ENERGY EFFICIENT COMPUTING-IN-MEMORY ACCELERATORS FOR DEEP LEARNING

LIU DINGBANG

PhD

The Hong Kong Polytechnic University

2025

The Hong Kong Polytechnic University

Department of Computing

Energy Efficient Computing-In-Memory Accelerators For Deep Learning

LIU Dingbang

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Dec 2024

CERTIFICATE OF ORIGINALITY

Abstract

The exponential data growth in the information era, necessitates a re-evaluation of traditional hardware architectures for data-oriented computing. This is particularly critical in the context of emerging deep learning applications, especially in resource-limited edge systems (1; 2). The limitations of pure software-based platforms in handling such vast datasets have become increasingly apparent. To address this challenge, we require scalable hardware solutions capable of both efficient data storage and processing. Overcoming the memory bottleneck is a key hurdle in designing energy-efficient architectures that can support future big data-driven applications.

To meet edge-computing demands, optimizing algorithms and designing specialized circuit systems cooperatively for hardware acceleration is essential. Techniques like precision quantization, data sparsity processing, and network fine-tuning reduce computational load and algorithm complexity. For instance, (3) leverages network redundancy to quantize neural networks to 8-bit integers, improving power consumption and performance. However, uniform quantization not only fails to eliminate all redundancy but can also degrade output accuracy. Method in (4) uses mixed-precision quantization to maintain Transformer performance with lower computational cost, its dependence on a static architecture limits adaptability in dynamic environments. Although (5) proposes a layer-wise fine-grained pruning technique for model sparsification, their approach is limited because it does not incorporate cross-layer sensitivity into the optimization. Consequently, it fails to fully leverage the potential of highly efficient low-bit operations (e.g., 2-bit or 1-bit).

Computing-In-memory (CIM) accelerators have the characteristics of storage and computing integration, which has the potential to break through the limit of Moore's law and the bottleneck of Von Neumann architecture. CIM architectures can be

categorized into current-based, charge-based, time-based, and digital-domain implementations. Current-based CIM architectures perform multiply-accumulate (MAC) operations by modulating bit-line currents in accordance with Kirchhoff's law. While such architectures achieve high density, they are susceptible to non-idealities such as process, voltage, and temperature (PVT) variations and read disturbances. Previous current-based CIM designs have mitigated device nonlinearity and PVT effects through additional peripheral circuitry and limited parallelism (6–8). Charge-domain CIM architectures employ charge accumulation schemes to improve error tolerance and linearity compared to conventional analog approaches (9–11). However, these architectures remain constrained by parasitic effects, which can compromise overall computational performance (12–14). Time-based CIM architectures utilize programmable delay units to perform computations and accumulate results through delay lines (15). A key advantage of this approach is its superior sensing margin compared to current- or charge-based alternatives, though this comes at the cost of increased latency due to sequential processing. Digital CIM architectures perform multiplication and dimension reduction directly using digital logic (16). Despite their robustness, the significant overhead associated with full-precision digital circuits currently limits their energy efficiency.

Static random access memory (SRAM), a conventional memory technology, is renowned for its high speed, high precision, and immunity to noise, making it suitable for accuracy-oriented applications. Resistive random access memory (ReRAM) has also emerged as a promising candidate for power-oriented applications. Its non-volatile nature minimizes leakage power, while it provides high-density storage capabilities. This integration of logic and memory functions within a single device offers significant power and area advantages. Moreover, both memory devices could perform in-memory computations without relying on external I/O operations makes it a potential universal memory solution for future big data applications.

In the first work a fabricated SRAM based charge-domain CIM macro is presented. This work firstly employs neural network search (NAS) method to find out the layer-wise optimized precisions and sparsities for convolutional neural networks (CNNs). Then, a 144-Kb charge-domain signed mixed-precision (2/4/8-bit) CIM accelerator employing bootstrapped SRAM cells with 9-transistors and 1-capacitor (9T1C)

structure is proposed that incorporates a bit-level sparsity-aware analog-to-digital converter (ADC). This work not only achieves highly linear parallel accumulation operations to meet AI computing demands but also implements a hardware and software co-optimization system tailored to specific data characteristics.The design is verified on NAS-optimized networks VGG-16 and ResNet-18 using CIFAR-10 dataset, which could achieve an equivalent accuracy at 4-bit of 68.68% while maintaining a high energy efficiency at 2-bit of 135.19TOPS/W by measurements.

The second work presents a comprehensive hardware-algorithm co-design solution: First, we develop a novel ternary weight splitting (TWS) binarization technique that enables Brain-Floating-Point-16xINT1 (BF16×1-b) transformers, achieving competitive accuracy while dramatically reducing model size compared to full-precision counterparts. Second, we design a fully digital SRAM-based CIM accelerator that integrates bit-parallel SRAM macros within an efficient group vector systolic architecture, capable of storing one complete BERT-Tiny column with stationary systolic data reuse. Implemented in 28nm technology, our design requires only 2KB SRAM within 2mm^2 area while delivering 6.55TOPS throughput at 419.74mW power consumption, achieving state-of-the-art area efficiency of 3.3TOPS/mm^2 and normalized energy efficiency of 20.98TOPS/W for BERT-Tiny model acceleration.

In the last work, an ReRAM based CNN accelerator is designed. Mixed-bit operations from 1 bit to 8 bits are supported by an effective bitwidth configuration scheme to implement Neural Architecture Search (NAS)-optimized layer-wise multi-bit CNNs. Besides, column-parallel readout is achieved with excellent energy-efficient performance by a variation-reduction accumulation mechanism and low-power readout circuits. Additionally, we further explore systolic data reuse in an ReRAM-based PE array. Experiments are implemented on NAS-optimized ResNet-18. Benchmarks show that the proposed ReRAM accelerator can achieve peak energy efficiency of 2490.32 TOPS/W for 1-bit operation and average energy efficiency of 479.37 TOPS/W for 1∼8-bit operations with evaluating NAS-optimized multi-bitwidth CNNs. When compared with the state-of-the-art works, the proposed accelerator shows at least 14.18× improvement on energy efficiency.

In summary, the proposed work is dedicated to the highly efficient computing of deep learning applications. We deploy hardware-software co-optimization methodologies to alleviate computational complexity while maintaining acceptable performance. Specifically, Neural Architecture Search (NAS) is used to automatically design layer-wise mixed-precision networks, enhancing efficiency and minimizing quantization error. A pruning-based methodology is also deployed to introduce sparsity, further reducing computational complexity and creating opportunities for sparse-aware data processing. Additionally, we explore more extreme binarization of Transformer models to boost computational efficiency while maintaining accuracy.

These neural network optimizations create additional requirements for circuit design. All three of our proposed accelerators are tailored at the macro and architectural levels to accommodate these pre-optimized networks with high energy efficiency. Finally, the proposed work addresses prominent issues in charge-based and digital Computing-In-Memory (CIM) architectures, specifically concerning signal integrity, PVT effects, parallelism, and computing accuracy.

Publications Arising from the Thesis

1. Dingbang Liu, et al. "An Energy-Efficient Mixed-Bit CNN Accelerator with Column Parallel Readout for ReRAM-based In-memory Computing.", *IEEE Journal of Emerging and Selected Topics in Circuits and Systems*, vol. 12, no. 4, pp. 821-834, Dec. 2022.

2. Dingbang Liu, et al. "Low-power computing with neuromorphic engineering", *Wiley-VCH: Advanced Intelligence System*, Vol. 2, no.3, 2000150, Sept 2020.

3. Dingbang Liu, et al. "An Energy-Efficient Mixed-Bit ReRAM-based Computing-in-Memory CNN Accelerator with Fully Parallel Readout.", in *2022 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 1-5. 2022.

1. Dingbang Liu, et al. "A Highly Energy-Efficient Binary BERT Model on Group Vector Systolic CIM Accelerator.", in *2025 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, Iceland, 2025.

4. Wei Mao, Dingbang Liu, et al, "A 28-nm 135.19 TOPS/W Bootstrapped-SRAM Compute-in-Memory Accelerator with Layer-wise Precision and Sparsity", *IEEE Transactions on Circuits and Systems I: Regular Papers.* pp. 1-11. 2024.

5. Haoxiang Zhou, Haiqiao Hong, Dingbang Liu, et al, "RISC-V based Fully-Parallel SRAM Computing-in-Memory Accelerator with High Hardware Utilization and Data Reuse Rate", in *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS).* pp. 1-5. 2023.

6. Wei Mao and Zhihua Xiao and Peng Xu and Hongwei Ren, Dingbang Liu, et al, "Energy-Efficient Machine Learning Accelerator for Binary Neural Networks", in *Proceedings of the 2020 on Great Lakes Symposium on VLSI (GLSVLSI)*. pp. 77–82. 2020.

Acknowledgments

Firstly, I would like to express my sincere gratitude to my supervisors, Prof. YU Hao from the School of Microelectronics at Southern University of Science and Technology and Prof. CHEN Changwen and Prof. LYU Mingsong from the Department of Computing at Hong Kong Polytechnic University, for providing me with this invaluable opportunity to pursue my Ph.D. degree. I am deeply grateful for their unwavering support and guidance throughout my academic journey. Their profound knowledge has inspired countless ideas, their scholarly insights have significantly improved my research manuscripts, and their generous support for chip tape-outs has enabled me to achieve a significant milestone as a Ph.D. student. They have instilled in me the importance of setting high standards and cultivating a global perspective. They serve as exemplary researchers, demonstrating unwavering enthusiasm, innovative thinking, and a dedicated work ethic. I am particularly grateful to my co-supervisor Prof. LV Mingsong for his mental and financial support during the challenging pandemic period.

Secondly, I would like to express my sincere gratitude to my former supervisors, Prof. GUAN Nan for accepting me as a Ph.D. candidate. Although unforeseen circumstances prevented me from completing my Ph.D. under his guidance, his early mentorship and support have had a profound impact on my academic journey. I also would like to thank the faculty members at SUSTECH and PolyU, including Prof. MAO Wei, Prof. HUANG Mingqiang, Prof. AN Fengwei, Prof. LIN Longyang and Prof. CHAI Yang for their invaluable guidance and support. Their contributions were essential to my completion of my studies. I am also deeply grateful to my exceptional group members at SUSTECH: ZHOU Haoxiang, LIU Jun, LI Kai , GUAN Ziyi, CHEN Qinglong, YANG Jiaqi, DONG Jingyao, CHENG Quan, XIA

Yu, YANG Shuxin, and LIU Hang. I also extend my thanks to the outstanding members of the PolyU group, including HE Qingqiang, YU Bruce and YU Xiaotong, and many others who accompanied me during these memorable years.

Finally, I would like to express my deepest gratitude to my family, including my parents, grandmother, wife, and other relatives, for their unwavering support. Their encouragement has been instrumental in allowing me to focus on my studies. I am incredibly fortunate to have them in my life.

Table of Contents

Abstract **i**

Publications Arising from the Thesis **v**

Acknowledgments **vii**

List of Figures **xiii**

List of Tables **xviii**

1 Introduction **1**

1.1 Motivation 1

1.2 Volatile/Non-Volatile Memory for CIM 5

1.2.1 Volatile Memory 6

1.2.2 Emerging Non-volatile Memory 8

1.3 Analog and Digital-Domain CIM 9

1.4 Precision Reconfiguration in CIM 15

1.5 Challenges and Contribution 18

1.6 Thesis Organization 22

2 Neural Network Optimization **23**

2.1 Precision and Sparsity Co-optimization 23

2.1.1 Bit-level Sparse Mixed-Precision Network 23

2.1.2 Stage I: Differentiable Neural Architecture Search 24

2.1.3 Stage II: Bit-level Sparsity Quantization 26

2.2 Binarization of BERT 28

2.2.1 Ternary Weight Splitting 28

2.2.2 Quantization with Knowledge Distillation 30

2.2.3 Activation Quantization 31

2.2.4 Algorithm Flow 32

2.3 NAS-optimized Mixed-bit CNNs 34

3 Analog-domain SRAM-based CIM Accelerator 37

3.1 Introduction 37

3.2 Related Work 39

3.2.1 CNN NAS Optimization Challenges for Edged AI 39

3.2.2 CIM Architecture Challenges for Parallel Computing 40

3.2.3 Sparsity Data Processing Challenges on CIM structure 41

3.3 CIM Accelerator Architecture 41

3.3.1 Charge-domain parallel CIM array 41

3.3.2 Bootstrap CIM SRAM Cell 42

3.4 Sparsity-aware Column-parallel Readout 46

3.4.1 Bit-level Sparsity-aware ADC 46

3.4.2 Signed-mixed-precision MAC 49

3.4.3 Data Mapping and Reuse Method 50

4 Digital-domain SRAM-based CIM Accelerator 52

4.1 Introduction 52

4.2 Related Work 54

4.2.1 Transformer Binarization Challenges for Edge AI 54

4.2.2 Parallel Computing Challenges in Digital CIM Architecture . 55

4.2.3 Attention Computing Challenges on CIM Structure 56

4.3 BF16x1-b CIM PE 57

4.3.1 Bit-parallel CIM Macro Design 57

4.3.2 Batch-wise Mantissa Shifter 60

4.4 Group-Vector-Systolic CIM Array 64

4.5 Data Frame Standardization 66

5 ReRAM-based CIM Accelerator 68

5.1 Introduction 68

5.2 Related Work 71

5.2.1 NAS-Optimized Mixed-Bit CNNs 71

5.2.2 ReRAM CIM Accelerator Readout 71

5.3 Mixed-bit CIM Accelerator 73

5.3.1 Mixed-bit MAC 73

5.3.2 Mixed-bit Operation 76

5.4 Column Parallel Readout 78

5.4.1 Column Parallel Structure 78

5.4.2 Low-Power Cycle-reduced ADC By Weight Preprocessing . . . 81

5.5 Systolic Dataflow 83

6 Experiment Results 86

6.1 Results on Analog-domain SRAM-based CIM Accelerator 86

6.1.1 NAS-optimized Network 87

6.1.2 CIM Array with Bootstrapped SRAM Cell 88

6.1.3 Sparsity-aware ADC 89

6.1.4 System Evaluation 91

6.1.5 Conclusion 93

6.2 Results on Digital-domain SRAM-based CIM Accelerator 94

6.2.1 Experiment Setup 94

6.2.2 System Performance Evaluation 95

6.2.3 Conclusion 101

6.3 Results on Analog-domain ReRAM-based CIM Accelerator 102

6.3.1 Simulation Settings 102

6.3.2 Performance Summary 103

6.3.3 Conclusion 109

7 Conclusion and Future Work **110**

7.1 Conclusion 110

7.2 Future Work 114

References **118**

List of Figures

1.1 (a). Power statistics of MAC operations. (b). Power breakdown of single accumulation in 45nm technology of 0.9V (19). 2

1.2 Data transaction in conventional Von Neumann architecture of data x and intermediate result $f(x)$ results with substantial latency and energy increment. 3

1.3 Evolution of Parameter Size and Memory Capacity in Deep Neural Networks and Deep Learning Accelerators (23). 4

1.4 In CIM architecture, the data and results transaction are eliminated and therefore decrease latency and energy. Both volatile memories like SRAM and DRAM, and non-volatile memories like ReRAM, PCM and SST-MRAM can be deployed as a core computing element in CIM architecture (30). 6

1.5 Power breakdown of analog-domain and digital-domain CIM architecture (32). 9

1.6 Tour mainstream CIM architectures (33). (a). Analog-domain current-based CIM architecture. (b). Analog-domain charge-based CIM architecture. (c). Analog-domain time-based CIM architecture. (d). Digital-domain CIM 10

1.7 CIM macro designs for current-based CIM architecture. (a). ReRAM-based 2T2R CIM macro design (6). (b). SRAM-based Twin 8T CIM macro design (7). (c). DRAM-based 3T1C CIM macro design (8). . . 11

1.8 CIM macro designs for charge-based CIM architecture. (a). SRAM-based 8T1C CIM macro design (34). (b). SRAM-based 8T2C CIM macro design (35). (c). SRAM-based 9T CIM macro design with C2C ladder (13). 13

1.9 Digital-domain CIM architectures. (a). 1T1R ReRAM-based digital-domain CIM architecture (36). (b). Full adder integrated customized 6T SRAM-based digital-domain CIM architecture (37). (c). Full parallel customized 6T SRAM-based digital-domain CIM architecture (16). 14

1.10 Multi-bit weight realization approaches (13). (a). WL signal magnitude modulation scheme (40). (b). WL signal pulse width modulation scheme (41). (c). weight mapping scheme. 15

1.11 (a). Bit-serial multi-bit CIM structure. (b). Bit-serial multi-bit data flow. 16

1.12 A multi-bit CIM array with binary weight capacitor ladder (13). . . . 17

1.13 Conventional separated signed weight in CIM array. 17

2.1 Comparison of sparsity mapping schemes in CIM arrays. 24

2.2 Search flow of bit sparse network and mixed-precision CNNs. 25

2.3 Overview of the two-step binarization flow of BF16×1-b BERT model. 29

2.4 (a)DNAS Workflow. (b)NAS-optimized VGG16 Structure. 35

3.1 Proposed CIM system: (a) layer-wise network optimization flow by NAS-based algorithm. (b) column-parallel 9T1C CIM macro structure. 38

3.2 Proposed CIM array with parallel computing. 42

3.3 Schematic of proposed 9T1C bootstrapped CIM macro. 43

3.4 Control peripherals for macro array. (a). array column control circuits; (b). array row control circuits. 43

3.5 Macro "AND" logic operation procedures. 44

3.6 Capacitive accumulation method comparison. 45

3.7 Schematic of proposed sparsity-aware ADC. 47

3.8 ADC control signal generation method in the form of one-hot code. . 47

3.9 Proposed Sparsity-aware SAR logic block design. 48

3.10 Proposed Sparsity-aware logic cell design. 48

3.11 Waveform of ADC control signals in different sparse rates. 49

3.12 Signed-mixed-precision module and the corresponding workflow. . . . 50

3.13 Input feature reuse module. 51

4.1 Proposed CIM accelerator with BF16x1-b CIM PE, group vector systolic CIM PE groups and data frame standardization for efficient inference of binarized BF16x1-b BERTs. 53

4.2 Proposed BF16x1-b CIM PE with 2-to-1 multiplexed adders. 57

4.3 Proposed bit-parallel CIM macro. (a). Schematic; (b). Physical Layout. 58

4.4 Transient time diagram of proposed CIM macro. 59

4.5 Comparisons between XOR units. (a). Typical 2T XOR structure; (b). Typical 4T XOR structure with 2 transmission gates; (c). 4T XOR structure with non-inverted input; (d). Proposed 4T-XOR unit. 59

4.6 Alignment challenge of BF-based MAC operations in conventional CIM architecture. 61

4.7 Comparison between floating-point-based CIM architectures. (a). CIM architecture with 8-b mantissa computing space; (b). Proposed CIM architecture with extra 4-b mantissa shifting space. 62

4.8 Proposed Mantissa shifter design for activation in BF16 format. . . . 63

4.9 Comparison of different systolic array design in CIM.(a). Classical atomic systolic CIM array; (b). Vector systolic CIM arrray; (c). Proposed group vector systolic CIM array. 64

4.10 Register file (Area) and fan-out (latency) analysis on different systolic array. 65

4.11 Proposed Group vector systolic CIM array design. 66

4.12 Neural network mapping of proposed digital CIM accelerator. 67

5.1 Proposed ReRAM-based CIM accelerator for Mixed-bit CNNs. 69

5.2 Mixed-bit MACs for 2-bit, 4-bit and 8-bit operations with corresponding input sequences. 74

5.3 Macro cell in proposed ReRAM-based CIM accelerator: (a) Macro cell schematic and (b) timing waveform. 75

5.4 Mixed-bit operation method of the proposed system: (a) structure of mixed-bit PE and (b) mixed-bit dataflow. 77

5.5 ReRAM-based CIM array comparison with different readout schemes: (a) conventional column-serial readout scheme; (b) proposed column-parallel readout scheme. 79

5.6 Low-power cycle-reduced ADC System: (a) working flow, (b) schematic and (c) waveforms. 82

5.7 System structure with systolic dataflow and weight mapping strategy. 84

6.1 "0"s distribution of ResNet-18 networks: (a) fixed-precision, (b) mixed-precision, (c) bit-level sparse mixed-precision. 87

6.2 Linearity performance of bootstrapped CIM array under Monte-Carlo simulation @100MHz: (a)array analog output voltage curve, (b)INL and DNL. 88

6.3 Linearity performance of ADC under input voltage range variation: (a)dynamic performance, (b)static performance. 89

6.4 ADC power consumption evaluation with different gated bit numbers. 90

6.5 (a) Layer-by-layer energy efficiencies for ResNet-18 and VGG-16 with input bit width 2/4/8. (b) Layer-by-layer energy efficiency improvement for ResNet-18 and VGG-16 with input bit width 2/4/8. 90

6.6 (a) Experiment setup. (b) Die photograph. 91

6.7 Chip power breakdown. 91

6.8 FOM comparison between this work with NAS optimized NN and previous CIM macros (normaized to 28nm and 4-bit configuration). . 92

6.9 Layout photograph and chip summary. 94

6.10 Area and power breakdown of proposed CIM accelerator. (a). Area breakdown. (b). Power breakdown. 95

6.11 Comparison of compression ratio and accuracy for various quantization configurations on the SST-2 Dataset with (126). 96

6.12 Conventional CIM implementation for floating point operation accuracy evaluation. 97

6.13 Proposed CIM accelerator for floating point operation accuracy evaluation 98

6.14 FoM comparison between this work and digital CIM implementations for BERT (126–129) (FoM=Normalized Energy efficiency $\times$ Area efficiency). 100

6.15 Energy efficiency evaluation in mixed-bit ResNet-18. 104

6.16 ResNet-18 inference accuracy with ImageNet datasets. Different configurations for variation sets are examined. 105

6.17 ADC and system power reduction under different sparsity rates. . . . 106

List of Tables

1.1 Memory devices comparison. 7

5.1 NAS-optimized multi-bitwidth CNN benchmarks. 70

5.2 Parameters of proposed ReRAM-based CIM accelerator 73

5.3 The True Table of Proposed Macro Cell 74

6.1 Results of Optimized ResNet-18 on CIFAR-10 Dataset 86

6.2 Measurement Result and Comparison Table 92

6.3 Measurement results and comparison. 99

6.4 CIM Macro Cell Specification Summary 103

6.5 ADC Power Consumption under Cycle-reduced ADC 106

6.6 System Performance Evaluation 108

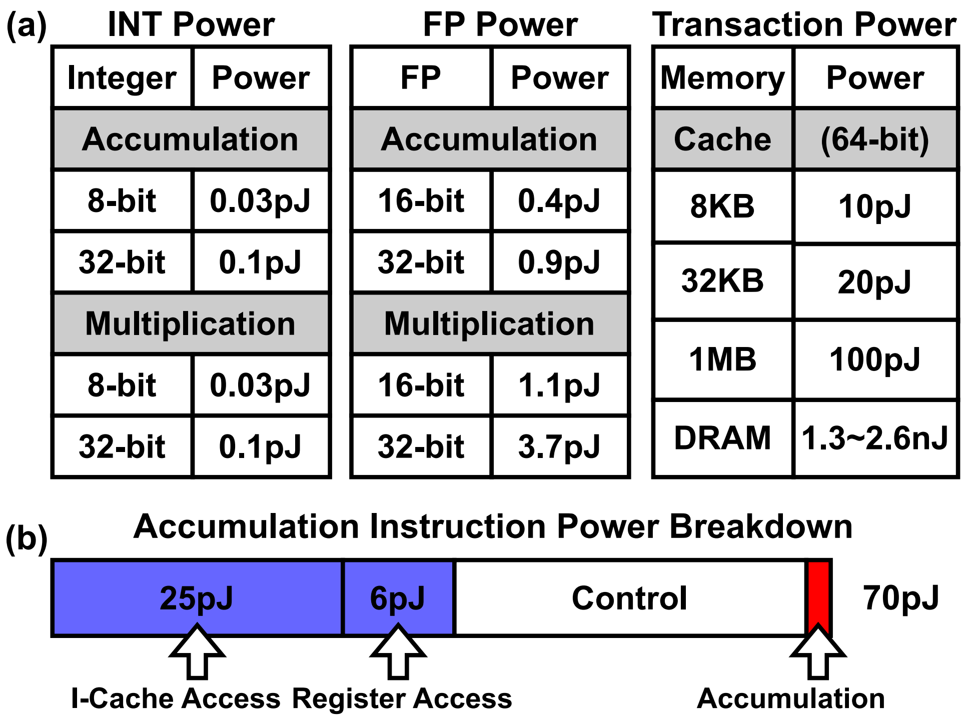
Chapter 1

Introduction

1.1 Motivation

Deep neural networks (DNNs) have achieved remarkable accuracy by extracting high-level representations from massive datasets through complex statistical models. From simple tasks like handwritten digit recognition to sophisticated multimodal chatbots, DNNs have evolved rapidly, leading to a surge in parameter counts. The initial 2.8KB of LeNet (17) has ballooned to over 100 billion parameters in models like GPT (18). While DNNs incorporate various operations such as striding, non-linear comparisons, concatenation, and transposition, the dominant operation remains Multiply-Accumulate (MAC). This explosive growth in parameters has placed significant computational and storage pressure on traditional processors. From the perspective of power, as illustrated in Figure 1.1, data movement within on-chip caches during MAC operations consumes 2-3 orders of magnitude more power than the actual multiplication and accumulation operations. Furthermore, off-chip DRAM data movement incurs an additional 5-6 orders of magnitude higher power consumption. Consequently, modern deep learning applications suffer from significant data movement overhead, impacting overall power consumption.

