Lee, N. Verma, "a programmable neural-network inference accelerator based on scalable in-memory computing," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2021.

[57] G. Huang, S. Liu, L. Van der Maaten, K.Q. Weinberger, "Condensenet: An efficient densenet using learned group convolutions," in *IEEE conference on computer vision and pattern recognition*, 2017.

[58] K. Bong, S. Choi, C. Kim, S. Kang, Y. Kim and H.J. Yoo, "A 0.62 mW ultra-low-power convolutional-neural-network face-recognition processor and a CIS integrated with always-on haar-like face detector.," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2017.

[59] M. Gao, J. Pu, X. Yang, M. Horowitz, C. Kozyrakis, "TETRIS: Scalable and efficient neural network acceleration with 3D memory,," in *ACM International Conference on Architec-tural Support for Programming Languages and Operating Sys-tems (ASPLOS)*, 2017.

[60] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio, "Binarized Neural Networks," in *30th Conference on Neural Information Processing Systems (NIPS)*, 2016.

[61] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," in *arXiv preprint arXiv:1606.06160*, 2016.

[62] R. Banner, I. Hubara, E. Hoffer, D. Soudry, "Scalable methods for 8-bit training of neural networks," in *Advances in neural information processing systems (NIPS)*, 2018.

[63] Y. Yang, L. Deng, S. Wu, T. Yan, Y. Xie, G. Li, "Training high-performance and large-scale deep neural networks with full 8-bit integers," *Neural Networks*, vol. 125, pp. 70-82, 2020.

PUBLICATIONS

1. **H. Jiang**, S. Huang, W. Li, S. Yu, "ENNA: An Efficient Neural Network Accelerator Design Based on ADC-Free Compute-In-Memory Subarrays," **IEEE Transactions on Circuits and Systems I: Regular Papers**, early access, 2022.
2. **H. Jiang**, W. Li, S. Huang, S. Yu, "A 40nm analog-input ADC-free compute-in-memory RRAM macro with pulse-width modulation between sub-arrays," **IEEE Symposium on VLSI Technology and Circuits (VLSI)**, 2022.
3. **H. Jiang**, W. Li, S. Huang, S. Cosemans, F. Catthoor, S. Yu, "Analog-to-digital converter design exploration for compute-in-memory accelerators," **IEEE Design & Test**, vol. 39, no. 2, pp, 48-55, 2022.
4. **H. Jiang**, S. Huang, X. Peng, J.-W. Su, Y.-C. Chou, W.-H. Huang, T.-W. Liu, R. Liu, M.-F. Chang, S. Yu, "A two-way SRAM array based accelerator for deep neural network on-chip training," **ACM/IEEE Design Automation Conference (DAC)**, 2020.
5. **H. Jiang**, X. Peng, S. Huang, S. Yu, "CIMAT: A compute-in-memory architecture for on-chip training based on transpose SRAM arrays," **IEEE Transactions on Computers,** vol. 69, no. 7, pp. 944-954, 2020.
6. **H. Jiang**, X. Peng, S. Huang, S. Yu, "MINT: Mixed-precision RRAM-based in-memory training architecture," **IEEE International Symposium on Circuits and Systems (ISCAS), 2020**.
7. **H. Jiang**, R. Liu, S. Yu, "8T XNOR-SRAM based parallel compute-in-memory for deep neural network accelerator," **IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)**, 2020.

8. **H. Jiang**, X. Peng, S. Huang, S. Yu, "CIMAT: A transpose SRAM-based compute-in-memory architecture for deep neural network on-chip training," **ACM/IEEE International Symposium on Memory Systems (MEMSYS)**, 2019.
9. J.-W. Su, X. Si, Y.-C. Chou, T.-W. Chang, W.-H. Huang, Y.-N. Tu, R. Liu, P.-J. Lu, T.-W. Liu, J.-H. Wang, Y.-L. Chung, J.-S. Ren, **H. Jiang**, S. Huang, S.-H. Li, S.-S. Sheu, C.-I. Wu, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, S. Yu, M.-F. Chang, "Two-way transpose multibit 6T SRAM computing-in-memory macro for inference-training AI edge chips," **IEEE Journal of Solid-State Circuits**, vol. 57, no. 2, pp. 609-624, 2022.
10. S. Huang, X. Sun, X. Peng, **H. Jiang**, S. Yu, "Achieving high in-situ training accuracy and energy efficiency with analog non-volatile synaptic devices," **ACM Transactions on Design Automation of Electronic Systems**, vol. 27, no. 4, p. 37, 2022.
11. W. Li, J. Read, **H. Jiang**, S. Yu, "A 40nm RRAM compute-in-memory macro with parallelism-preserving ECC for iso-accuracy voltage scaling," **IEEE European Solid-State Circuits Conference (ESSCIRC)**, 2022.
12. S. Yu, **H. Jiang**, S. Huang, X. Peng, A. Lu, "Compute-in-memory chips for deep learning: recent trends and prospects," **IEEE Circuits and Systems Magazine**, vol. 21, no. 3, pp. 31-56, 2021.
13. X. Peng, S. Huang, **H. Jiang**, A. Lu, S. Yu, "DNN+NeuroSim V2.0: An end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training," **IEEE Trans. CAD**, vol. 40, no. 11, pp. 2306-2319, 2021.