From a storage perspective, networks with massive parameters can easily saturate bandwidth or pipeline resources, significantly increasing computational latency. Experiments (20) have shown that deploying the GPT-2 (1GB) (21) model on a TITAN



(a)

INT Power

Integer	Power
Accumulation	
8-bit	0.03pJ
32-bit	0.1pJ
Multiplication	
8-bit	0.03pJ
32-bit	0.1pJ

FP Power

FP	Power
Accumulation	
16-bit	0.4pJ
32-bit	0.9pJ
Multiplication	
16-bit	1.1pJ
32-bit	3.7pJ

Transaction Power

Memory	Power
Cache	**(64-bit)**
8KB	10pJ
32KB	20pJ
1MB	100pJ
DRAM	1.3~2.6nJ

(b)
Accumulation Instruction Power Breakdown
25pJ
6pJ
Control
70pJ
I-Cache Access
Register Access
Accumulation

Figure 1.1: (a). Power statistics of MAC operations. (b). Power breakdown of single accumulation in 45nm technology of 0.9V (19).

Xp GPU results in a 370ms latency for processing a 30-token sentence. In contrast, MobileNet-v2 (4.3MB) (22) achieves a latency of only 6ms for image inference on the same hardware. This two-order-of-magnitude latency gap underscores the substantial impact of model size on computational performance. The disparity becomes even more pronounced on bandwidth-constrained edge devices. For example, deploying GPT-2 on a Raspberry Pi leads to a staggering 43-second latency for a single sentence, rendering real-time applications impractical. This highlights the significant challenges associated with running large models on resource-constrained devices and underscores the need for efficient model compression and hardware acceleration techniques.

The Von Neumann architecture, characterized by its strict separation between computation and memory, significantly contributes to the current challenges of power consumption and storage in deep learning. As illustrated in Figure1.2, this architecture requires a processor to fetch data from memory, execute computations, and then write the results back to memory. The limited bandwidth for data exchange, coupled with high energy demands, creates substantial obstacles for compute-intensive

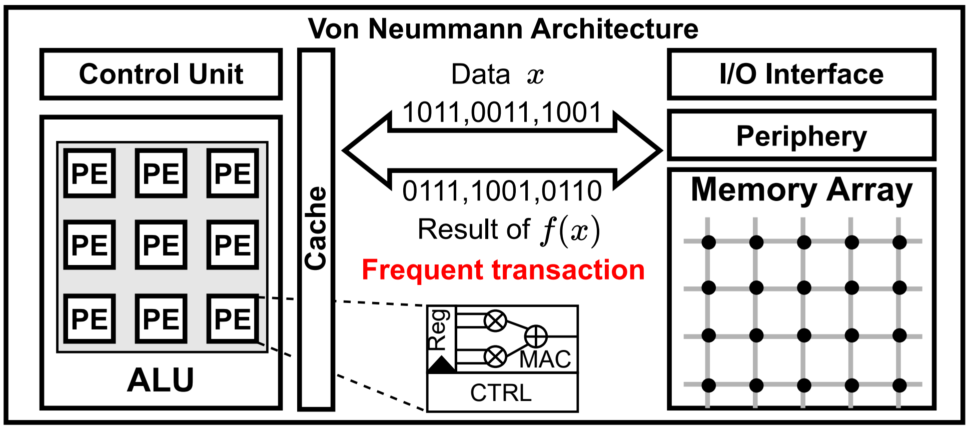


Figure 1.2: Data transaction in conventional Von Neumann architecture of data x and intermediate result $f(x)$ results with substantial latency and energy increment.

tasks. This inherent limitation, commonly referred to as the 'memory wall,' has impeded the progress of AI applications. Most modern computing processors, such as central processing units (CPUs), graphics processing units (GPUs), and various deep learning accelerators, are based on this architecture, leading to significant time and energy expenditure on data transfer and memory access. As depicted in Figure1.3, the memory wall issue intensifies as Moore's Law slows, becoming a critical bottleneck for advancements in deep learning (23).

In response to the aforementioned issues, various algorithmic-level methods, such as network quantization (3) and model pruning (4; 24), have been proposed to optimize neural networks. These techniques aim to reduce the number of parameters and decrease algorithmic complexity. However, many of these neural network structure optimizations rely on manual design and expert intuition to create classic architectures, often resulting in redundant parameters. As the number of parameters grows, manual optimization becomes increasingly challenging.

Neural Architecture Search (NAS) offers a solution by automating the process of network design. By optimizing hyperparameters like data bit width, NAS can search for optimal weight precision layer by layer, ensuring accurate inference while significantly reducing data volume and enabling DNN compression. Furthermore, recent advancements in neural network binarization (25; 26) present untapped potential for accelerator design. To fully harness the benefits of NAS-optimized, and binarized networks, more flexible circuit designs are necessary.

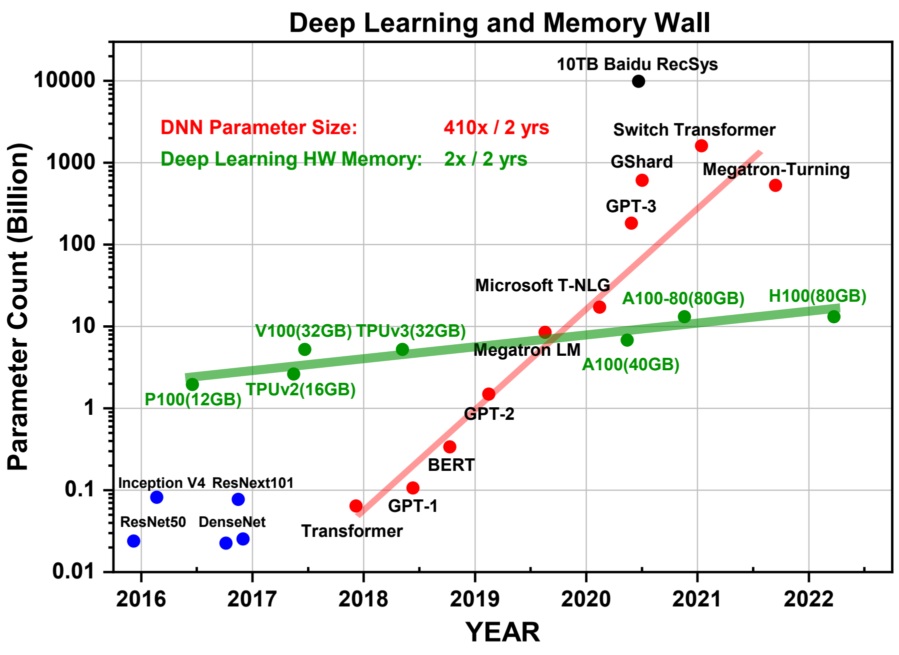


Figure 1.3: Evolution of Parameter Size and Memory Capacity in Deep Neural Networks and Deep Learning Accelerators (23).

To address the performance mismatch between memory and processors at the hardware design level, the Computing In-Memory (CIM) architecture has been proposed. As shown in Figure 1.4, the CIM architecture integrates computing units with storage units, placing some simple logic calculations and MAC operations within the storage units themselves. This enables data to be computed simultaneously as it is read out, minimizing data movement and reducing the power consumption associated with data interaction, thereby achieving higher computational efficiency.

While CIM offers inherent advantages in parallelism and low power consumption, several significant challenges persist.

- Firstly, CIM architectures can be broadly categorized into analog and digital domains, each with distinct advantages and limitations. Analog-domain CIM excels in throughput and area efficiency but is susceptible to device nonlinearity and routing parasitic effects, which can degrade accuracy. Digital-domain CIM, while enabling lossless computing, incurs significant area and power overhead due to technological constraints. To fully realize the potential of CIM, it is essential to explore the boundaries of various CIM architectures and optimization topologies in specific application contexts.

- Secondly, traditional CIM architectures primarily support fixed-precision or multi-precision computing, limiting their ability to execute DNNs with layer-wise mixed-precision networks optimized by NAS. To fully leverage the advantages of NAS-optimized mixed-precision networks, supporting layer-wise mixed-precision on CIM architectures is highly desirable.

- Thirdly, MAC operations remain a significant power consumer in CIM architectures. Prior research often relies on block-level or element-level data sparsity to reduce MAC power consumption. However, these techniques have limited flexibility and are only effective for specific data flow patterns. For more fine-grained sparse data, utilization decreases, leading to a substantial reduction in computational efficiency. Consequently, supporting fine-grained bit-wise sparse data flow in CIM architectures is preferred.

1.2 Volatile/Non-Volatile Memory for CIM

As introduced in previous section, conventional computing architectures suffer from significant latency and energy overhead due to frequent data transfers between off-chip memory and the processor. To mitigate this bottleneck, CIM has emerged as a promising approach. By minimizing the physical distance between processing and storage elements, as depicted in Figure 1.4(b), CIM aims to reduce latency and energy consumption.

The concept of CIM was first introduced in the 1960s (27; 28), but its implementation was hindered by technological limitations. Today, as computing demands intensify, memory devices for CIM must possess high linearity, long retention, low stochastic behavior, and low energy consumption. Emerging Non-Volatile Memory (eNVM) technologies, such as Resistive Random Access Memory (ReRAM), Spin-Transfer-Torque Magnetic Random Access Memory (STT-MRAM), Phase-Change Memory (PCM), Conductive Bridge Random Access Memory (CBRAM), and Optoelectronic Resistive Random Access Memory (ORRAM), offer promising solutions due to their fast switching speeds, low operating voltages, and low energy consump-

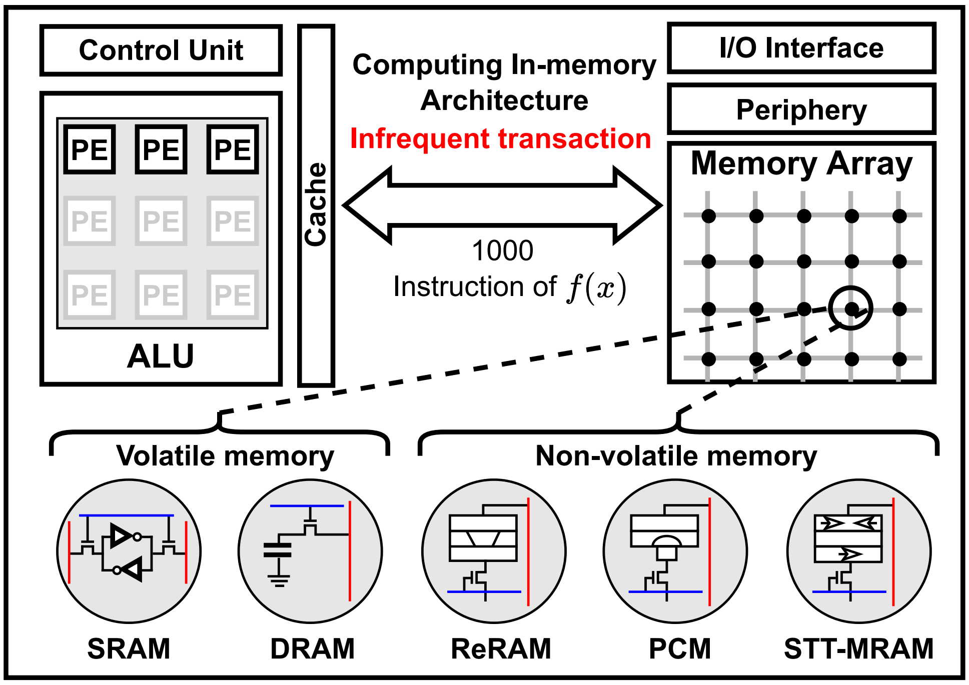


Figure 1.4: In CIM architecture, the data and results transaction are eliminated and therefore decrease latency and energy. Both volatile memories like SRAM and DRAM, and non-volatile memories like ReRAM, PCM and SST-MRAM can be deployed as a core computing element in CIM architecture (30).

tion (29). Additionally, volatile memories like SRAM and DRAM, with their robust performance and long retention, are widely used in CIM architectures. A comparison of these memory devices is summarized in Table 1.1.

In the following subsections, we will delve into the working principles of several key memory technologies.

1.2.1 Volatile Memory

As one of the two primary subcategories of memory devices for CIM, volatile memory primarily utilizes charge as the information storage medium, with SRAM and DRAM being prominent examples.

Static Random Access Memory (SRAM)employs a bistable transistor structure, typically a 6T cell, as depicted in Figure 1.4(b). This structure leverages a dynamic latch, composed of two interconnected inverters, to form a stable positive

Table 1.1: Memory devices comparison.

Device Type	SRAM	DRAM	ReRAM	PCM	STT-MRAM
Area	$>100F^2$	$6F^2$	$4F^2$	$6\sim20F^2$	$6\sim20F^2$
Composition	6T	1T1C	1T1R	1T1R	1T1R
Supply voltage	<1V	<1V	<3V	<3V	<3V
Read latency	$\sim1ns$	$\sim10ns$	$<10ns$	$<10ns$	$\sim10\mu s$
Write latency	$\sim1ns$	$\sim10ns$	$<10ns$	$<50ns$	$<5ns$
Write power	$\sim1pJ$	$\sim10pJ$	$\sim0.1pJ$	$\sim10pJ$	$\sim0.1pJ$
Endurance	$>10^{16}$	$>10^{16}$	$10^6\sim10^{12}$	$>10^9$	$>10^{15}$
Multi-bit capability	N.A.	N.A.	3-bit	8-bit	8-bit
Technology maturity	High	High	Low	Low	Low

feedback loop. The low barrier height of field-effect transistors (FETs) (0.5 eV) necessitates frequent charge replenishment from external sources to maintain data integrity (30). SRAM offers near-unlimited write-read endurance cycles and sub-nanosecond access times, but at the cost of a larger cell area compared to other memory technologies.

Dynamic Random Access Memory (DRAM), similar to SRAM, relies on charge storage to retain information. However, DRAM stores charge in a capacitor's dielectric layer, which possesses a higher and more stable barrier height (1.1 eV) (30) and forms a 1T1C structure. While this allows for longer data retention compared to SRAM, periodic refresh cycles are still required to prevent charge leakage. Owing to technological maturity, DRAM offers nearly unlimited write-read endurance cycles without significant area overhead, albeit at the cost of slower access times (typically <10 ns).

1.2.2 Emerging Non-volatile Memory

Unlike volatile memories, emerging non-volatile memories (eNVMs) store information by altering the atomic arrangement or orientation of ferromagnetic metal layers. Common examples of eNVMs include ReRAM, PCM, and STT-MRAM. To control the switching behavior of eNVMs, a 1T1R structure is often employed, where a transistor is connected in series with the eNVM cell, as illustrated in Figure 1.4(b).

Resistive Random Access Memory (ReRAM) is a sandwich-like structure composed of a metal oxide layer sandwiched between two metal electrodes, as shown in Figure 1.4(b). It operates as a two-terminal device, exhibiting two distinct resistance states: a high resistance state (HRS, or OFF state) and a low resistance state (LRS, or ON state). The transition between these states is controlled by applying specific electrical pulses, which induce the formation or rupture of conductive filaments within the metal oxide layer. This resistive switching mechanism is attributed to the generation and drift of oxygen vacancies (29).

Phase Change Memory (PCM) is a two-layer device that utilizes the phase change properties of a specific material. The transition between the high resistance state (HRS) and low resistance state (LRS) is achieved by switching the material between its amorphous and crystalline phases. To induce the HRS state, a high current pulse is applied to melt the crystalline material into an amorphous state. Subsequently, a lower current pulse is used to recrystallize the material, returning it to the LRS state (29). As depicted in Figure 1.4(b), a typical PCM device features a mushroom-shaped structure, where the bottom electrode confines heat and current, while the hemisphere-shaped top layer represents the phase-change material (30).

Spin Transfer Torque Magnetoresistive Random Access Memory (STT-MRAM) utilizes a magnetic tunnel junction (MTJ) structure comprising three layers: a free layer (FL) for data storage, a pinned layer (PL) for reference, and a tunnel barrier (typically MgO) for magnetic insulation and data readout. The relative orientation of the magnetic moments in the FL and PL determines the device's resistance state. When the magnetic moments are parallel, the MTJ exhibits a low resistance state (LRS). Conversely, an antiparallel alignment results in a high resistance state (HRS). By applying a spin-polarized current, the magnetic moment

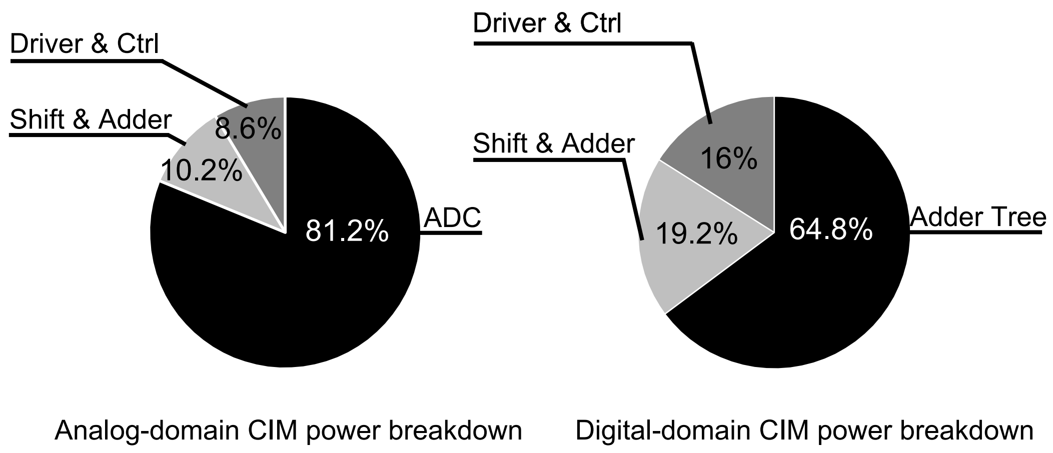


Figure 1.5: Power breakdown of analog-domain and digital-domain CIM architecture (32).

of the FL can be switched, enabling data writing. Additionally, STT-MRAM can be engineered to exhibit multiple resistance states by introducing additional magnetic layers or materials.

1.3 Analog and Digital-Domain CIM

To accomplish in-situ logic operations, corresponding circuitry designs can be categorized into two primary approaches: analog-domain CIM and digital-domain CIM (16; 31; 32).

Analog-domain CIM encompasses several subcategories, including current-based (6), charge-based (11), and time-based CIM (15). These analog-domain architectures integrate custom analog circuits with memory units to achieve higher energy efficiency. While they offer lower power consumption, particularly for applications with lower precision requirements, their computational accuracy is more limited and susceptible to various analog computing errors.

In contrast, digital-domain CIM leverages digital storage units (e.g., SRAM) to achieve high-precision and reliable in-memory computations. This approach is robust against noise and manufacturing variations, offering flexibility for various computational tasks. However, its reliance on digital storage limits its applicability.

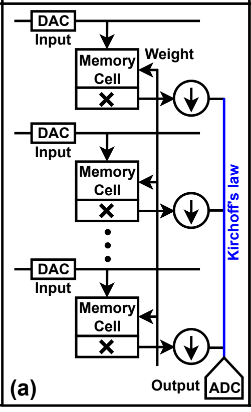
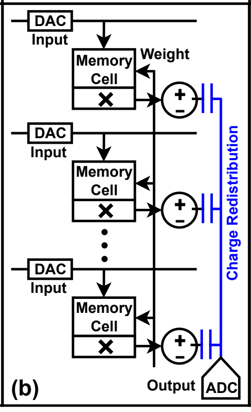
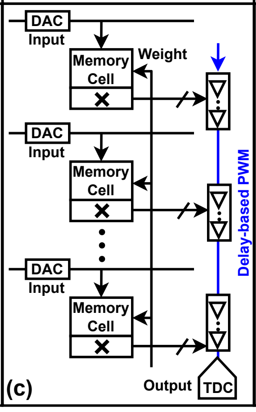
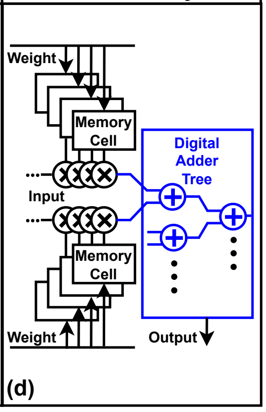
	Analog-domain CIM			Digital-domain CIM
	Current-based	Charge-based	Time-based	
Pros.	High Energy Efficiency High Density	High Energy Efficiency Moderate Accuracy	High Energy Efficiency Moderate Accuracy	High Accuracy
Cons.	Low Accuracy Limited Parallelism	Moderate Density	Limited Parallelism	Moderate Efficiency Low Density
CIM Architectures				

Figure 1.6: Tour mainstream CIM architectures (33). (a). Analog-domain current-based CIM architecture. (b). Analog-domain charge-based CIM architecture. (c). Analog-domain time-based CIM architecture. (d). Digital-domain CIM .

The optimization focus for analog and digital designs varies significantly. As illustrated in Figure 1.5, ADCs in analog in-memory computing can consume up to 81.2% of the total power, while adder trees dominate power consumption in digital in-memory computing, accounting for 62.2%. To enhance the computational efficiency of in-memory computing systems, it is crucial to innovate further in ADCs and adder trees to minimize their power consumption.

Current-based Analog CIM: As an early mainstream CIM implementation, current-based CIM architecture leverages Ohm's law for current modulation and accumulates weighted currents on bitlines (BL) to naturally perform MAC operations, adhering to Kirchhoff's law, as illustrated in Figure 1.6 (a). (6) proposed a 2T2R CIM array based on ReRAM with signed weighting, as depicted in Figure 1.7 (a), where negative weights are subtracted from positive ones, enhancing efficiency by reducing accumulated current and mitigating the impact of IR drop on computational accuracy. The chip achieved a high recognition accuracy of 94.4% and peak energy efficiency of 78.4 TOPS/W on the MNIST dataset.

(7) introduced a novel 8T SRAM cell, as shown in Figure 1.7 (b), by adding two extra transistors to the standard 6T SRAM cell, resolving the read/write interfer-

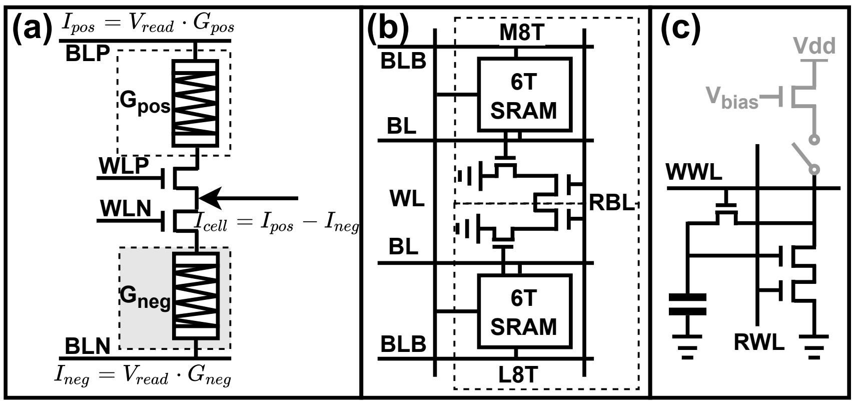

(a)
$I_{pos} = V_{read} \cdot G_{pos}$
BLP
G_{pos}
WLP
WLN
$I_{cell} = I_{pos} - I_{neg}$
G_{neg}
BLN
$I_{neg} = V_{read} \cdot G_{neg}$
(b)
M8T
BLB
6T SRAM
BL
WL
RBL
BL
6T SRAM
BLB
L8T
(c)
Vdd
V_{bias}
WWL
RWL

Figure 1.7: CIM macro designs for current-based CIM architecture. (a). ReRAM-based 2T2R CIM macro design (6). (b). SRAM-based Twin 8T CIM macro design (7). (c). DRAM-based 3T1C CIM macro design (8).

ence problem inherent in traditional current-domain in-memory computing based on 6T SRAM. (8) proposed a novel 3T1C CIM design, as shown in Figure 1.7 (c), innovatively unifying weight programming and computation in the current domain, effectively improving the signal-to-noise ratio (SNR) and reducing errors stemming from transistor size variations and read disturbances.

Charge-based Analog CIM: Current-based CIM relies heavily on the stable performance of Metal-Oxide-Semiconductor Field-Effect Transistors (MOSFETs). However, MOSFETs are susceptible to Process, Voltage, and Temperature (PVT) variations, and the increased discharge currents in parallel architectures can further degrade computational accuracy. In contrast, capacitors offer greater stability. As reported in (34), a 1.33 fF capacitor in the TSMC 28nm process node exhibits a standard deviation of only 0.6%, providing a favorable foundation for high-precision implementation.

As illustrated in Figure 1.6 (b), charge-based CIM leverages capacitor charging and redistribution to perform MAC operations. As shown in Figure 1.8 (a), (34) proposed an 8T1C SRAM structure, augmenting the standard 6T SRAM cell with two extra transistors and a capacitor. The standard 6T cell stores weights of 1 or -1, while the additional transistors implement XNOR logic. This design, with an area overhead of 1.8 times that of a standard 6T SRAM cell, was fabricated in a 65nm process and evaluated using Binary Neural Networks (BNNs). It achieved comparable inference accuracy to software results and an energy efficiency of 866 TOPS/W, with a throughput of 19 TOPS. However, this design is limited to BNNs and cannot readily adapt to complex, large-scale Convolutional Neural Networks (CNNs).

(35) proposed an 8T2C charge-sharing in-memory computing unit based on SRAM. As illustrated in Figure 1.8 (b), in storage mode, word-lines (WL_i and WL_o) are simultaneously triggered to control read and write operations between the internal storage cell and BL. In computing mode, WL_i is first activated to input the activation value, performing an AND operation with the stored weight and charging the switching capacitor. Then, WL_o is activated while WL_i is closed, redistributing the charge between BL and BLB through the left and right switching capacitors, completing a 1-bit $\times$ 1-bit charge-domain computation. This design offers several advantages. Firstly, it effectively avoids the common write disturbance problem in 6T SRAM cells by adopting a dual-switch mechanism, enhancing system robustness. Secondly, the dual-capacitor design reduces the capacity requirement of a single capacitor, leading to lower dynamic power consumption during charging and discharging processes.

(13) proposed a 9T SRAM structure and a C-2C stepped capacitor structure for in-memory computing, as shown in Figure 1.8 (c). Compared to the traditional 6T SRAM, the 9T SRAM cell adds three transistors to improve disturbance tolerance and storage stability, decoupling the read and write operations of the SRAM from compute-in-memory operations. The 1:2 ratio C-2C capacitor stepped structure, where each step branch contains only one unit capacitor, enables multi-bit computation through series-connected capacitors, enhancing computational accuracy and energy efficiency while maintaining low complexity. Test results show that the proposed CIM unit achieved a peak energy efficiency of 32.2 TOPS/W. Using four-layer Multi-Layer Perceptron (MLP) and LeNet-5 CNN networks for MNIST image classification, the hardware implementation achieved accuracies of 98.1% (MLP) and 96.85% (LeNet-5), respectively.

Time-based Analog CIM: Time-based CIM performs calculations by controlling the delay of delay elements and accumulates results through delay lines, as depicted

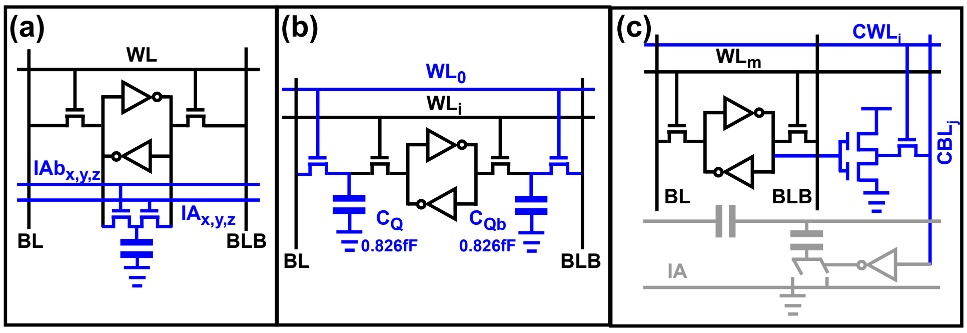


Figure 1.8: CIM macro designs for charge-based CIM architecture. (a). SRAM-based 8T1C CIM macro design (34). (b). SRAM-based 8T2C CIM macro design (35). (c). SRAM-based 9T CIM macro design with C2C ladder (13).

in Figure 1.6 (c). Compared to current-based and charge-based methods, time-based in-memory computing offers a larger noise margin despite the larger latency caused by sequential processing. Additionally, due to the lower flip rate, time-based computation can achieve higher computational efficiency. However, similar to current-based CIM, time-based CIM is susceptible to PVT variations, which can degrade computational accuracy.

(15) proposed a Time-Domain Incremental-Accumulation (TDIA) scheme with a Dynamic Differential-Reference Time-to-Digital Converter (D2REF-TDC). TDIA technology is used for MAC operations, gradually accumulating results in the time domain to ensure high precision and a larger signal margin. The D2REF-TDC converts time-related signals into digital signals using a dynamic reference to improve time-to-digital conversion accuracy. Through co-design optimization, effectively combining software programming and physical hardware structures, read energy consumption is significantly reduced.

Digital-domain CIM: To address the challenges associated with analog-domain CIM, such as PVT variations, energy and area overhead due to data conversion, noise sensitivity, computational accuracy, and scalability, digital-domain CIM has emerged as a promising research area in both academia and industry. Digital-domain CIM, particularly SRAM-based technology, leverages SRAM as the storage unit and integrates it with logic gate units to create CIM modules capable of performing various computational tasks, as depicted in Figure 1.6 (d).

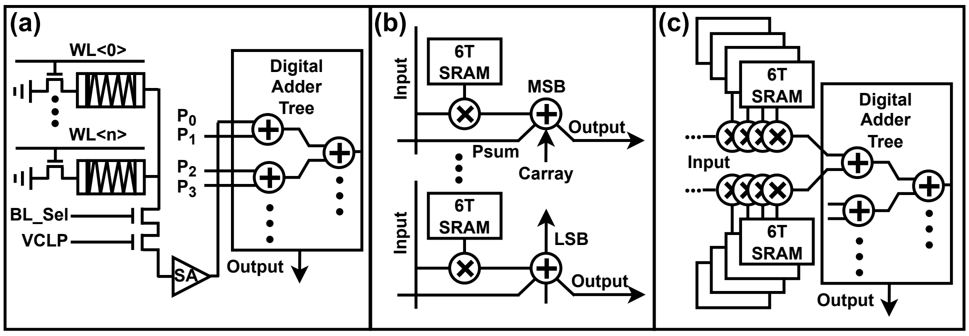


Figure 1.9: Digital-domain CIM architectures. (a). 1T1R ReRAM-based digital-domain CIM architecture (36). (b). Full adder integrated customized 6T SRAM-based digital-domain CIM architecture (37). (c). Full parallel customized 6T SRAM-based digital-domain CIM architecture (16).

(36) proposed a 1T1R ReRAM-based digital-domain CIM accelerator which employs time-multiplexed strategy for row-wise ReRAM output digitization, as shown in Figure 1.8 (a). The design utilize dynamic voltage sense amplifier and OR logic based approximation tree to leverage sparsity in data flow and reduce power consumption. (37) proposed a fully digital unit based on 6T SRAM, as shown in Figure 1.9 (b). This unit employs a fully custom design to support MAC operations in CNNs. By combining XNOR gates and full adders for bit-level MAC operations and utilizing multiplexers for flexible configuration, this design achieves efficient bit-serial MAC computation. Through the combination of different types of units, the precision of weights and output can be flexibly adjusted. This technology supports neural networks with different topologies by using multiple macros in multiple configurations, and effectively reduces memory access energy consumption through the accumulation operation between units, achieving high-efficiency and high-throughput MAC computation.

(16) proposed a 6T SRAM-based fully digital-domain CIM accelerator, providing an innovative computational framework for MAC operations in CNNs, as depicted in Figure 1.9(c). The design supports flexible input activation bit width selection (1-8 bits) and can handle both signed and unsigned data, while allowing for four different weight bit widths, increasing the system's versatility and flexibility. The design also improves energy efficiency by optimizing data flow and reusing weights and activation data. It can support diverse neural network computations by altering

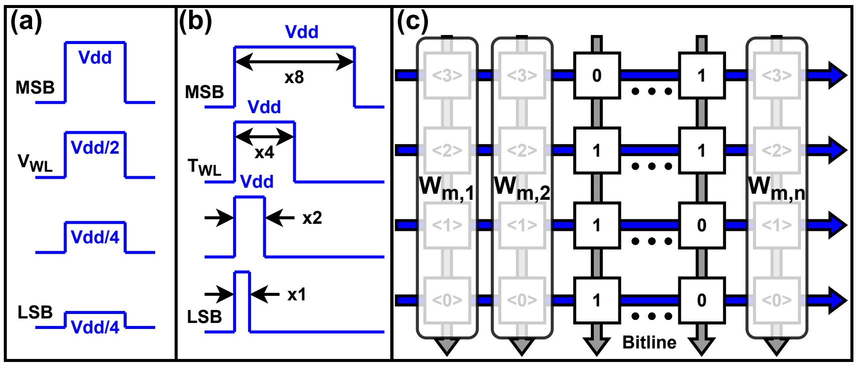


Figure 1.10: Multi-bit weight realization approaches (13). (a). WL signal magnitude modulation scheme (40). (b). WL signal pulse width modulation scheme (41). (c). weight mapping scheme.

the topology.

1.4 Precision Reconfiguration in CIM

Binary neural networks (BNNs) are commonly used in applications where high recognition accuracy is not a strict requirement, such as handwritten digit recognition. While sacrificing some inference accuracy, BNNs can achieve significant network compression. Some CIM implementations can accelerate these networks (35; 38; 39), and compared to accelerating multi-bit neural networks, these designs generally have very high computational efficiency.

While single-bit computation is sufficient for many applications, complex scenarios demand multi-bit data computation. While digital in-memory computing can readily handle multi-bit data, analog in-memory computing faces challenges in converting multi-bit weight data into analog signals. One approach involves adjusting the WL signals associated with vertically aligned multi-bit weight data. Two common methods are WL voltage modulation and WL pulse width modulation.

As shown in Figure. 1.10 (a), the WL magnitude modulation method (40) assigns different voltage levels between VDD and VSS to the WL signals. During computation, all WLs associated with a multi-bit weight are activated simultaneously, with the MSB connected to VDD and subsequent bits assigned progressively lower

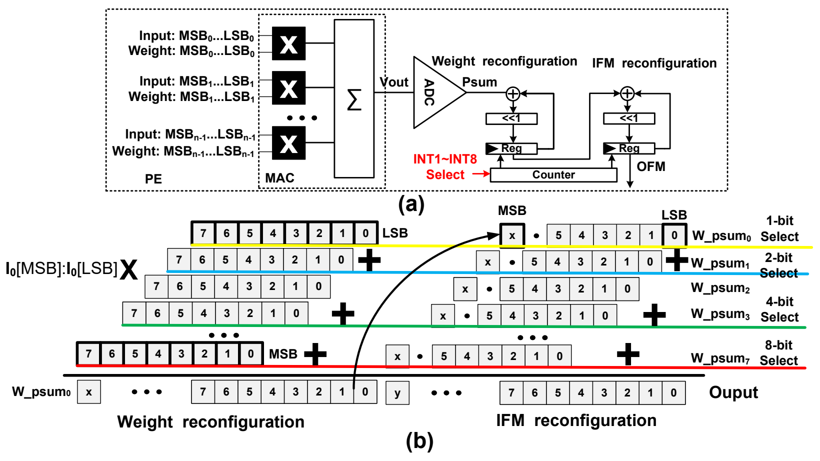


Figure 1.11: (a). Bit-serial multi-bit CIM structure. (b). Bit-serial multi-bit data flow.

voltages. The interaction between the WL signals and binary multi-bit data stored in SRAM cells results in an analog voltage on the additional bitline (BL), representing the binary multi-bit data. However, this method suffers from a non-linear relationship between WL voltage and BL voltage.

WL pulse width modulation, as illustrated in Figure. 1.10 (b), offers a solution to the linearity problem. By modulating the pulse width of the WL signals, a more linear relationship between WL pulse and BL voltage can be achieved (41). However, as precision increases, the pulse width of the MSB grows exponentially, leading to significant computational latency for high-precision computations.

Bit-serial processing is a common approach for handling multi-bit weight data computation (37). As illustrated in Figure 1.11 (b), this method involves multiplying a 1-bit input data with an n-bit weight data within the storage array. The resulting partial products are then shifted and accumulated in the digital domain to obtain the final result. However, this process requires multiple cycles per result, negatively impacting overall throughput performance.

Binary-weighted capacitor ladders are another common approach for multi-bit computation (42; 43). As depicted in Figure 1.12 (13), a k-bit ladder comprises a series of capacitors arranged in a binary-weighted ratio. The weight data is stored in k storage units, with each unit controlling a corresponding branch in the ladder. The ladder's output is the product of the applied weight and the input voltage.

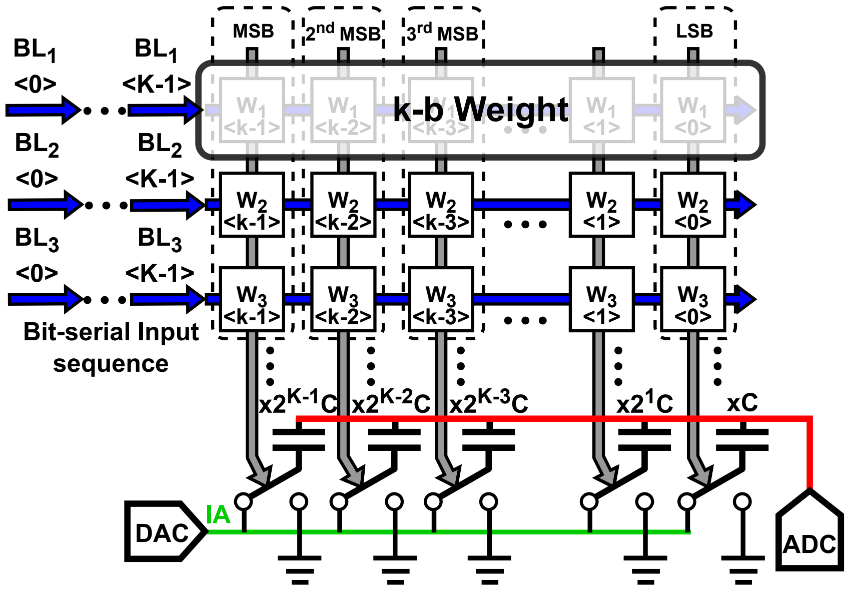


Figure 1.12: A multi-bit CIM array with binary weight capacitor ladder (13).

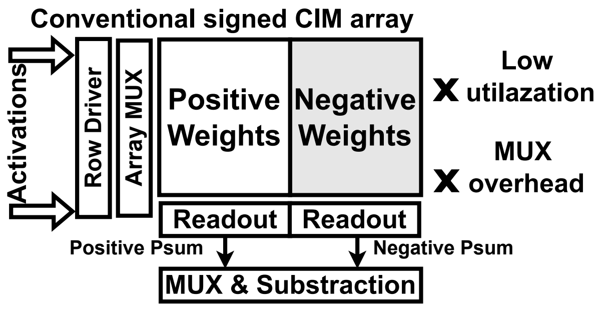


Figure 1.13: Conventional separated signed weight in CIM array.

While this scheme enables multi-bit computation, the capacitance grows exponentially with increasing bit width, leading to significant decreases in throughput and area efficiency as precision increases.

In a typical CNN computation process, the output of the previous layer, after passing through the ReLU activation function, is non-negative. However, weight data can be both positive and negative. While digital in-memory computing can handle signed data due to its flexible architecture, multi-row parallel analog computation faces challenges in handling multi-bit signed numbers.

A common approach is to store positive and negative weights separately in two distinct storage blocks, as depicted in Figure 1.13. This method requires 2K storage units for a K-bit signed weight, significantly reducing storage density. For instance, (44) employs this approach to implement an in-memory computing chip that supports signed weights. The weights are divided into positive and negative arrays, with each ReRAM BL storing 1 bit of information. Four ReRAM units on the same WL are used to store a 3-bit signed weight (1-bit sign, 2-bit data). By interacting with the input data using both positive and negative weights and then subtracting the results, the correct product of the signed weight and input data is obtained. This method not only reduces storage density but also necessitates additional subtraction circuitry.

1.5 Challenges and Contribution

As elaborated in the preceding sections, while in-depth analysis has revealed the high throughput and energy efficiency advantages of CIM architectures, this research also clearly highlights a series of challenges encountered in practical applications. These challenges primarily include:

- From a system-level perspective, determining the optimal neural network architectures for various CIM architectures is a complex task. Different network optimization methodologies offer varying trade-offs between accuracy and computational complexity. Given that the suitability of CIM architectures for different networks may vary, co-optimization of architecture and algorithms is crucial to achieve higher parallelism and energy efficiency while maintaining accuracy.

- From an architectural-level perspective, it is crucial to explore the potential of different CIM architectures for various applications. Analog-domain CIM architectures, while offering advantages in energy efficiency and storage density, are limited by accuracy and parallelism constraints, making them suitable for relatively small network applications in edge computing. In contrast, digital-domain CIM architectures, though less energy-efficient and denser, are robust to PVT variations and well-suited for high-frequency, accuracy-oriented tasks. However, the use of SRAM in digital-domain CIM architectures can limit storage density and overall area/energy efficiency. A comprehensive analysis of application-specific CIM architectures is therefore essential to optimize performance and resource utilization.

- From a macro-level perspective, both volatile and emerging non-volatile memories are susceptible to PVT variations and exhibit intrinsic non-linearity, particularly in analog-domain CIM architectures. While advanced macro designs (6; 35) have been proposed to enhance computational robustness against such variations, the full potential of SRAM and ReRAM-based macros for practical CIM applications remains to be realized. The challenge lies in the intricate task of specifying CIM macro implementations that effectively harness the capabilities of SRAM and ReRAM memories.

To address these issues, this thesis is dedicated to achieving higher energy efficiency and computational performance through in-depth technical research and innovative design.

For system perspective, this thesis focus on in-depth optimization towards neural networks. We proposed two major types of optimization topologies for CNNs and transformers.

- For CNNs, this thesis employs Neural Architecture Search (NAS) to search for the optimal precision of each layer. The objective of this approach is to minimize data bitwidth while maintaining network inference accuracy, thereby reducing storage requirements and computational complexity, and improving overall computational efficiency. Furthermore, Lasso regularization is applied to the network for bit-level sparse quantization. Through this approach, the network training process is biased towards driving each bit of every weight closer to "0", resulting in a higher proportion of bit-level zeros. This strategy makes the network significantly sparser, further reducing computational complexity and energy consumption, while also providing more flexibility for subsequent CIM optimization. By combining NAS and bit-level sparse quantization, our CNN optimization strategy not only improves energy efficiency but also preserves model inference accuracy.

- For transformers, this thesis employs a Ternary Weight Splitting (TWS) strategy, leveraging the flatness of the ternary loss landscape to facilitate the training of binary models. While weight binarization inevitably introduces quantization loss and leads to a significant degradation in accuracy, we mitigate this by adopting knowledge distillation during training. After weight splitting and initial training, we further fine-tune the binary model using distillation loss to seek an optimal solution in the new loss landscape, ensuring the binarized model performs well on downstream tasks. For activations, we utilize Brain Floating Point 16 (BF16) to quantize activations, striking an optimal balance between accuracy and computational complexity. By combining TWS and knowledge distillation, our transformer optimization strategy achieves a balance between computational complexity and inference accuracy.

For the CIM architecture, the target is to determine the optimal CIM architecture for various algorithms and data characteristics. We propose three types of architectures, both SRAM-based and ReRAM-based, for acceleration: analog-domain charge-based SRAM CIM accelerator, digital-domain SRAM CIM accelerator, and analog-domain charge-based ReRAM CIM accelerator. Notably, this PhD thesis summarizes the following works:

- Firstly, an analog-domain charge-based SRAM CIM accelerator is proposed to effectively support NAS-sparsity co-optimized mixed-precision neural networks. This design enable efficient mixed-precision operation by introducing a novel analog CIM circuit for signed weight mixed-precision shift-and-add operations. This circuit significantly reduces area overhead and offers flexible operand bit-width adjustment, enabling efficient handling of mixed-precision computations with signed weight data. Furthermore, a bit-level sparsity-aware ADC is proposed to adaptively adjust to the network's sparsity, leading to reduced power consumption and improved computational efficiency.
- Secondly, a digital-domain SRAM CIM accelerator is proposed to effectively support weight-binarized BERT models. We propose a BF16$\times$1-b SRAM-based CIM PE to support floating-point operations with high parallelism. The proposed batch-wise mantissa shifter in CIM PEs and exponent rounder in CIM PE groups reduce computational error by two orders of magnitude while incurring only a $\sim 5\%$ area and power overhead. We further integrate a group

vector systolic array with the CIM architecture, balancing throughput and register dynamic power. A data frame standardization strategy is employed to eliminate tensor reshaping overhead and facilitate systolic data flow.

- Thirdly, an analog-domain charge-based SRAM CIM accelerator is proposed to effectively support NAS-optimized mixed-precision neural networks. This work extends upon the first work by incorporating effective systolic data control.

From macro perspective, the objective is to minimize the impact of device nonlinearity on accuracy, enhance robustness to PVT variations, improve throughput and reduce power consumption. To achieve these goals, we proposed three types CIM macros: charge-based bootstrap 9T1C SRAM macro, digital-domain BF16×1-b SRAM CIM macro, charge-based 5T2R1C ReRAM CIM macro. Notably, this PhD thesis summarizes the following works:

- Firstly, we propose an innovative charge-based boostrap 9T1C SRAM macro that enables 1-bit MAC operations while preserving standard SRAM functionalities. Our design focuses on mitigating the impact of PVT variations by employing bootstrap sampling of capacitors, ensuring robust performance in highly parallel systems.

- Secondly, as digital-domain CIM is relatively immune to device variations, the BF16×1-b SRAM CIM macro design aims to enhance throughput and increase floating-point computational accuracy while minimizing power consumption. This macro enables bit-parallel computation, significantly reducing the cycle count compared to conventional bit-serial methods. Additionally, the deployed XOR logic gate requires fewer peripherals than traditional XOR gates, which use two transmission gates in CIM implementations (45).

- Thirdly, a charge-based 5T2R1C ReRAM macro is proposed. By leveraging the voltage transfer characteristics of inverters to mitigate the impact of ReRAM device variations, this macro significantly enhances MAC accuracy. This macro enables 1-bit MAC operations while preserving standard ReRAM functionalities. Our approach offers significant improvements in throughput and energy efficiency for MAC operations while satisfying noise constraints.