14. A. Lu, X. Peng, W. Li, **H. Jiang**, S. Yu, "NeuroSim simulator for compute-in-memory hardware accelerator: validation and benchmark," **Frontiers in Artificial Intelligence**, vol. 4, 659060, 2021.
15. S. Huang, **H. Jiang**, X. Peng, W. Li, S. Yu, "Secure XOR-CIM engine: Compute-in-memory SRAM architecture with embedded XOR encryption," **IEEE Trans. VLSI Systems**, vol. 29, no. 12, pp. 2027-2039, 2021.
16. W. Li, X. Sun, **H. Jiang**, S. Huang, S. Yu, "A 40nm RRAM compute-in-memory macro featuring on-chip write-verify and offset-cancelling ADC references," **IEEE European Solid-State Circuits Conference (ESSCIRC)**, 2021.
17. W. Li, S. Huang, X. Sun, **H. Jiang**, S. Yu, "Secure-RRAM: A 40nm 16kb compute-in-memory macro with reconfigurability, sparsity control, and embedded security**," IEEE Custom Integrated Circuits Conference (CICC)**, 2021.
18. S. Huang, X. Peng, **H. Jiang**, Y. Luo, S. Yu, "Exploiting process variations to protect machine learning inference engine from chip cloning," **IEEE International Symposium on Circuits and Systems (ISCAS)**, 2021.
19. A. Lu, X. Peng, W. Li, **H. Jiang**, S. Yu, "NeuroSim validation with 40nm RRAM compute-in-memory macro," **IEEE International Conference on Artificial Intelligence Circuits & Systems (AICAS)**, 2021.
20. S. Huang, **H. Jiang**, S. Yu, "Mitigating adversarial attack for compute-in-memory accelerator utilizing on-chip finetune," **IEEE Non-Volatile Memory Systems and Applications Symposium (NVMSA)**, 2021.
21. J-W. Su, X. Si, Y-C. Chou, T-W. Chang, W-H. Huang, Y-N. Tu, R. Liu, P-J. Lu, T-W. Liu, J-H. Wang, Z. Zhang, **H. Jiang**, S. Huang, S. Yu, K-T. Tang, C-C. Hsieh, R-S.

Liu, S-H. Li, S-S. Sheu, H-Y. Lee, S-C. Chang, M-F. Chang, "A 28nm 64Kb inference-training two-way transpose multibit 6T SRAM computing-in-memory macro for AI edge chips," **IEEE International Solid-State Circuits Conference (ISSCC)**, 2020.

22. S. Huang, X. Sun, X. Peng, **H. Jiang**, S. Yu, "Overcoming challenges for achieving high in-situ training accuracy with emerging memories," IEEE/ACM Design, Automation & Test in Europe **(DATE) 2020**.
23. S. Huang, **H. Jiang**, X. Peng, W. Li, S. Yu, "XOR-CIM: Compute-in-memory SRAM architecture with embedded XOR encryption," **IEEE/ACM International Conference on Computer-Aided Design (ICCAD)**, 2020.
24. W. Shim, **H. Jiang**, X. Peng, S. Yu, "Architectural design of 3D NAND Flash based compute-in-memory for inference engine," **ACM/IEEE International Symposium on Memory Systems (MEMSYS)**, 2020.

VITA

Hongwu Jiang was born in Dalian, China, 1989. He received the B.S. degree in Automation from Dalian University of Technology in 2012 and the M.S. degree in Electrical Engineering from Arizona State University in 2014. In summer 2021, he held an internship at IMEC, U.S.A. In fall 2022, he received the Ph.D. degree in Electrical and Computer Engineering at the Georgia Institute of Technology. His research interests include SRAM-/eNVM-based hardware architecture and accelerator design of deep learning. He received the best paper nomination in ACM/IEEE Design Automation Conference 2020.