For every neural network implemented on analog-domain charge-based SRAM CIM accelerator, digital-domain SRAM CIM accelerator, and analog-domain charge-based ReRAM CIM accelerator, we evaluate their performance in device, architecture and system levels. All the implementations show higher throughput and accuracy performance with better energy-efficiency

1.6 Thesis Organization

This thesis involves the entire design flow from device, circuitry, architecture to system perspectives for Computing In-Memory. It is arranged into following chapters. Chapter 2 introduces the optimization deployed in this thesis to reduce computational redundancy without sacrificing performance in CIM. Optimization methodologies like neural architecture search, prune and binarization are discussed. Chapter 3 proposes a analog-domain charge-based 9T1C SRAM CIM accelerator for mixed-precision and sparsity co-optimized neural networks, which demonstrates 4-bit equivalent accuracy while maintaining 2-bit accuracy. Chapter 4 presents a digital-domain SRAM CIM accelerator for binarized BF16X1-bit BERT networks. Chapter 5 shows a analog-domain charge-based 5T2R1C ReRAM CIM architecture for NAS optimized mixed-precision networks. Chapter 6 reveals the performance of proposed accelerators for different neural networks, and Chapter 7 summarizes this thesis and shows some potential topic as future work.

Chapter 2

Neural Network Optimization

2.1 Precision and Sparsity Co-optimization

This section showcases the NAS-sparsity co-optimization flow for Chapter 3. The idea of NAS has been well applied in many cases (like (46–49)). In this section, we introduce a two-stage NAS method designed for bit-level mixed-precision quantization and sparsity, as depicted in Figure 2.2. The primary objective of this method is to enhance the performance of Bit-level Sparsity Quantization (BSQ) (50), which utilizes bit-level group lasso but may yield less stable training outcomes. To bolster robustness, we propose a two-stage approach. The first stage involves differentiable neural architecture search, aimed at determining the initial weight bit assignments. Subsequently, in the second stage, we employ BSQ in conjunction with bit-level group lasso to further introduce sparsity at the bit level for each element. The synergy of these two stages yields enhanced robustness and greater energy efficiency compared to the individual stages in isolation. And we introduce our algorithm in algorithm 1.

2.1.1 Bit-level Sparse Mixed-Precision Network

In this section, we explore the impact of different sparsity levels on network accuracy and hardware acceleration performance during network sparsification. As shown in Figure 2.1, most current sparsification efforts focus on block-level and element-

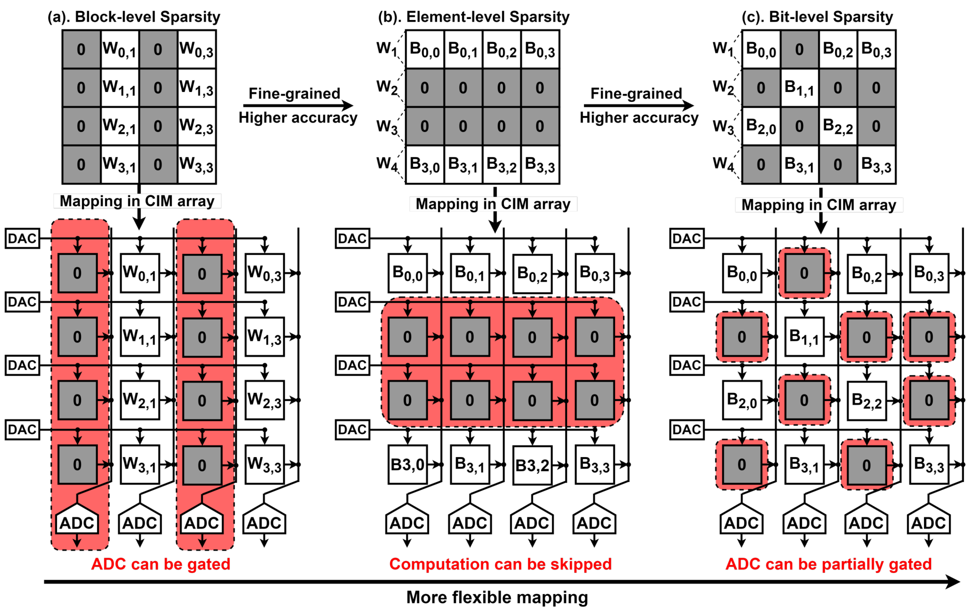


Figure 2.1: Comparison of sparsity mapping schemes in CIM arrays.

level sparsity. In contrast, finer-grained bit-level sparsity has received less research attention. However, employing bit-level sparsity allows for more flexible mapping of CNN onto CIM architectures. Therefore, to further optimize the performance of neural networks and better integrate them with hardware circuit design, enabling more efficient and flexible computation in resource-constrained environments by proper gating of computing resource. this paper proposes a novel approach: Bit-level Sparse Mixed-Precision Neural Network Quantization. This method combines the advantages of both quantization and sparsification techniques, aiming to achieve mixed-precision neural network computation while attaining high bit-level sparsity, thereby optimizing the computational efficiency and storage requirements of neural networks.

2.1.2 Stage I: Differentiable Neural Architecture Search

DARTS offers high search efficiency and robustness for network architecture exploration (51). It produces a supernetwork with multiple branches, each representing a quantization option (2, 4, or 8 bits) in the search space. The hyperparameter α for each branch indicates its contribution and is automatically trained alongside the supernetwork using gradient descent. The regularized branch contributions α^* are

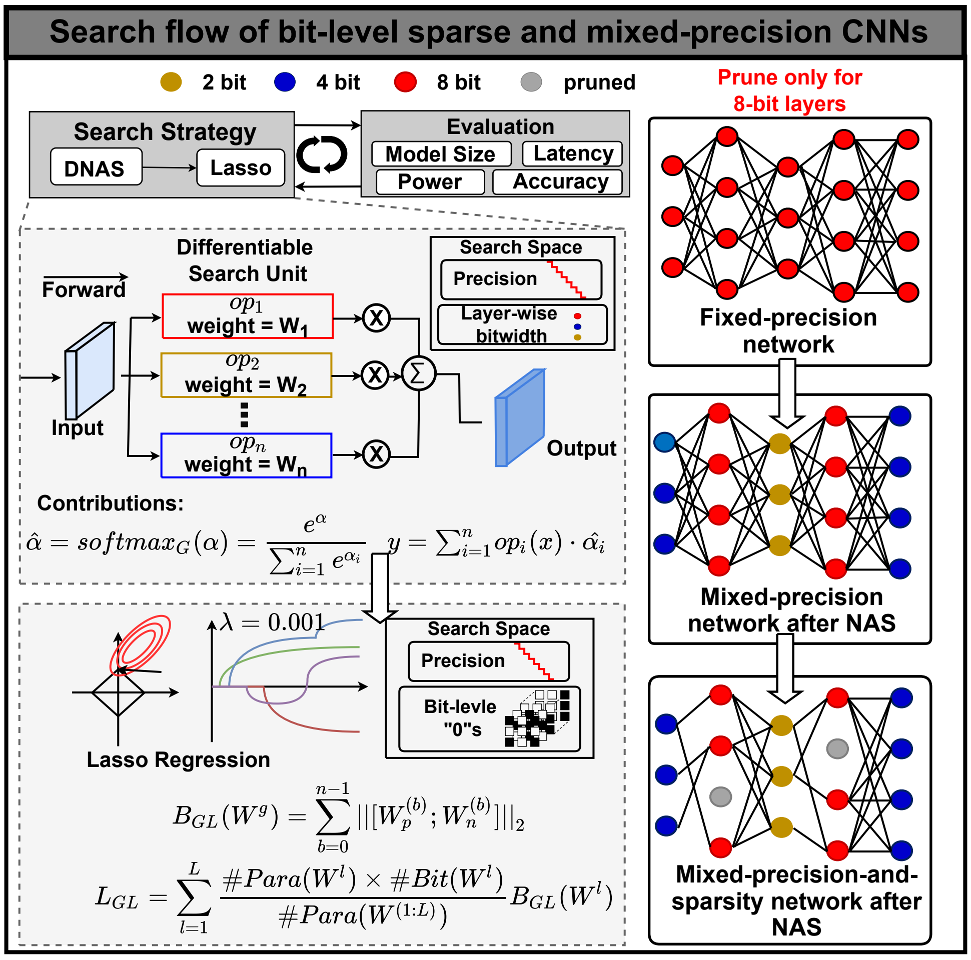

Search flow of bit-level sparse and mixed-precision CNNs
2 bit
4 bit
8 bit
pruned
Prune only for 8-bit layers
Search Strategy
DNAS
Lasso
Evaluation
Model Size
Latency
Power
Accuracy
Differentiable Search Unit
Forward
Input
op_1 weight = W_1
op_2 weight = W_2
op_n weight = W_n
Output
Search Space
Precision
Layer-wise bitwidth
Contributions:
$\hat{\alpha} = softmax_G(\alpha) = \frac{e^{\alpha}}{\sum_{i=1}^{n} e^{\alpha_i}}$
$y = \sum_{i=1}^{n} op_i(x) \cdot \hat{\alpha}_i$
$\lambda = 0.001$
Lasso Regression
Bit-levle "0"s
$B_{GL}(W^g) = \sum_{b=0}^{n-1} ||[W_p^{(b)}; W_n^{(b)}]||_2$
$L_{GL} = \sum_{l=1}^{L} \frac{\#Para(W^l) \times \#Bit(W^l)}{\#Para(W^{(1:L)})} B_{GL}(W^l)$
Fixed-precision network
Mixed-precision network after NAS
Mixed-precision-and-sparsity network after NAS

Figure 2.2: Search flow of bit sparse network and mixed-precision CNNs.

defined as:

$$\alpha^* = softmax_G(\alpha) = \frac{exp^{\alpha}}{\sum_i^n exp^{\alpha}} \tag{2.1}$$

The outputs of these branches are multiplied by their respective hyperparameter and then summed as:

$$y = \sum_{i=1}^{n} op_i(x) \cdot \hat{\alpha}_i \tag{2.2}$$

where the op_i represents the operation in the branch i, x denotes the input and y is the summed output. Furthermore, to induce the weights to low bit, a complexity-accuracy co-aware loss function (46) is applied as:

$$L = L_{CE} + \alpha \cdot L_{PF} \tag{2.3}$$

Algorithm 1 Bit-level sparsity mixed-precision CNN

```
Input: the input feature $x$, the NAS searching space.
Output: the output feature $y$, the bit-level sparsity mixed-precision model.
 1: Stage I: Differentiable neural architecture search
 2: for each layer i in CNN do
 3:     for each searching bit $b$ in NAS searching space do
 4:         $y+ = op_b(x) \cdot \alpha^*$, search the bit-width for each layer operations.
 5:     end for
 6:     $L = \#PARAMs + \beta \cdot \#FLOPs$, use loss function induces efficient bit-width and
        train the model.
 7: end for
 8: Select the best bit width through $\alpha^*$ which is a rough bit width for each layer in the
    CNN.
 9: Stage II: Bit-level sparsity quantization
10: Use the pre-train mixed-precision model from stage I to set the origin bit-width of
    each layer.
11: Forward train with $W_q = \frac{1}{2^n - 1} \cdot round[\sum_{b=0}^{n-1} W^b 2^b]$.
12: Using bit-level group lasso $B_{GL}(W) = \sum_{b=0}^{n-1} \parallel W_p^{(b)}; W_n^{(b)} \parallel_2$ to induce the weight
    sparsity.
13: Get the final bit-level sparsity mixed-precision model.
```

$$L_{PF} = \#PARAMs + \beta \cdot \#FLOPs \tag{2.4}$$

where L_{PF} is the co-aware loss of parameters and flops to train the supernetwork, L_{CE} is the cross-entropy loss, α and β are hand-set parameters for determining the allocation of different parts. After the training of supernetwork, the most recommended branch would be selected. With the quantization bits of each layer determining, this model will be fine-tuned for few epochs to form the pre-training model for *stage II*.

2.1.3 Stage II: Bit-level Sparsity Quantization

In *Stage I*, we obtain a mixed-precision pre-training model that exhibits high robustness and accuracy. However, this model primarily operates at the element-level, and still consumes unnecessary energy on hardware. To further prune the model, the bit-level sparsity quantization is applied. The essential idea of the pruning is to train and induce more bits of an element to zero, and then the bit-level group lasso is proposed and formulated as:

$$B_{GL}\left(W\right) = \sum_{b=0}^{n-1} \parallel W_p^{(b)}; W_n^{(b)} \parallel_2 \tag{2.5}$$

where $W_p^{(b)}$ and $W_n^{(b)}$ denotes positive and negative parameters. $W_p^{(b)}$;$W_n^{(b)}$ means the concatenation of those two tensors. With the group lasso, the loss function is designed as:

$$L = L_{CE} + \alpha \sum_{l=1}^{L} \frac{\#Para\left(W^l\right) \times \#Bit\left(W^l\right)}{\#Para\left(W^{1:l}\right)} B_{GL}\left(W^l\right) \tag{2.6}$$

where the L_{CE} represents the conventional cross entropy loss, α is a hyperparameter determining the loss proportion of the number of parameters and B_{GL} means the function of group Lasso regularizer. With the approach, bit can thus be safely removed for the precision reduction.

2.2 Binarization of BERT

This section showcases the transformer binarization flow for Chapter 4. While transformer models like BERT achieve exceptional performance (52), their high computational and memory requirements pose challenges for deployment on edge devices. Binarization reduces model size and computational complexity by representing weights and activations with fewer bits. However, directly binarizing transformer models often leads to significant accuracy degradation due to the sensitivity of attention mechanisms to quantization.

To address this, we propose a two-step binarization process illustrated in Fig. 2.3 to decompose ternary weights into binary components while preserving performance. To further mitigate accuracy loss, we employ Knowledge Distillation with a full-precision teacher model. The overall process is summarized in Algorithm 2.

2.2.1 Ternary Weight Splitting

The core idea of TWS is to approximate full-precision weights using a combination of binary bases and scaling factors. Specifically, each full-precision weight matrix W is decomposed into two binary matrices $W^{(b1)}$ and $W^{(b2)}$ with corresponding scaling coefficients α_1 and α_2. This decomposition allows the weights to be represented using ternary values initially, which are then split into binary components while preserving the original model's performance.

Given a pre-trained ternary weight matrix W^t and its quantized counterpart $\hat{W}^t$, the splitting equivalency requires:

$$W^t = W^{(b1)} + W^{(b2)}, \quad \hat{W}^t = \hat{W}^{(b1)} + \hat{W}^{(b2)} \tag{2.7}$$

To satisfy this condition, we define the binary weight components as:

$$W_i^{(b1)} = \begin{cases} a \cdot W_i^t, & \text{if } \hat{W}_i^t \neq 0 \\ b + W_i^t, & \text{if } \hat{W}_i^t = 0,\ W_i^t > 0 \\ b, & \text{otherwise} \end{cases} \tag{2.8}$$

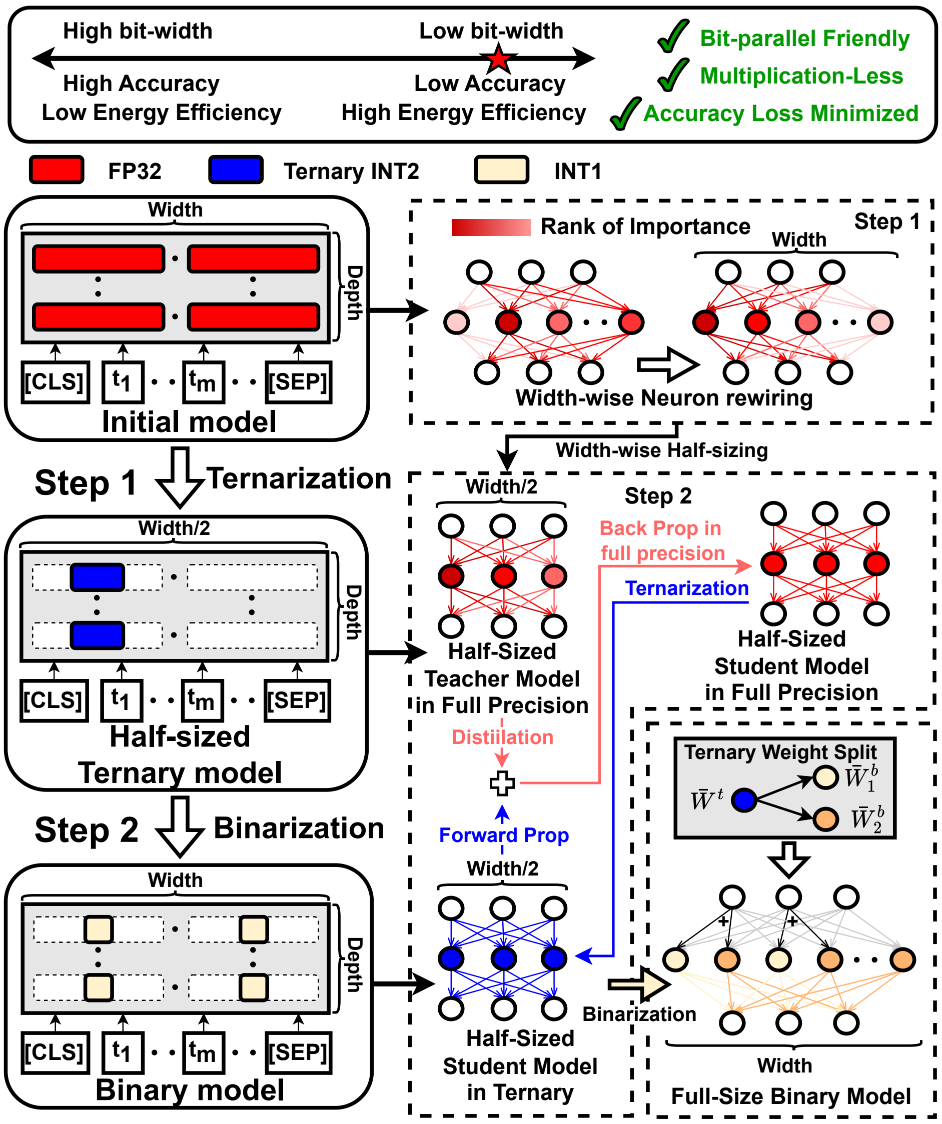


Figure 2.3: Overview of the two-step binarization flow of BF16×1-b BERT model.

$$W_i^{(b2)} = \begin{cases} (1-a)\cdot W_i^t, & \text{if } \hat{W}_i^t \neq 0 \\ b, & \text{if } \hat{W}_i^t = 0,\ W_i^t > 0 \\ b + W_i^t, & \text{otherwise} \end{cases} \tag{2.9}$$

Here, a and b are variables determined by solving the equations to minimize the reconstruction error between the original and approximated weights. By carefully designing $W^{(b1)}$ and $W^{(b2)}$, we ensure that the binary model inherits the performance of the ternary model after splitting.

Algorithm 2 Binarization Flow of TransFormer Model

Input: Pre-trained transFormer model weights W, learning rate η, number of epochs E, batch size B.

```
 1: Initialize scaling factors $\{\alpha^{(l)}\}$ For each layer $l$.
 2: Step 1: Ternarization and Knowledge Distillation
 3: for epoch = 1 to $E$ do
 4:     for each batch $\mathcal{B}$ of data do
 5:         for each layer $l$ do
 6:             Compute threshold $\Delta^{(l)} = t \cdot \mathbb{E}[|W^{(l)}|]$.
 7:             Quantize weights to ternary values using Equation (2.8) and (2.9).
 8:             Quantize activations to low-precision Format (e.g., BF16).
 9:             Compute layer output using $\hat{W}^{(l)}$ and quantized activations.
10:         end for
11:         Compute loss $\mathcal{L}$ with knowledge distillation.
12:         Compute gradients $\nabla_W \mathcal{L}$.
13:         for each layer $l$ do
14:             Update weights: $W^{(l)} \leftarrow W^{(l)} - \eta \nabla_{W^{(l)}} \mathcal{L}$.
15:         end for
16:     end for
17: end for
18: Step 2: Binarization and Fine-tuning
19: for each layer $l$ do
20:     Convert ternary weights to binary weights:
            $$\tilde{W}_i^{(l)} = \begin{cases} +1, & \text{if } \hat{W}_i^{(l)} > 0 \\ -1, & \text{if } \hat{W}_i^{(l)} < 0 \end{cases}$$
21: end for
22: Fine-tune the binary model with knowledge distillation.
23: Output: Binarized transFormer model with weights $\{\tilde{W}^{(l)}\}$.
```

2.2.2 Quantization with Knowledge Distillation

While weight binarization reduces model size and computational complexity, it can lead to accuracy degradation. To mitigate this, we simultaneously quantize activations and employ knowledge distillation during training.

Activation Quantization: Activations are quantized to a low-precision format to complement the binarized weights. We utilize Brain Floating Point 16 (BF16) for activations, which balances dynamic range and precision. The BF16 format retains the 8-bit exponent of standard Floating Point 32 (FP32) but reduces the mantissa to 7 bits, enabling efficient hardware implementation with acceptable accuracy.

Knowledge Distillation: We use a full-precision transformer model as the teacher and the binarized model as the student. The distillation process involves:

- *Intermediate-layer Distillation:* Minimizing the mean squared error between the student's and teacher's embeddings (E vs. $\hat{E}$), multi-head attention outputs ($M^{(l)}$ vs. $\hat{M}^{(l)}$), and feed-forward network outputs ($F^{(l)}$ vs. $\hat{F}^{(l)}$) for all layers l.
- *Prediction-layer Distillation:* Minimizing the soft cross-entropy between the student's logits $\hat{y}$ and the teacher's logits y.

The total loss function combines the intermediate and prediction-layer distillation losses:

$$\mathcal{L}\text{total} = \mathcal{L}_{\text{intermediate}} + \mathcal{L}_{\text{prediction}} \tag{2.10}$$

By training with knowledge distillation, the binarized model learns to mimic the teacher's behavior, which helps in recovering the performance lost due to quantization.

Further Fine-tuning: After splitting and initial training, the binary model may not have reached optimality in the new loss landscape. We further fine-tune the binary model using the distillation losses to seek a better solution, ensuring that the binarized model performs well on downstream tasks.

2.2.3 Activation Quantization

While weight binarization reduces model size and computational complexity, it can lead to accuracy degradation. This challenge is exacerbated in transformer models like BERT, where complex attention mechanisms are highly sensitive to quantization errors.

In this work, we utilize Brain Floating Point 16 (BF16) to quantize activations, striking an optimal balance between precision and computational efficiency. BF16 retains the 8-bit exponent from the standard 32-bit floating-point format, while reducing the mantissa to 7 bits. This configuration preserves the dynamic range necessary for accurate activation representation, significantly reducing both memory usage and computational overhead.

The use of BF16 offers several advantages for activation quantization. First, it enables efficient storage and faster computation without compromising the range needed for deep networks. Second, BF16 is naturally supported by many modern accelerators, facilitating hardware-level optimizations and ensuring minimal performance loss during inference. As a result, BF16 provides a suitable solution for activation quantization, achieving a substantial reduction in resource consumption while maintaining model accuracy, especially when combined with techniques like knowledge distillation.

2.2.4 Algorithm Flow

The complete binarization process is summarized in Algorithm 2. The algorithm consists of two main steps: ternarization with knowledge distillation, and subsequent binarization with fine-tuning.

In **Step 1**, we initialize scaling factors $\alpha^{(l)}$ for each layer l and perform ternarization of the weights. For each layer, we compute a threshold $\Delta^{(l)}$ based on the mean of the absolute weights:

$$\Delta^{(l)} = t \cdot \mathbb{E}[|W^{(l)}|] \tag{2.11}$$

where t is a hyperparameter. We then quantize the weights to ternary values using this threshold:

$$\hat{W}_i^{(l)} = \begin{cases} +1, & \text{if } W_i^{(l)} > \Delta^{(l)} \\ 0, & \text{if } |W_i^{(l)}| \leq \Delta^{(l)} \\ -1, & \text{if } W_i^{(l)} < -\Delta^{(l)} \end{cases} \tag{2.12}$$

Activations are quantized to a low-precision format, such as BF16, to complement the quantized weights. The model is trained using knowledge distillation from a full-precision teacher model, where the loss function $\mathcal{L}$ incorporates both the original task loss and distillation terms.

In **Step 2**, we convert the ternary weights to binary weights by mapping non-zero ternary weights to their signs:

$$\tilde{W}^{(l)} = \text{Sign}(\hat{W}^{(l)}) \tag{2.13}$$

The binary model is then fine-tuned with knowledge distillation to recover any performance loss due to binarization. This fine-tuning helps the model adapt to the new weight representations while maintaining accuracy.

The final output is a binarized transformer model with weights $\tilde{W}^{(l)}$, optimized for efficient inference on edge devices with limited computational resources.

2.3 NAS-optimized Mixed-bit CNNs

This section showcases the NAS optimization flow for Chapter 5. Mixed-bit quantization refers to the transformation of weights and feature graphs in the network model into integer types, and different layers in the network have different integer bit-widths. Given that an extremely low-bit quantized network is energy-efficient but inaccurate, mixed-bit CNN is competent at balancing network performance and hardware efficiency.

As shown in Figure 2.4, the NAS algorithm in this section of work is based on differential Neural Architecture Search (DNAS) approach(53). DNAS begins with a super net to denote the architecture space $\mathcal{A}$, which is a computational Direct Acyclic Graph (DAG) and can be computed as:

$$\min_{\boldsymbol{\theta}} \min_{\boldsymbol{w}_a} \mathbb{E}_{\mathrm{a} \sim P_{\boldsymbol{\theta}}} \left[\mathcal{L} \left(\boldsymbol{m}_a, \boldsymbol{w}_a \right) \right] \tag{2.14}$$

where a represents a neural network belongs to the architecture space $\mathcal{A}$, $\boldsymbol{w}_a$ are the trainable weights of architecture a. Mask function $m_a \in \{0, 1\}$ indicates the execution on certain edges with probability of p, and $\mathcal{L}$ the loss function.

To benefit from lower-precision weights and activations in terms of effective and efficient inference of low-bit operations, we apply the following loss function, balancing the cross entropy term and the cost term with weighting function $\mathcal{C}(\cdot)$:

$$\mathcal{L} \left(a, \boldsymbol{w}_a \right) = \mathrm{CrossEntropy}(a) \times \mathcal{C}(\mathrm{Cost}(a)) \tag{2.15}$$

And the $Cost(a)$ is designed to lower the computational cost and adapt the lower-bit computation, with FLOP$(\cdot)$ denoting the number of floating operations:

$$\mathrm{Cost}(a) = \sum_{e_k^{ij} \in E} m_k^{ij} \times \# \mathrm{FLOP} \left(e_k^{ij} \right) \times \text{ weight-bit } \left(e_k^{ij} \right) \times \mathrm{act} - \mathrm{bit} \left(e_k^{ij} \right) \tag{2.16}$$

To be specific, $\boldsymbol{m}_a$ is the vector consisting of m_k^{ij} for all $e_k^{ij} \in E$ in architecture a, and $\mathbb{E}_{\mathrm{a} \sim P_{\boldsymbol{\theta}}} \left[\mathcal{L} \left(\boldsymbol{m}_a, \boldsymbol{w}_a \right) \right]$ is the loss function of stochastic super net. Besides, Gumbel SoftMax function(54) is used to directly compute the gradient of optimal architecture parameter, and make the neural network fully differentiable by 'soft-control' the edge selection, balancing between bias and variance. The probability

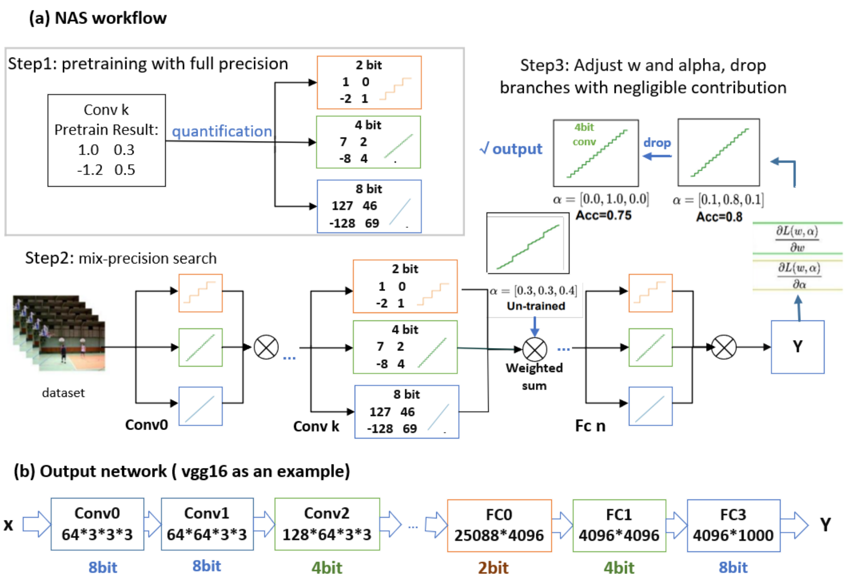


Figure 2.4: (a)DNAS Workflow. (b)NAS-optimized VGG16 Structure.

density function of Gumbel-Softmax distribution is defined as:

$$p_{\pi,\tau}\left(y_1,\cdots,y_k\right)=\Gamma(k)\tau^{k-1}\left(\sum_{i=1}^{k}\pi_i/y_i^{\tau}\right)^{-k}\prod_{i=1}^{k}\left(\pi_i/y_i^{\tau+1}\right) \tag{2.17}$$

In our work, we firstly pretrain the network with full precision to give a stable start point of the mixed-bit searching. Secondly, we apply retrainings after the mixed-bit searching, which is just like quantizations for normal networks.While the basic square variance loss function is used for accuracy concern, the complexity loss function is given in equation 2.18, in which model parameters number(#PARAMs) and model parameter number(#OPs) evaluate spatial complexity and temporal complexity respectively, and γ is a coefficient.

$$L_{comp}=\gamma\cdot\#PARAMs+(1-\gamma)\cdot\#OPs \quad (0\leq\gamma\leq 1) \tag{2.18}$$

The values of #PARAMs and #OPs are determined by both network and hardware in eq.(2.19) and eq.(2.20). l refers to different layers in the network, and each layer has its own bit-width($BW^{[l]}$), input channel number($C_{in}^{[l]}$), output channel number($C_{out}^{[l]}$), kernal size($K_y^{[l]}, K_x^{[l]}$) and input feature map size($H^{[l]}, W^{[l]}$) . Specifically, The ratio of storage space and the ratio of computation time required for different data bit-width varies among different hardware, which leads to different utilization. We apply two look up tables($LUT_{storage}$ and LUT_{calc}) of the two ratios to punish low hardware utilization. Therefore, the searching algorithm can also improve the utilization of our accelerator, which further improves the efficiency of the model.

$$\#PARAMs = \sum_{l=1}^{L}(LUT_{storage}(BW^{[l]}) \cdot C_{out}^{[l]}C_{in}^{[l]}K_x^{[l]}K_y^{[l]}) \tag{2.19}$$

$$\#OPs = \sum_{l=1}^{L}(LUT_{storage}(BW^{[l]}) \cdot C_{out}^{[l]}C_{in}^{[l]}(H^{[l]}W^{[l]})^2) \tag{2.20}$$

For the last step, we apply a gradient-descending strategy on differential NAS. We drop branches of bit-width assignment in the super-net if the branch has negligible contributions (α in Figure 2.4) on the final result. By doing so, the searching time can be reduced and searching is more stable. The process above is repeated until the mixed-bit network is determined and can be formulated as:

$$\begin{aligned} f\prime_o(x,y,z) = \sigma(&\sum_{a=0}^{U-1}2^a\sum_{b=0}^{U-1}2^b\sum_{i=0}^{K-1}\sum_{j=0}^{K-1}\sum_{z=1}^{C}f_i(x+i,y+j,z) \\ &\cdot K_{M,b}(i,j,z)) \end{aligned} \tag{2.21}$$

where the parameter U is precision operational mode. Index a and b represent the ath and bth bit in U-bit weight and IFM respectively. $\sigma()$ is denoted as the subsequent activation and pooling layer after convolution.

Chapter 3

Analog-domain SRAM-based CIM Accelerator

3.1 Introduction

The rapid advancement of artificial intelligence (AI) algorithms has profoundly transformed public's daily live. By optimizing algorithms and providing robust hardware support, AI applications have made significant impacts across various industries (55). However, the continuous iteration of algorithms has led to increased computational complexities and parameter sizes, challenging the capabilities of hardware computing power, particularly in resource-constrained edge systems (1; 2).

Edge-AI systems are indispensable for driving the advancement of artificial intelligence of things (AIoT) technology (56). To address the unique demands of edge-AI, collaborative optimization of algorithms and specialized circuit systems for hardware acceleration is paramount. Techniques such as precision quantization, data sparsity processing, and network fine-tuning are instrumental in reducing computational load and algorithm complexity. For instance, (3) leverages network redundancy to quantize neural networks to 8-bit integers while maintaining high accuracy and efficiency. (24) employs mixed-precision post-training quantization to optimize Transformer models, achieving a significant reduction in computational overhead without compromising performance. Furthermore, (4) introduces layer-wise fine-grained pruning

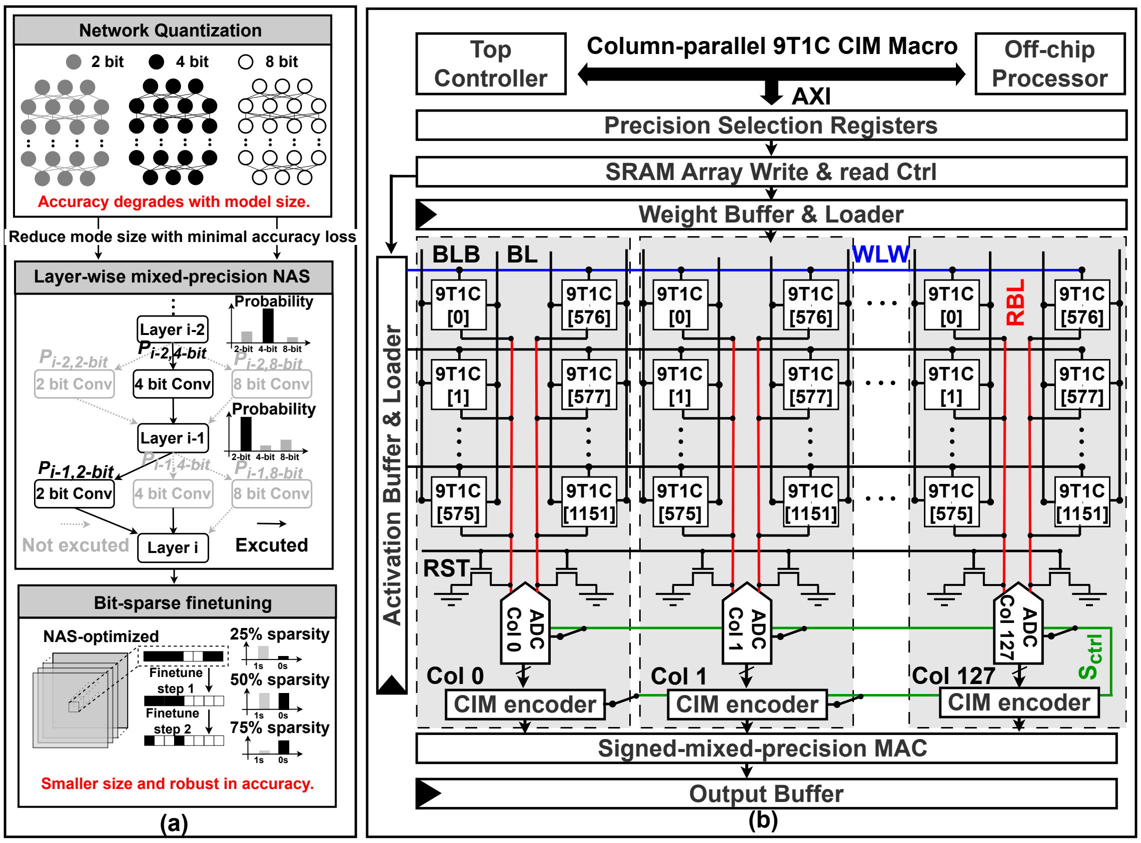


Figure 3.1: Proposed CIM system: (a) layer-wise network optimization flow by NAS-based algorithm. (b) column-parallel 9T1C CIM macro structure.

for efficient sparsification.

Simultaneously, efficient computation requires hardware design tailored to computational demands and data characteristics. Neural networks mainly use parallel multiply-accumulate (MAC) operations for tasks like convolution and matrix calculations (1). To enhance efficiency and reduce latency, many studies suggest computing-in-memory (CIM) architectures with parallel processing capabilities (44; 57–59). CIM architectures using charge-domain accumulation schemes improve error mitigation and linearity compared to traditional analog methods (9–11). However, these schemes are limited by parasitic effects, which can affect overall computational capacity (12–14). Additionally, hardware designs optimized for specific data characteristics achieve excellent computational performance. However, they need data feature extraction for energy-efficient processing while retaining standard computing capabilities. For instance, (50) introduces a bit-level sparse algorithm that offers finer-grained sparsity, enabling network compression while maintaining high accuracy. However, implementing bit-level sparsity in hardware design is challenging. Previous studies (60–63) focused on hardware optimization for block-level and element-level sparse networks. These approaches do not effectively use bit-level "0"s for energy efficiency and struggle with mixed-precision network calculations.

This work enhances energy efficiency without compromising accuracy through a two-pronged approach, as depicted in Figure 3.1. Firstly, it employs neural architecture search (NAS) to determine layer-wise optimized precisions and sparsities for convolutional neural networks (CNNs). Subsequently, it proposes a 144-Kb charge-domain signed mixed-precision (2/4/8-bit) CIM accelerator, featuring a bootstrapped 9-transistor and 1-capacitor (9T1C) SRAM cell and a bit-level sparsity-aware Analog-to-Digital Converter (ADC). The design achieves high-linearity parallel accumulation operations for AI computing and implements a hardware and software co-optimization system for data characteristics. This work demonstrates an impressive energy efficiency of up to 135.19 TOPS/W with NAS-optimized ResNet-18, while maintaining accuracy equivalent to 4-bit precision in 28nm CMOS process.

3.2 Related Work

3.2.1 CNN NAS Optimization Challenges for Edged AI

To address the hardware constraints of edge computing, NAS (51; 64–66) has shown it can automatically create layer-wise quantized architectures balancing accuracy and complexity. Differentiable Architecture Search (DARTS) (51) uses a continuous SGD-based search strategy, reducing resource needs compared to reinforcement-learning-based NAS methods (64; 66). However, DARTS and similar approaches struggle with weight sparsity due to difficulties in defining the design space and achieving hardware adaptivity (51; 65).

Other network compression techniques, such as pruning and its various optimization methods, are used to reduce model size and build optimized networks (67–69). For example, (70) proposes a layer-wise trainable threshold for each layer, while

(71) views pruning as a minimization problem in the training objective. However, higher pruning ratios can disrupt the balance between complexity and accuracy. For instance, pruned MobileNet-v2 sees a 19% accuracy drop due to inter-layer dependencies when compared to unpruned one.

Designing neural networks with layer-wise quantization and sparsity faces two main challenges: accuracy loss and altered gradient propagation. This work employ a differentiable NAS-obtained network backbone to combine mixed-precision NAS and bit-level sparsity optimization with balanced high energy efficiency and accuracy for CIM-based edge devices.

3.2.2 CIM Architecture Challenges for Parallel Computing

To enhance the throughput of CNN accelerators, parallel computation is essential for CIM architectures. In digital CIM work, which offers excellent noise immunity and computation accuracy, an adder tree block is typically used for parallel accumulation. However, the hardware overhead including area and power consumption is significant. In previous works, much effort has been made to optimize large bitwidth adder trees to ensure the performance of the CIM accelerators (72; 73).

Analog CIM architectures excel in parallel computing by using analog data accumulation instead of digital adders. However, they face challenges like PVT sensitivities and device variations. Charge-domain CIM architectures, benefiting from capacitors' high linearity, offer better computational accuracy than other analog CIM ones. As CIM array size grows, the parasitic capacitance of the bitline for readout (BLR) also increases, which reduces the output voltage range (13; 14). To enhance throughput while maintaining computational accuracy, this work proposes a new charge-domain bootstrapped CIM macro cell with a wide programmable output range.

3.2.3 Sparsity Data Processing Challenges on CIM structure

Bit-level sparsification enhances flexibility in weight mapping by breaking down weights into individual bits. This allows more efficient use of storage and mapping operations. Compared to block-level and element-level sparsification methods, bit-level sparsification offers better efficiency in both storage and computational resources (60–63). By providing more granular control over weight representation, it optimizes the use of hardware resources, potentially improving performance and energy efficiency in CIM architectures.

In (74), a bit-level sparsity-induced activation quantization method is proposed to reduce the dynamic computational energy of CIM accelerators. In a serial-in-bit computing accelerator using a 1-bit DAC, leveraging higher ratios of bit-level sparse input activation data increases column current amplitude accumulation, reducing dynamic power consumption. They propose a non-uniform activation quantization method based on the sum of quadratic terms to enhance the bit-level sparsity of activation data, without applying bit-level sparsity to weights. (75) used bit-level weight sparsity to deactivate certain capacitors in weighted accumulation, reducing dynamic power consumption. However, using a binary ladder capacitor accumulation scheme decreases area efficiency due to the large number of capacitors involved. To optimize neural network performance and enhance integration with hardware circuit design for efficient computation, this design proposes a bit-level sparsity-aware ADC to fully exploit network characteristics.

3.3 CIM Accelerator Architecture

3.3.1 Charge-domain parallel CIM array

In Figure 3.2, the SRAM array comprises 1152 $\times$ 128 9T1C SRAM cells, which are responsible for the multiplication operation of the input bits and the stored weight bits. These multiplication results are accumulated by charge sharing through the accumulation capacitors. Finally, the MAC results of the input data and weight

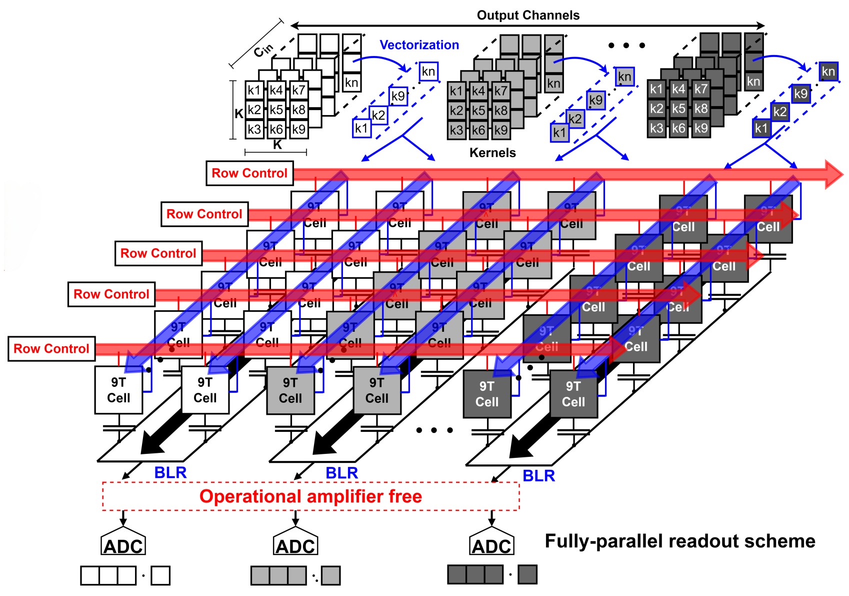


Figure 3.2: Proposed CIM array with parallel computing.

data are output on BLR.

For the readout module as the array interface, the column-parallel structure is employed. Thanks to the direct voltage output generated on BLR as the accumulation result, no voltage clamp circuit is required and BLR is connected to ADC directly. The ADC design also omits a sampling-and-hold (S&H) circuit, the BLR voltage can be preserved throughout ADC conversion process and subsequently utilized as the input to the comparator. Within the accumulator block, ADC outputs from individual columns undergo shifting based on predetermined weight factors and are concurrently accumulated to enhance throughput performance. These shifting operations are configurable to facilitate mixed-bit operations.

3.3.2 Bootstrap CIM SRAM Cell

In Figure 3.3, the proposed bootstrapped CIM macro features a 9T1C SRAM cell, combining a 6T SRAM for weight storage, a 3T unit with 3 IO NMOS transistors for input signal toggle, and an accumulation capacitor for charge-domain processing. The low standard deviation of capacitors (0.6% for 1.33fF in 28nm process) ensures high linearity and accuracy in analog accumulation (34). This design integrates digital storage with precise analog computation for improved CIM performance.

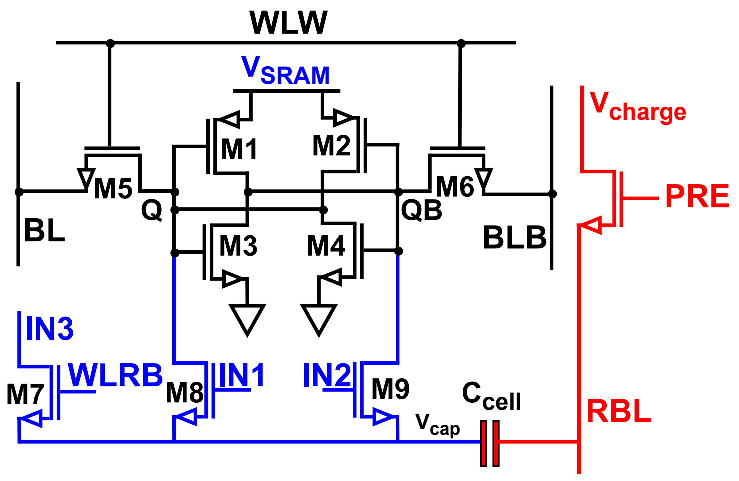


Figure 3.3: Schematic of proposed 9T1C bootstrapped CIM macro.

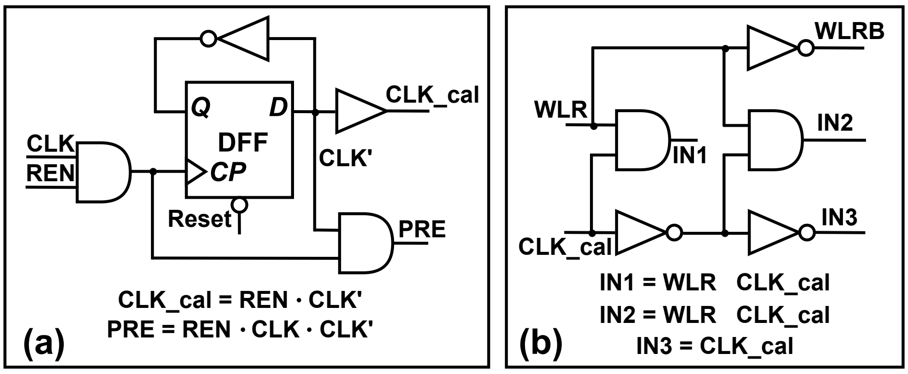


Figure 3.4: Control peripherals for macro array. (a). array column control circuits; (b). array row control circuits.

To enable "AND" operation with bootstrap mechanism, control peripherals and expressions for macro array are presented in Figure 3.4. The column control peripheral is deployed for column precharge signal and gated clock signal generation. The row control peripheral is responsible for preprocessing the input signal WLR from one pulse signal to three step signals IN1∼IN3. Concurrently, the precharge voltage is loaded, thereby enabling the bootstrap of the accumulation capacitor and raising output voltage. As illustrated in Figure 3.5, detailed "AND" operation procedure is presented.

The power supply of the row control block driver is controlled by V_{ctrl}, allowing the array output voltage to be adjusted. The timing control signal generator divides the external clock signal CLK to obtain the calculation clock signal CLK_calc and the precharge control signal.

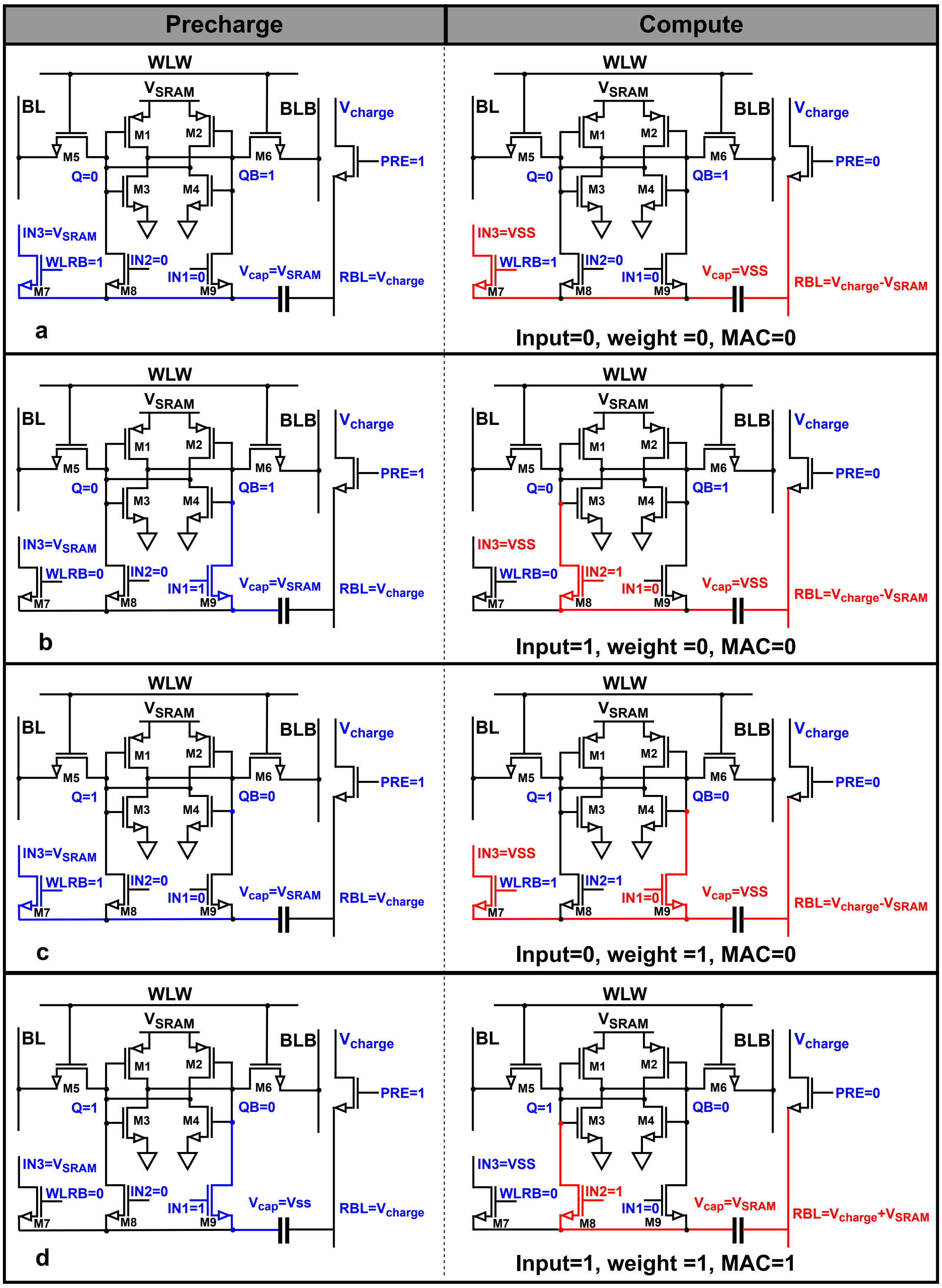


Figure 3.5: Macro "AND" logic operation procedures.

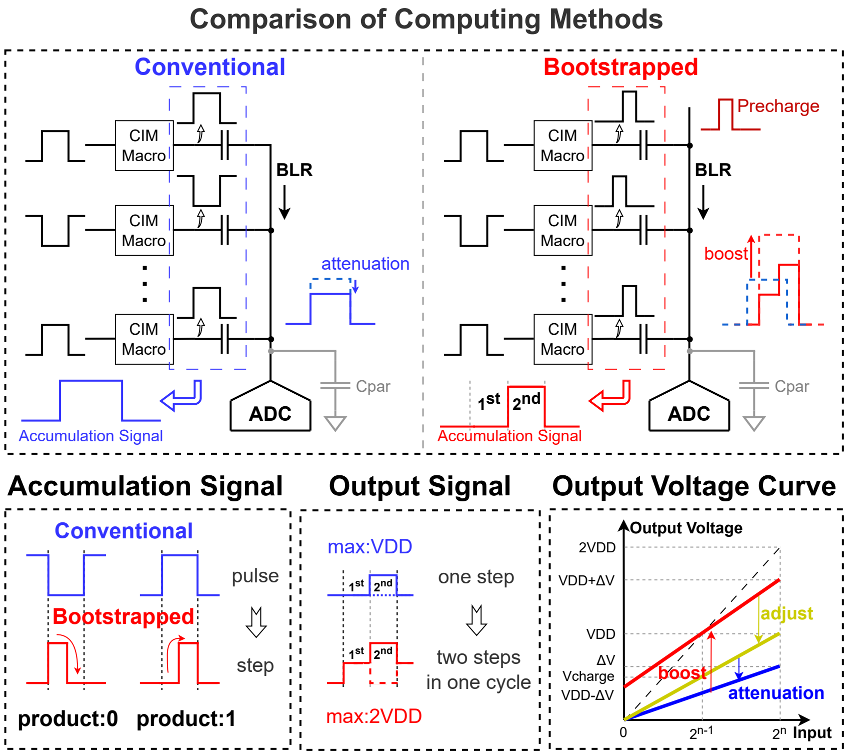


Figure 3.6: Capacitive accumulation method comparison.

The calculation cycle of the bootstrapped array is divided into precharge and output phases. In 1^{st} phase, the accumulation capacitor is precharged with a voltage based on the product. In 2^{nd} phase, the output voltage swing can be augmented to a level that is twice the SRAM operating voltage ($2V_{SRAM}$) through the input signal toggle for parasitic capacitance compensation. The output voltage range of the proposed cell is determined by the operating voltage of the SRAM cell and the precharge voltage, and is defined as $V_{charge} - V_{SRAM} \sim V_{charge} + V_{SRAM}$. The SRAM operating voltage V_{SRAM} acts as the drive voltage to regulate the output voltage swing, while the precharge voltage V_{charge} acts as a bias voltage. This results in a more optimal matching of the ADC quantization range, taking into account the uncontrollable parasitic capacitance influence. The output voltages of the proposed cell and the conventional cell are expressed as:

$$V_{out,conventional} = \frac{C_1}{C_0 + C_1 + C_{par}} \cdot VDD \tag{3.1}$$

$$V_{out,bootstrapped} = V_{charge} + \frac{(C_1 - C_0) \cdot V_{SRAM}}{C_0 + C_1 + C_{par}} \tag{3.2}$$

where C_0 and C_1 represents the equivalent capacitance with a product of "0" and "1" in the CIM array, respectively. C_{par} represents the parasitic capacitance of bitline for read (BLR).

Figure 3.6 compares the proposed accumulation method with the conventional ones (9; 10), highlighting three key differences. First, the accumulation signal changes from a pulse signal to a step signal. Second, the reset voltage of BLR shifts from ground to V_{charge}. Third, the output signal transforms from an attenuated single-step signal to a bootstrapped double-step signal. The proposed SRAM design, leveraging capacitor robustness and the bootstrapped accumulation method, enables fully parallel computing with minimal accuracy degradation from device nonlinearities and parasitics.

3.4 Sparsity-aware Column-parallel Readout

3.4.1 Bit-level Sparsity-aware ADC

Working Principle

Figure 3.7 depicts the proposed bit-level sparsity-aware ADC schematic. The structure of successive-approximation-register (SAR) ADC is selected as the output of CIM array in this article due to its excellent energy efficiency characteristics and programmability (76). In Figure 3.8, an innovative approach for generating the control signal to gate the ADC capacitor array is introduced.

During the initial cycle, the ADC output signal serves as inputs to the one-hot code generation module. after inputs is reversed, 8-bit input data are reversed as "A". Then the calculation of "$A - 1$" is performed. This step leverages binary coding characteristics to reverse the data in "A" from left to right until reaching the first "1" position. By computing "$A\&\overline{A-1}$", a one-hot code is derived, representing the position of the first "1" from left to right in "A". In the following, the generated one-hot code is arranged in reverse order to obtain a one-hot code signifying the position of the first "1" in the inputs. Finally, the one-hot code is subsequently utilized to control whether the ADC capacitors are gated or not.

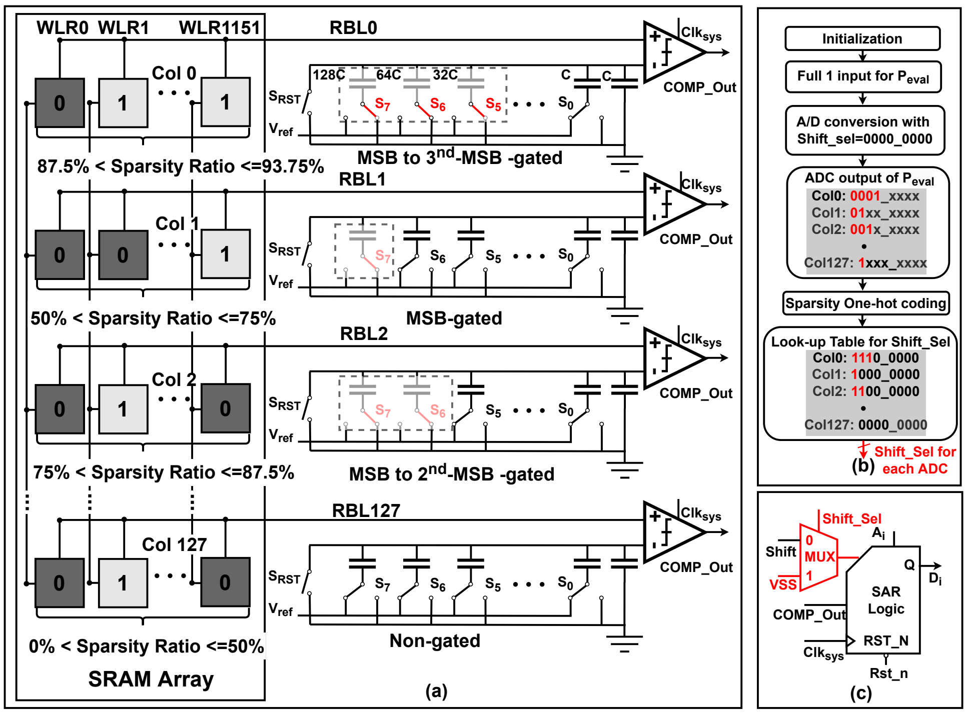


Figure 3.7: Schematic of proposed sparsity-aware ADC.

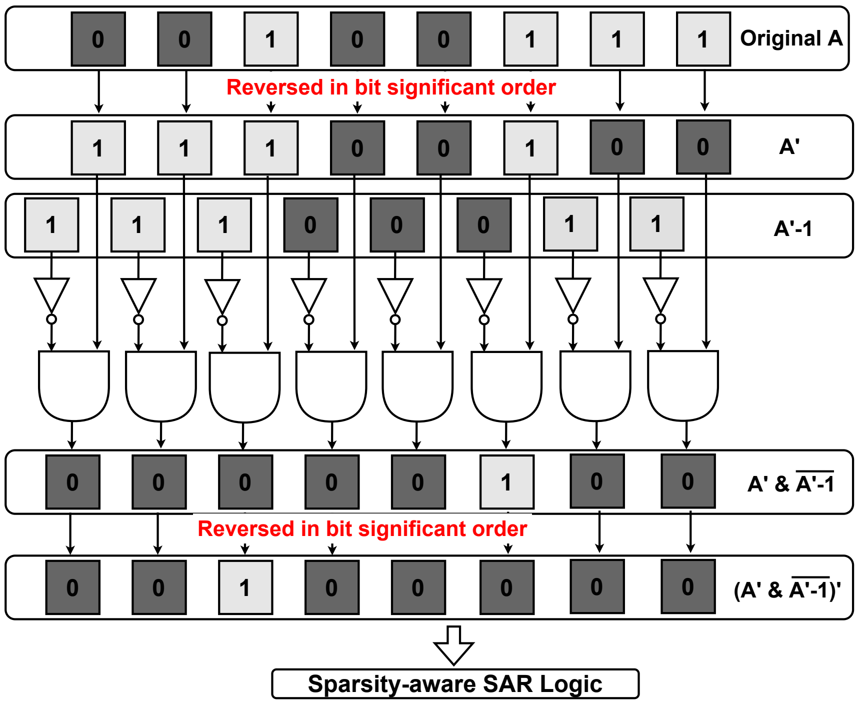

Original A
Reversed in bit significant order
A'
A'-1
A' & A'-1
Reversed in bit significant order
(A' & A'-1)'
Sparsity-aware SAR Logic

Figure 3.8: ADC control signal generation method in the form of one-hot code.

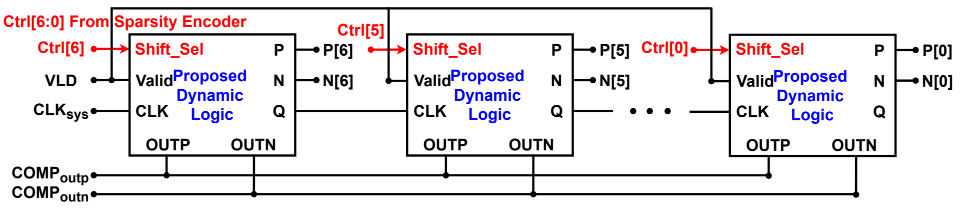


Figure 3.9: Proposed Sparsity-aware SAR logic block design.

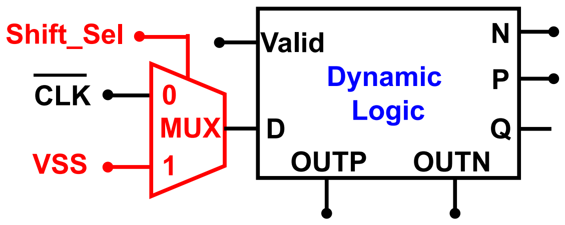


Figure 3.10: Proposed Sparsity-aware logic cell design.

SAR logic Design

Figure 3.9 and 3.10 show the proposed SAR logic block and cell designs to support sparsity-aware control signal to skip MSBs switching. Compared to conventional SAR logic block, this work adds a *Shift_ctrl* interface, which is used to receive the encoding results from the one-hot code generation module. The one-hot code result output indicates the comparison starting point. For example, if the generated code is "0010_0000", the SAR logic block with the proposed logic cell will start from the sixth bit and perform successive approximation for the digital signal from that point onward.

Waveform

The ADC uses two stages to enhance energy efficiency. In the first stage, shown in Figure 3.11, inputs are set to "1" and undergo "MUL" operations with their weights in the CIM array. Multiple rows of data are accumulated to obtain the result, representing the maximum achievable outcome during "AND" operations. For example, in a CIM array with 1152 rows and a column sparsity rate between 94.44% and 97.2%, the top three MSB bits of the column's result are consistently "0," while the lower five bits remain uncertain. Consequently, the ADC three MSB bits for this column are not used in subsequent calculations. Then the evaluation

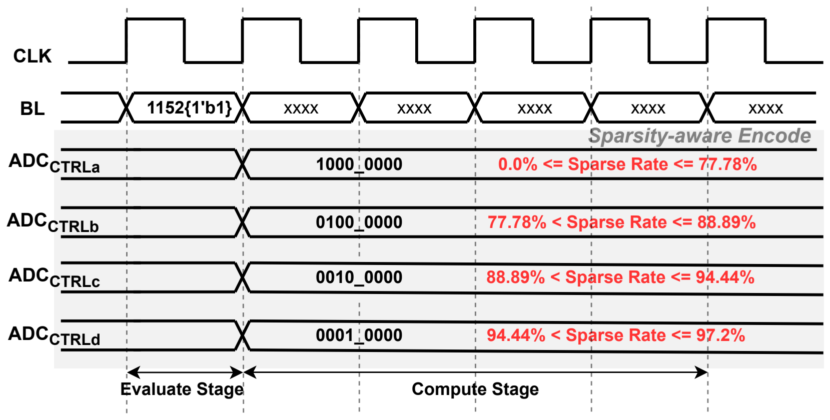


Figure 3.11: Waveform of ADC control signals in different sparse rates.

results of the first stage are sent to the one-hot code generation module to create control signals for gating the ADC MSB bits. In the second stage, all ADCs are partially gated based on these control signals, which remain unchanged during the calculation process to ensure stable system operation, thereby avoiding redundant power and latency.

3.4.2 Signed-mixed-precision MAC

In previous CIM implementations (77), processing signed weight data typically involves pre-extending the sign bit of each weight to a sufficient bit width before performing shift&add operation. However, this method requires additional storage array resources to store extended sign bits, resulting in storage resource waste. In addition, significant hardware overhead will be introduced after sign bit shift&add procedure. Figure 3.12 provides the design of signed-mixed-precision MAC. Each ADC generates an unsigned column MAC result. Signed computing contains two steps. **Step 1**, the configuration follows the bit-serial scheme for unsigned 2/4/8-bit modes.**Step 2**, sign bit extension is deployed to process positive and negative weights on one column. The output from sign bit column undergoes bit-width duplication accordingly for signed 2/4/8-bit modes, which could be formulated as:

$$\begin{aligned} S &= 2^n \cdot C_{ps} + 2^{n+1} \cdot C_{ps} + \cdots + 2^{n+d-1} \cdot C_{ps} \\ &= 2^n(2^d \cdot C_{ps} + \overline{C_{ps}} + 1) \quad n \in [2, 4, 8] \end{aligned} \tag{3.3}$$

where S stands for the total summation of sign bits after duplication, C_{ps} denotes

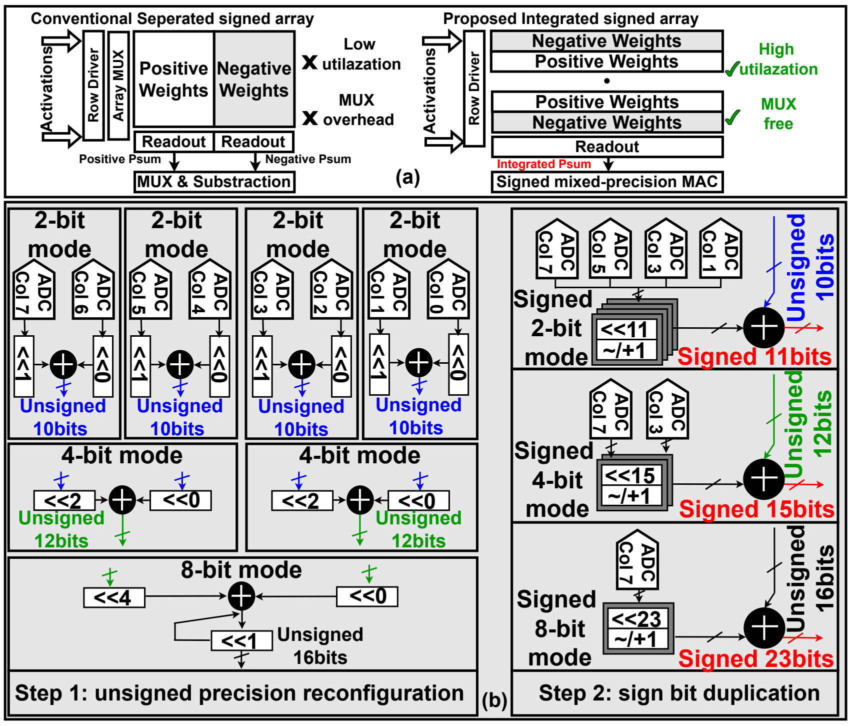


Figure 3.12: Signed-mixed-precision module and the corresponding workflow.

the accumulated partial sum of column-wise sign bits, e.g., ADC output of column 1, 3, 5 and 7 for signed 2-bit, column 3 and 7 for signed 4-bit and column 7 for signed 8-bit. Index d represent the required pre-extension bit-width. Compared to previous design that demands $n + (n+1) + \cdots + (n+d-1) = n \cdot d + d \cdot (d-1)/2$ bits of shifting space, proposed signed-mixed-precision only requires shifting space of $n + d$ bits. As shown in Figure 3.12, signed operations necessitate only 11, 15 and 23bits of shifting space for signed 2, 4, 8-bit respectively. Then the duplicated signed bit, together with its original 2's complement, is added back with the unsigned result from **Step 1** to generate the final signed MAC result. The proposed mixed-precision multiplier-accumulator (MAC) unit supports runtime cross-layer bit-width reconfiguration among 2-, 4-, and 8-bit operations with minimal overhead, achieved through the efficient implementation of three specialized counters.

3.4.3 Data Mapping and Reuse Method

To make use of the high throughput capability inherent in analog CIM, kernels of distinct input channels are vectorized on the host, which has a size of K^2C_{in} and

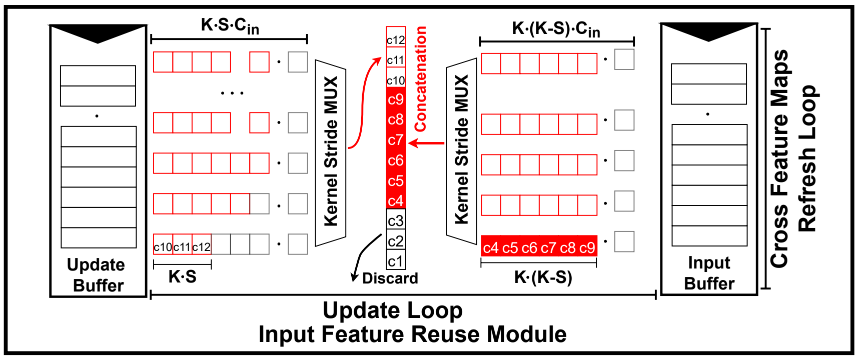


Figure 3.13: Input feature reuse module.

maximum C_{in} is 128. The reshaped weights are mapped into the same column within the CIM array, the row number of 1152 is specifically chosen to achieve an average utilization of 96.85% on ResNet-18. Besides, each column is connected to a SAR ADC and follows column-parallel scheme for output channel partial sum generation. To further optimized power consumption, a reuse module is deployed to prevent overlapped feature updating as shown in Figure 3.13. The data reuse module includes two SRAM buffers: 128x1152 input and 768x128 update, along with a selector for kernel size and stride granularity. When the convolution kernel strides, the overlapping features stay stationary, allowing for data reuse.

The corresponding experiment results of NAS-sparsity co-optimized CNNs on proposed analog-domain SRAM-based CIM accelerator is organized in section 6.1.

Chapter 4

Digital-domain SRAM-based CIM Accelerator

4.1 Introduction

Edge computing necessitates on-device network inference, which underscores the need to optimize transformer models for resource-constrained environments without compromising accuracy. The demands impose additional constraints on both algorithm optimization and hardware implementation in terms of area, power, and storage. To address the challenges of deploying transformer models on edge devices, various optimization methodologies have been explored. One common approach involves reducing network size and minimizing hardware usage. Popular techniques (24; 78; 79) include quantization, pruning and knowledge distillation.

Accelerating neural networks on edge devices often necessitates tailored hardware for specific dataflows. CIM architectures have emerged as promising candidates for MAC operations due to their potential for higher throughput and energy efficiency (11; 80). While analog-domain CIM approaches have been explored, they often suffer from device non-linearity and parasitic effects, limiting their performance (11). To address these limitations, full digital CIM accelerators have been proposed, enabling lossless acceleration of neural networks (16; 31; 45). However, digital CIM architectures typically introduce additional overhead due to memory cell connectivity and

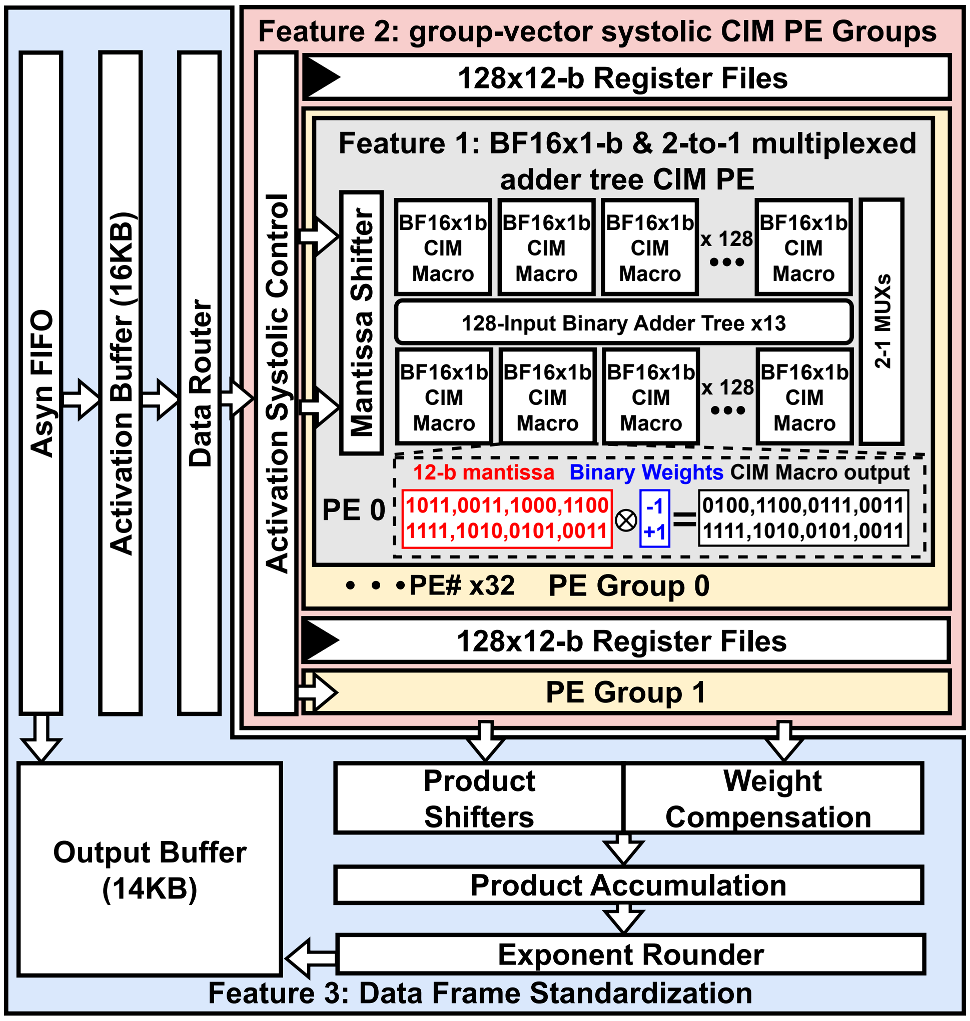


Figure 4.1: Proposed CIM accelerator with BF16x1-b CIM PE, group vector systolic CIM PE groups and data frame standardization for efficient inference of binarized BF16x1-b BERTs.

adder tree complexity (16). Furthermore, the large size of transformer models often exceeds the capacity of on-chip memory arrays, especially SRAM-based ones. This necessitates off-chip memory access for weights, leading to significant performance bottlenecks, particularly when frequent weight updates are required.

This work significantly enhances energy efficiency and area efficiency without sacrificing accuracy through a two-pronged approach. First, it employs ternary weight splitting (TWS) binarization on BERT models, achieving a 24× compression rate compared to FP32 models while minimizing quantization error. Second, we introduce a 2KB digital-domain CIM accelerator capable of performing batch-wise Brain-Floating-Point-16xINT1 (BF16×1-b) for activation×weight matrix multiplications for better accuracy. The CIM accelerator features BF16×1-b processing elements (PEs) with 2-to-1 multiplexed adder trees, group-vector systolic PE groups, standardized data frames, as illustrated in Fig. 4.1. Our contributions can be summarized as follows:

- We employ TWS to binarize BERT models into a BF16×1-b data format, achieving higher accuracy compared to INT4/INT8×1-b binarization at the same compression rate.
- We propose a BF16×1-b SRAM-based CIM macro to support floating-point operations with high parallelism. The proposed batch-wise mantissa shifter in CIM PEs and exponent rounder in CIM PE groups reduce computational error by two orders of magnitude while incurring only around 5% area and power overhead.

In this work, we further integrate a group vector systolic array with the CIM architecture, striking a balance between throughput and register dynamic power. A data frame standardization strategy is employed to eliminate the overhead associated with tensor reshaping and facilitate systolic data flow. The proposed design is verified with post-layout results in 28nm technology. The results demonstrate a 10.25× improvement in area efficiency and a 2.23× improvement in energy efficiency compared to state-of-the-art counterparts.

4.2 Related Work

4.2.1 Transformer Binarization Challenges for Edge AI

To deploy neural networks on edge devices, quantization is a common technique that replaces high-precision weights with lower-precision weights without altering the network architecture. This approach often involves using 2-, 4-, or 8-bit integers (INT2/4/8) instead of 32-bit floating points (FP32). Previous studies have demonstrated the effectiveness of INT8 (81; 82) or even INT2 (78) quantization in reducing the memory footprint of activations and weights without significant accuracy loss.

Due to the need for higher precision in certain operations like GLUE, Softmax, and Layer Normalization, full model quantization could not be realized (81; 82). This inconsistent precision configuration can introduce additional hardware design complexities and overhead. While (78) employs both approximation-based and distillation loss-aware ternarization to achieve INT2 BERT, these methods have not fully realized a 32x compression ratio or successfully replaced most floating-point multiplications with integer summations, as commonly found in binarized models.

Other model compression methods, such as neural architecture search (NAS), often seek to automatically discover optimal neural architectures within a given design space, balancing complexity and accuracy. While NAS has shown promise in optimizing CNN architectures (51; 53), its application to larger models can be computationally intensive due to the need for partial or complete training of sub-networks, requiring significant memory and storage resources.

Designing binarized neural networks faces one challenge: complex and irregular loss landscape of weight binarization. As for most weights of BERT concentrate on zero and increase randomness of binarization which generate unexpected quantization loss and render performance degradation. This work employs ternary weight splitting and use a half-sized ternary network as backbone to obtain binarized model with balanced area/energy efficiency and accuracy for CIM-based edge devices.

4.2.2 Parallel Computing Challenges in Digital CIM Architecture

Analog-domain CIM architectures, leveraging Kirchhoff's law for natural one-step vector-matrix multiplication, can potentially boost system throughput. However, current-based CIM approaches often struggle with device non-linearity issues, limiting their accuracy when dealing with large numbers of simultaneously activated rows ($<$128) (80). Charge-based CIM architectures, which employ foundry-provided capacitors, offer enhanced robustness against device non-linearity and enable higher parallelism, reaching up to 2000 rows (83). While charge-domain CIM architectures can improve parallelism, the use of capacitors and extensive wiring can introduce parasitic effects that may degrade output signal margin and compromise accuracy. Moreover, maintaining accuracy in high-parallelism charge-domain CIM architectures often requires higher ADC resolution, introducing additional overhead.

While digital-domain CIM architectures are immune to device non-linearity issues, they often face challenges in achieving high parallelism due to the overhead associated with adder trees. To mitigate this, various approaches have been proposed, including 8-to-1 multiplexed adder trees (84), modified 12T full adders (85), and low-resolution ADC-based partial sum compression (73). However, these methods may introduce trade-offs in terms of signal multiplexing overhead, approximation errors, or accuracy. To address these limitations, this work proposes a novel bit-parallel CIM PE featuring a 2-to-1 multiplexed customized adder tree.

4.2.3 Attention Computing Challenges on CIM Structure

As a core component of transformers, attention mechanisms compute dependencies between all search tokens through the calculation of $Atten = Q \cdot K^T$ and $Out = f(Atten) \cdot V$. Unlike convolutional operations or general matrix multiplication, the weight parameters in attention mechanisms are pre-determined, allowing for pre-loading in CIM architectures. However, the *query*(Q), *key*(K) and *value*(V) attention matrices are computed intermediate result, necessitating weight updates in CIM and potentially increasing off-chip data transaction power.

To address these challenges, (86) proposes pipeline/parallel CIM architecture that enable *atten* intermediate data stream to flow through various CIM engines without requiring off-chip memory loading or fetching. While this pipeline strategy can be effective on certain transformer-models, it may be interrupted when layer-wise parameters exceed the CIM array capacity, leading to weight update bubbles in the computation. Moreover, to adapt parallel computing in CIM array, matrix reshaping is requires for K and V matrix conversion from activation-shape to weight-shape. This can be particularly complex for high-dimensional multi-head QKV matrices.

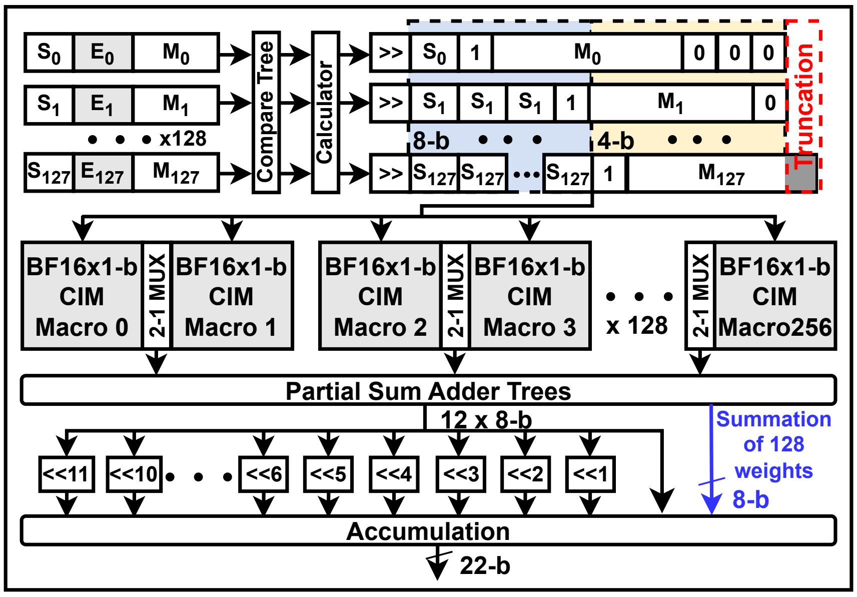


Figure 4.2: Proposed BF16x1-b CIM PE with 2-to-1 multiplexed adders.

4.3 BF16x1-b CIM PE

As shown in Figure 4.2, the proposed CIM PE contains 256 BF16×1-b CIM macros, organized in two rows for enhanced throughput. To reduce summation and multiplexing cost, 128 2-to-1 MUXs and 13 128-b adder trees are included. A batch-wise mantissa shifter with an extra 4-b shifting space is integrated to enable batch-wise exponent pre-alignment and improve computational accuracy.

The key innovation in our PE design lies in the synergistic combination of bit-parallel computation and efficient data movement. The 256 BF16×1-b CIM macros are carefully organized to maximize throughput while minimizing routing complexity. The 2-to-1 multiplexed adder tree structure represents an optimal trade-off between computational capability and hardware overhead, enabling efficient processing of both dense and sparse operations common in transformer architectures.

4.3.1 Bit-parallel CIM Macro Design

The proposed CIM macro incorporates a standard 6T SRAM for unsigned 1-bit weight storage and 12 4T-XORs for shifted mantissa with signed, as shown in Figure 4.3 (a), and forming a 58T SRAM-based cell. By sharing 1-bit weights and du-

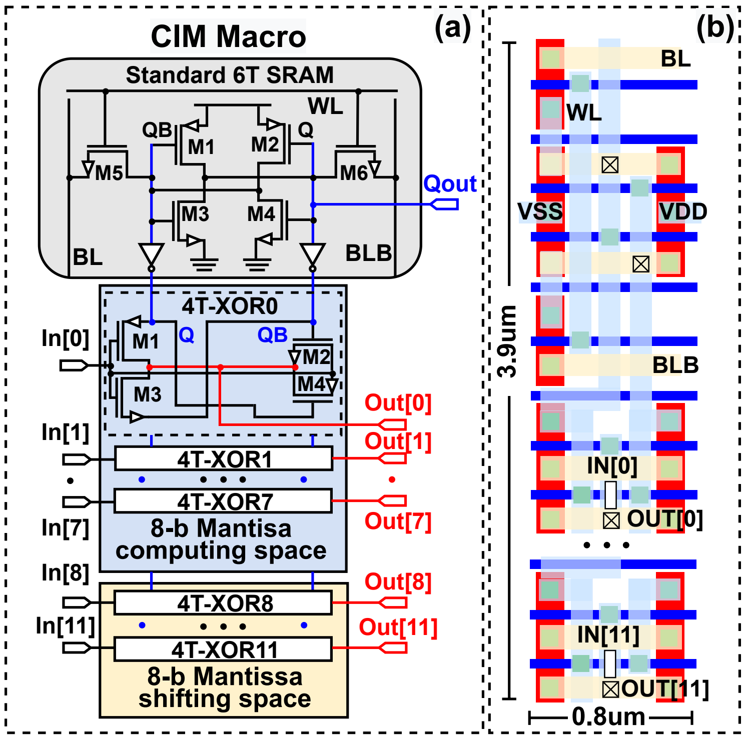


Figure 4.3: Proposed bit-parallel CIM macro. (a). Schematic; (b). Physical Layout.

plication of XOR units, bit-parallel computation paradigm could be realized which reduces the cycle count from 12 to 1 compared to bit-serial methods. As shown in Figure 4.3 (b), the macro has a size of $3.12\mu m^2$. Compared to conventional SRAM CIM macro with two transmission gate composed XOR gate(16), this macro achieves a 12x throughput improvement with only a 8.38x area overhead. The first eight 4T-XORs compose the original computing space, while the final four 4T-XORs implement the mantissa shifting space for BF16 activations, reducing the computing error (standard deviation) by 2-order of magnitude.

Figure 4.4 illustrates the timing diagram for weight loading and BF16 $\times$ INT1 operations. Pre-loaded weights are reused for a specified number of CH_{in} cycles before the next round of computation. The WL signal is clock-gated to ensure hold-up check compliance in static timing analysis. In each cycle, the 12-bit mantissa of the BF16 activation is simultaneously sent to the proposed CIM macro for bitwise XOR operations with the binary weight. The 12-bit output is generated in parallel during the same capture cycle as the corresponding inputs. Unlike the BLB-based MAC operations in (45), the proposed CIM macro separates the SRAM weight and XOR logic control signals, enabling simultaneous writing and operation. Notably, input

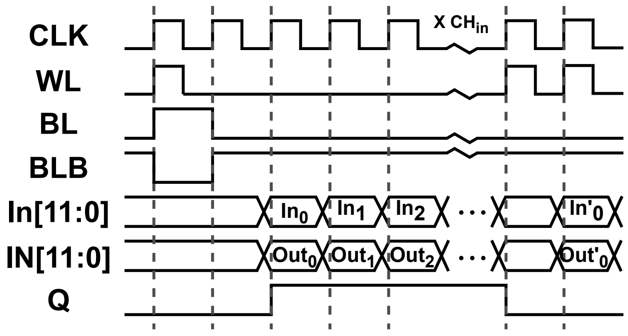


Figure 4.4: Transient time diagram of proposed CIM macro.

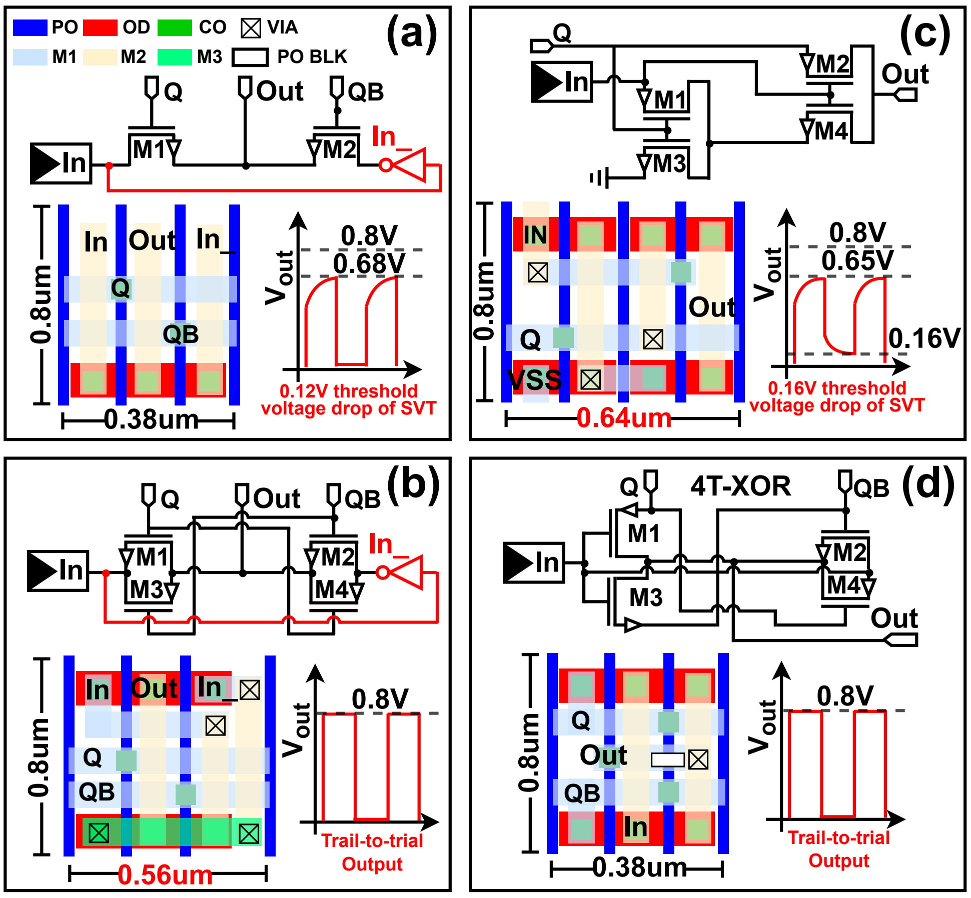


Figure 4.5: Comparisons between XOR units. (a). Typical 2T XOR structure; (b). Typical 4T XOR structure with 2 transmission gates; (c). 4T XOR structure with non-inverted input; (d). Proposed 4T-XOR unit.

activations have a total bit-width of 12 bits, the rationale behind this configuration will be discussed in Section 4.3.2.

Previous CIM studies for CNNs have often overlooked the critical role of prerequisite peripherals in CIM architecture. This is primarily because convolution operations exhibit a higher weight reuse rate than regular matrix-matrix (MM) multiplications. In convolution, a single activation can be shared by multiple weights on the same row, requiring only one activation register or prerequisite peripheral for that row. However, in regular MM, activations and weights have a one-to-one correspondence, necessitating an equal number of activation registers or prerequisite peripherals as weights.

As illustrated in Figure 4.5 (a) and (c), traditional 2T/4T XOR circuits commonly used in CIM architectures require additional inverters for XOR operations, leading to significant signal inversion overhead in large-scale MM operations. This work comprehensively compares the output signal quality and area of different XOR circuit implementations. For area comparison, all XOR units are constrained to a fixed height of 0.8um to enable automatic place and route, and the area associated with inverters is excluded. In output signal quality comparison, the power supply is maintained at 0.8V under typical corner conditions.

The 2T (NMOS) XOR circuit shown in Figure 4.5 (a) is limited by design rules and the threshold voltage drop of the NMOS switch, resulting in low area utilization and compromised signal margin. Although the typical 4T XOR circuit in Figure 4.5 (b) offers a trail-to-trail output, it requires additional routing area due to the complex connectivity of its transmission gates. The 4T XOR unit depicted in Figure 4.5 (c) utilizes a single pair of input and Q signals. However, the connection relationship between the drain and source of PMOS transistors M1 and M2 prevents their merging, leading to increased area. Additionally, the use of either PMOS or NMOS for switching introduces an inevitable threshold voltage drop. The proposed 4T-XOR circuit, shown in Figure 4.5, addresses these limitations by leveraging the inverted QB signal generated from the dynamic latch of the SRAM to compensate for threshold voltage drop while maximizing the utilization of the available design space.

4.3.2 Batch-wise Mantissa Shifter

Since the emergence of Brain Floating Point 16 (BF16) data format (87), it has been widely employed to reduce memory footprint of weights and activations on

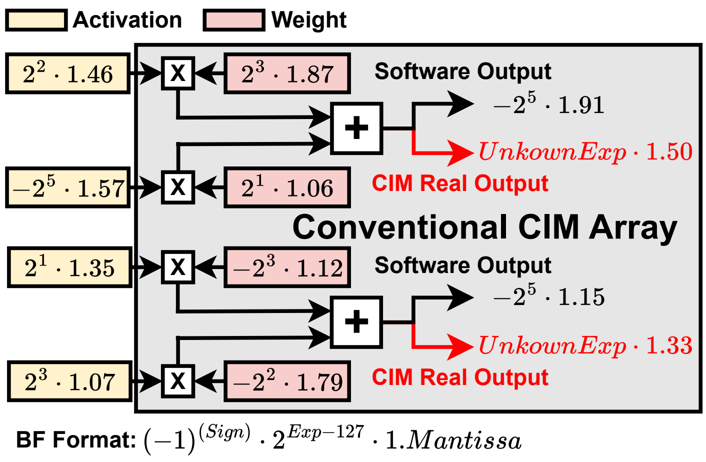


Figure 4.6: Alignment challenge of BF-based MAC operations in conventional CIM architecture.

not only clusters for large scale distributed parallel training procedure(88) but also on edges for low power deep neural network acceleration (89). Owing to its large dynamic range (equivalent to FP32) and shorter representation in fraction, BF16 offers a better choice for CIM implementation (45; 90; 91) than full or half precision for accuracy and storage capacity concern. However, computation of floating point operations imposes several additional questions for CIM architecture. First, current CIM architecture could only handle INT-based MAC operation under scaling factor and could not directly dealt with FP-based MAC operation with different exponents, as shown in Figure 4.6. Second, pre-alignment of exponent and corresponding mantissa shifting would introduce inevitable rounding error and degrades performance, shown in Figure 4.7 (a). Moreover one needs to consider the trade-off between storage capacity and computational accuracy (91). Third, previous CIM works (45) deploy layer-wise exponent pre-alignment strategies and fail to support fine-grain exponent pre-alignment thus impair system accuracy.

To address these challenges, this work employs exponent pre-alignment and mantissa shifter to conduct MAC operation for FP-based data format in CIM architecture. As shown in Figure 4.7 (b), proposed MAC macro enable extra 4-b space for mantissa shifting which could mitigate data rounding related computational error. Additionally, this work employs fine-grained batch-wise pre-alignment strategy to further data truncation error when layer-wise weight discrepancy is too large. As shown in

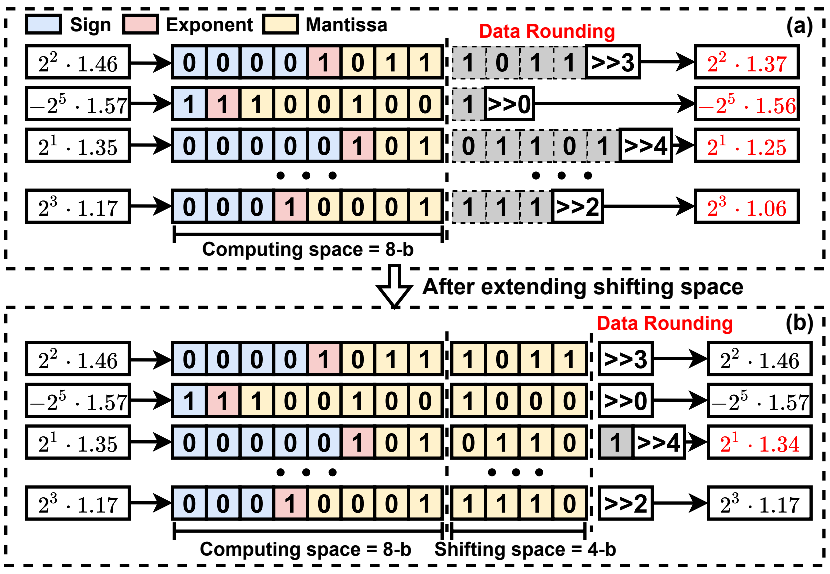


Figure 4.7: Comparison between floating-point-based CIM architectures. (a). CIM architecture with 8-b mantissa computing space; (b). Proposed CIM architecture with extra 4-b mantissa shifting space.

Figure 4.8, the proposed mantissa shifter has three procedures.

Step 1: A batch of 128 activations will be fetch via activation buffer to find the maximum exponent of E_{max}. Once E_{max} is obtained, activation will be shifted according to the different between E_{max} and E_i, which could be formulated as:

$$\begin{aligned} Activation &= M \cdot A_E, \\ M &= (-1)^{sign} \cdot 1.M_i \times (\frac{1}{2})^{E_{max}-E_i}, \\ A_E &= E_{max} - 127, \\ E_{max} &= argmax[E_i], i \in [1, 2..., 128]. \end{aligned} \tag{4.1}$$

where M is the off-line pre-encoded mantissa after shifting, which will sent back to activation buffer for subsequent computation in CIM macros. A_E is the maximum exponential value among 128 activations. M_i and E_i represents the corresponding mantissa and exponent of a specific activation in BF16 format. Notably, as shown in Figure 4.7 (b), an additional bit is added in between the sign bit and the most significant bit (MSB) to represent if the exponent is equal to zero or not.

Setp 2: Shifted mantissa A_M will be fed into CIM macro in parallel to conduct XOR with shared binary weight. The CIM macro then generates partial sum that

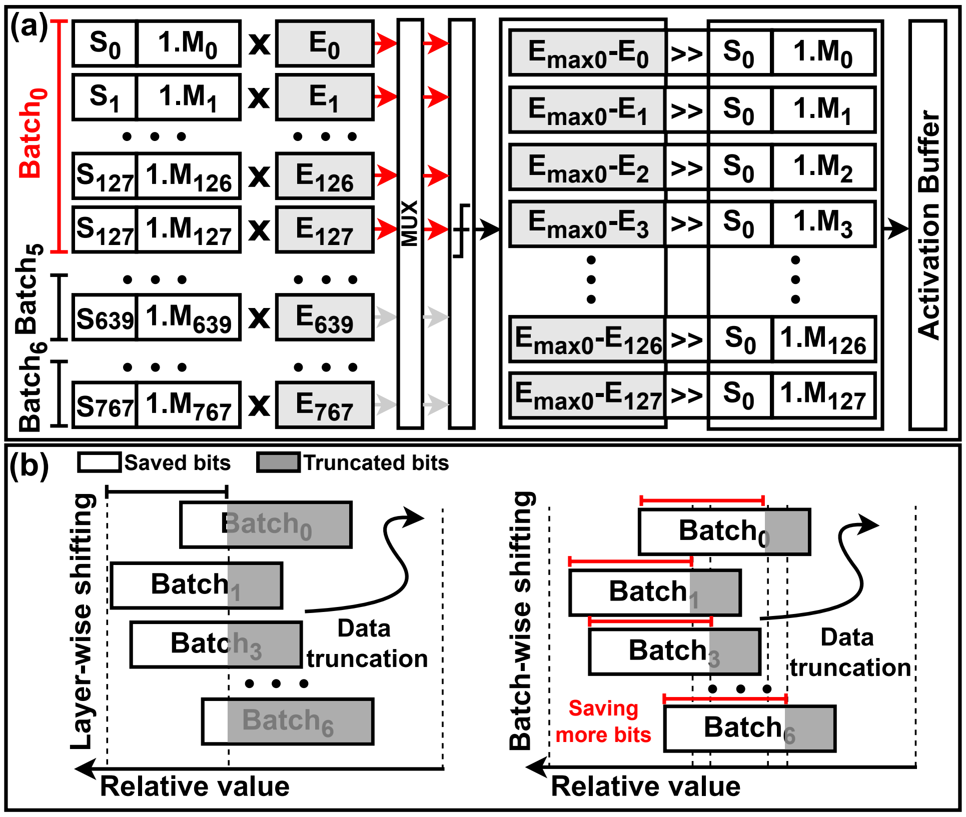


Figure 4.8: Proposed Mantissa shifter design for activation in BF16 format.

could be formulated as:

$$P = \sum_{i}^{12} A_i \oplus W + W \tag{4.2}$$

where W is the binary weight. The additional summation with W is indeed required to generate expected result, which explains the reason why proposed CIM macro has a output Q. Then the partial sum P will be fed to subsequent bit-wise adder tree and final shift & add module to generate the MM results. Finally, the obtained MM results will be combined with A_E to produce the final output value in BF16 format.

Setp 3: The obtained E_{max} of different batches will be stored in activation buffer as well as the shifted mantissa for subsequent batch-wise exponent alignment in exponent rounder to generate final results.

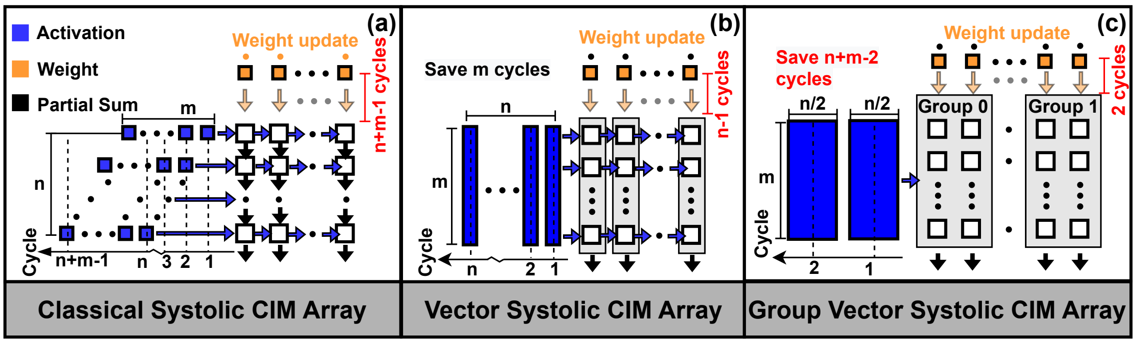


Figure 4.9: Comparison of different systolic array design in CIM.(a). Classical atomic systolic CIM array; (b). Vector systolic CIM arrray; (c). Proposed group vector systolic CIM array.

4.4 Group-Vector-Systolic CIM Array

Matrix-matrix multiplication and convolution dominate the computational workload in neural networks, imposing significant bandwidth demands on CIM architectures, particularly for parallelism-oriented designs. Previous CIM works (16; 80) often employ broadcast computing diagrams to handle large-scale convolution or MM tasks, which, while offering advantages in terms of small signal transition latency and area overhead, suffer from large fan-out and poor timing due to extensive propagation delays. Systolic array design (92) is a valuable technique for minimizing data fan-in/out requirements. In 2017, Google (93) introduced its renowned neural network processor TPU, which leverages a fine-grained, atomic systolic data-flow at the macro level.To boost throughput, CIM-based implementations have adopted similar atomic systolic strategies to increase system frequency, as depicted in Figure 4.9 (a). However, this approach necessitates a substantial number of register files adjacent to macros for intermediate data storage, leading to significant dynamic power and area overhead.

To mitigate this overhead increase, vector systolic arrays were proposed in (47; 49), which transfer data systolic operations from the macro level to the PE level in a row-by-row manner, resulting in a significant reduction in the number of required register files and associated overhead. In the context of CIM architectures, similar approaches can be found in (11), as illustrated in Figure 4.9 (b). Building upon this,

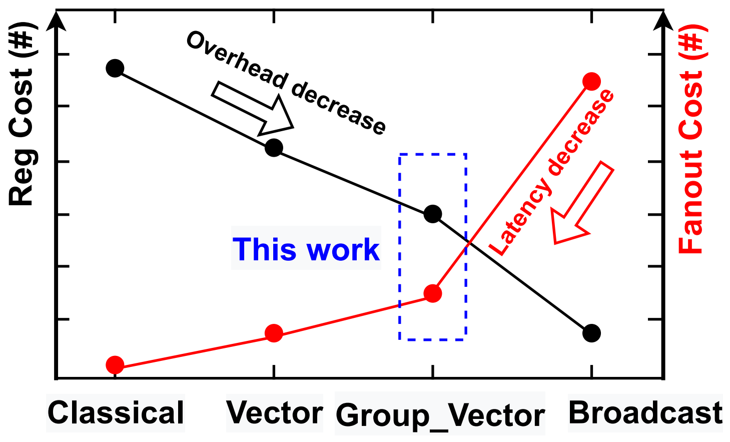


Figure 4.10: Register file (Area) and fan-out (latency) analysis on different systolic array.

our work deploys a group vector systolic design (94), elevating the data systolic level to the PE-group level in the CIM architecture, as shown in Figure 4.11 (c). The proposed group vector CIM array design further reduces the number of register files by enabling higher parallelism activation broadcasts within PE groups.

As depicted in Figure 4.9, the proposed group vector systolic CIM array requires only 2 cycles for weight updating, while the classic atomic systolic CIM array and vector systolic CIM array necessitate n+m-1 cycles and n-1 cycles, respectively, for an $m \times n$ matrix multiplication. Consequently, the proposed group vector systolic CIM array achieves a balanced trade-off between hardware efficiency and energy efficiency, as illustrated in Figure 4.10.

Figure 4.11 illustrates the detailed implementation of the proposed group vector systolic CIM array. This array comprises two PE groups, each containing 32 PEs. Each PE consists of a 2-to-1 multiplexed array of 256 BF16 $\times$ INT1 CIM macros. The 258 binary weights are pre-loaded onto each PE, and 128 12-bit activations are fed into each PE per cycle. Once the mantissa shifting procedure is completed and written back to the activation buffer, the PE groups initiate systolic computation. The accumulation unit within each PE concurrently generates a signed 20-bit partial sum. To reduce latency and area overhead, activations for PEs within a group are broadcast in parallel, while weights remain static until all activations have been processed. A 1-bit counter is employed to multiplex outputs from different groups.

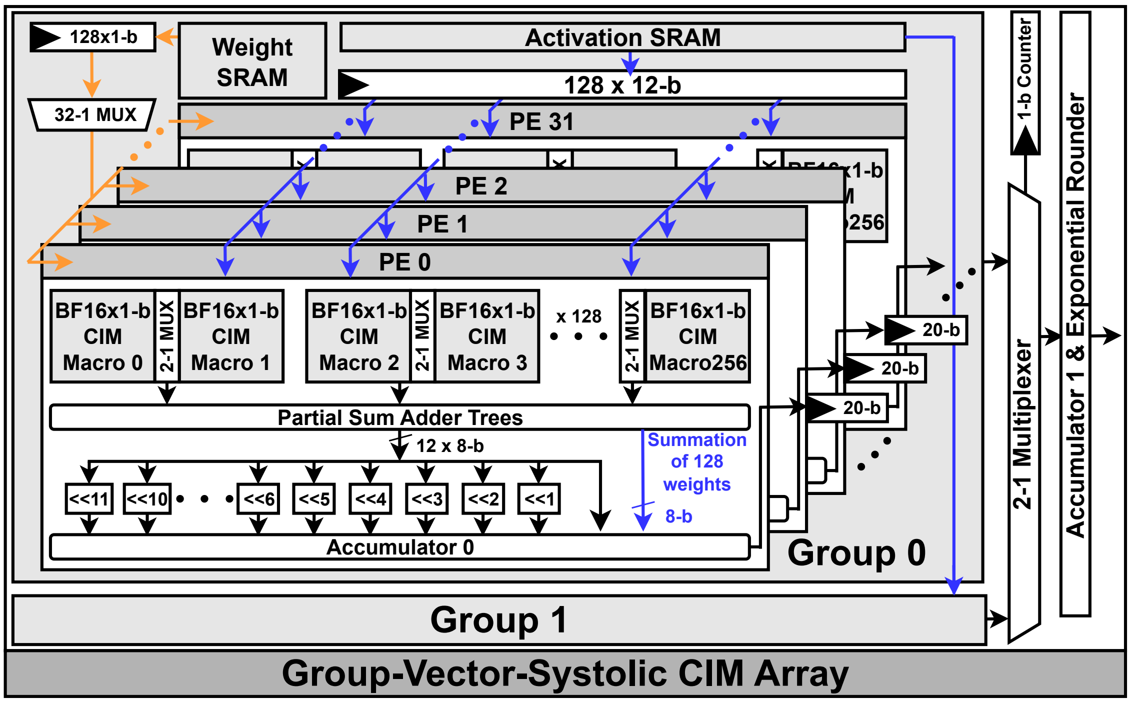


Figure 4.11: Proposed Group vector systolic CIM array design.

Accumulator then produces the final MM results, and the exponent rounder truncates the results to BF16 format.

4.5 Data Frame Standardization

Matrix operations in transformers have various tensor shapes, e.g., Matrix-multiplication, LayerNorm, Multi-head Self-Attention. General processing of these tensors requires extra temporary buffer for reshaping. This work proposes a standardized data frame for all kinds of tensors in transformer and facilitates group vector systolic. The fundamental principle combines adjacent output channels ($T_{in} = 128, T_{out} = 128$), as shown in Fig. 4.12. Specifically, the standardized data frame is a four-dimensional tensor, e.g., $((CH_{in} + T_{in} - 1)/T_{in}, H, W, T_{in})$. For low dimensional tensor like LayerNorm in Bert-Tiny with shape $(CH_{in}/T_{in}, Emb, T_{in}) = (1, 128, 128)$, the Emb is divided into H and W. For high dimensional tensor like multi-head self-attention (MHSA), the tensor shape is reorganized as $((CH_{in}+T_{in}-1)/T_{in}, Head\times H, W, T_{in})$. The benefits of data frame standardization are evident: as the outputs of all oper-

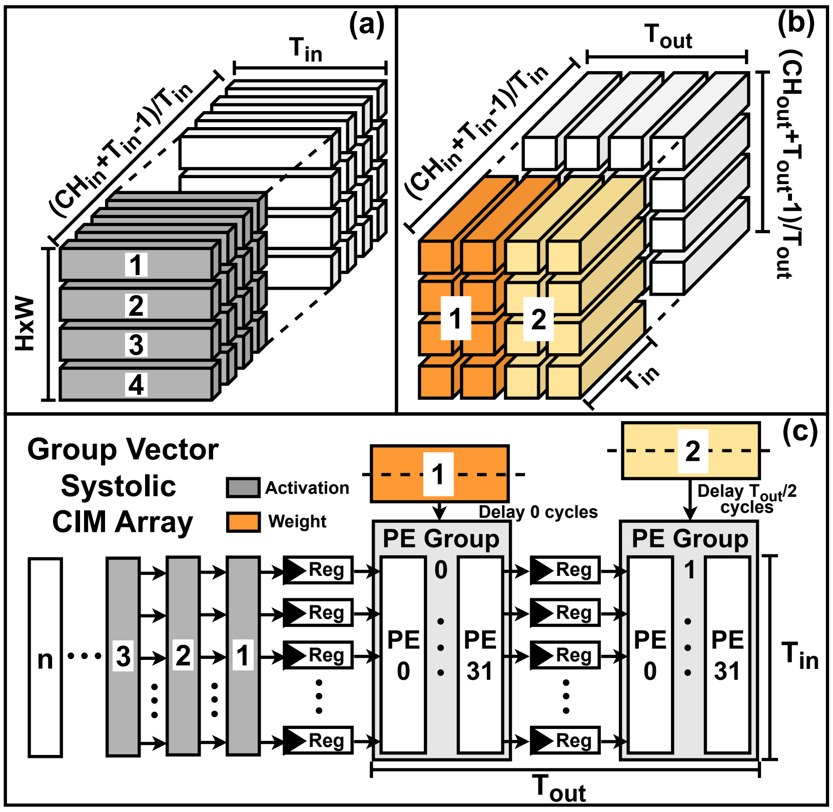


Figure 4.12: Neural network mapping of proposed digital CIM accelerator.

ations are equivalent to their inputs, the subsequent tensor alignment procedures commonly seen in other CIM implementations (45) can be omitted.

The proposed architecture demonstrates strong scalability characteristics. The group vector systolic design can be readily extended to accommodate larger model sizes by increasing the number of PE groups, while the standardized data frame structure ensures efficient handling of varied tensor operations regardless of scale. This flexibility makes our approach particularly suitable for future transformer variants with different attention mechanisms and model architectures.

The corresponding experiment results of weight binarized BERTs on proposed digital-domain SRAM-based CIM accelerator is organized in section 6.2.

Chapter 5

ReRAM-based CIM Accelerator

5.1 Introduction

The deployment of modern convolutional neural networks (CNN) on edge computing has imposed significant challenges on energy efficiency, especially when the networks are becoming more complicated with a larger number of parameters (64–66). Conventional CMOS accelerators in Von-Neumann architecture have to fetch data from cache memory according to programmed procedures for each calculation cycle, which hits the “memory wall” and “power wall” due to huge latency and power consumption (93; 95; 96).

To overcome these issues, computing-in-memory (CIM) architecture is proposed by shortening the gap between processing and storage, suppressing the amount of intermediate data (97). By constructing a 2-dimensional array, weight data is preloaded in the bit cells, which is multiplied by the input activation data on the horizontal word lines. Then the multiplication results are accumulated on the vertical bit lines for multiple parallel MAC operations. SRAM-based CIM works benefit from high processing speed and compatible processes. However, SRAM occupies a large area, and data is lost when power is cut off. (34; 83; 98). ReRAM-based CIM works are area-efficient and nonvolatile, which makes ReRAM array in CIM architecture extremely suitable for edge-computing applications (29; 44; 99–102). However, the performance of ReRAM-based CIM accelerators is still limited by conventional CNN

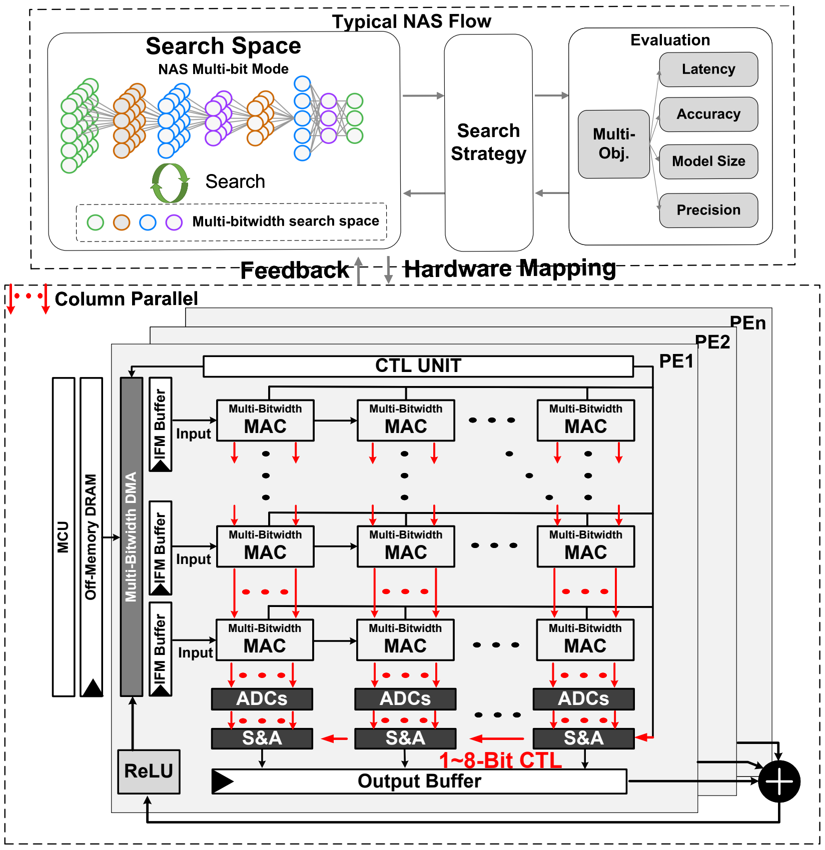


Figure 5.1: Proposed ReRAM-based CIM accelerator for Mixed-bit CNNs.

architectures and inefficient readouts (6; 103).

For energy-efficient performance improvement, an optimized CNN model is required and a low-power column parallel readout is necessary for edge-computing hardware. At the algorithm level, many methods such as pruning (68) and quantization (3) have been proposed to reduce the number of parameters. In order to keep the balance of accuracy and efficiency, layer-wise bitwidth quantization is necessary. However, when faced with CNNs, the search space of layer-wise bitwidth increases exponentially, which makes it infeasible to rely on hand-crafted strategies (64). Neural Architecture Search (NAS) method can be used to search the best layer-wise bitwidth CNN network automatically with minimized bits and acceptable accuracy (65; 104). At the hardware level, column serial readouts are often employed for CIM array due to less total hardware cost (105). However, these methods have to design additional MUXs with an asynchronous clock signal, which increases the design difficulties and hardware cost on average (80). Column Parallel readouts require large periph-

Table 5.1: NAS-optimized multi-bitwidth CNN benchmarks.

CNN Models	**Datasets**	**Weights (MB)**	**Bit Proportions (2bit/4bit/8bit)**	**accuracy**
VGG-16	ImageNet	71.6	20.3% /63.6%/16.1%	70.32 %
ResNet-18	ImageNet	5.1	26.2% /73.8%/0.00%	69.68 %

eral circuits, which demand energy-efficient circuit designs to reduce total power consumption. Besides, ReRAM nonidealities also affect the degree of parallelism, which lowers total throughput (106).

In this paper, a mixed-bit ReRAM-based CIM accelerator is proposed to improve energy-efficient performance for CNN inference on edge. In Figure 5.1, an effective bitwidth configuration scheme is employed to implement NAS-optimized mixed-bit CNNs for extremely high average energy efficiency. Column parallel readout is achieved with excellent energy-efficient performance by a variation-reduction accumulation mechanism and low-power readout circuits. The main contributions are summarized as follows:

Mixed-bit ReRAM-based accelerator for NAS-optimized CNNs. The proposed ReRAM-based accelerator supports 1-bit∼8-bit signed MAC operations for implementing NAS-optimized mixed-bit CNNs. By using serial activation inputs and parallel weights, the best average energy efficiency of 479.37 TOPS/W can be achieved on the mixed-bit ResNet-18 benchmark.

Column parallel readout with low-power cycle-reduced ADC. To improve energy-efficient performance further, column-parallel readout is proposed to enlarge the total throughput. Based on low-power cycle-reduced ADC, the power consumption of peripheral readout circuits is optimized, especially for CNN with sparsity characteristics. Peak energy efficiency of 2490.32 TOPS/W and 38.91 TOPS/W can be achieved for 1-bit and 8-bit operations, respectively.

5.2 Related Work

5.2.1 NAS-Optimized Mixed-Bit CNNs

As shown in Figure 5.1, a typical NAS flow results in that the bitwidth of each CNN layer is optimized by NAS algorithm based on search strategies and evaluation criteria (107). Such an optimized mixed-bit network model can then be used for inference on edge computing in order to balance accuracy and energy efficiency.

Modern CNNs deploy high precision floating point (FP) data formats in pursuit of high accuracy. However, computing models in such precisions bring along large power consumption and area. Previous works (39; 108) have proved that sizes and bitwidths of model parameters can be compressed with tolerable accuracy degradation by using model quantization. However, the ever-growing depth of CNNs makes the traditional model quantization infeasible for manual layer-wise mixed-bit configuration. Therefore, some NAS algorithms are proposed to incorporate model quantization and automatically search optimal neural architectures and layer bitwidths. The recent advances in NAS (64; 65; 104) have shown great potential for automatic optimization on CNN networks with large data and computing reduction, which balances the performance between network accuracy and hardware efficiency. Prior works have optimized DNNs to 1∼8 various bitwidths each layer with little loss in classification accuracy (109–111). The models, datasets, weight amounts, and bit proportions are summarized in Table I. To take advantage of NAS-optimized CNNs, some CMOS accelerators have been proposed for hardware implementation with excellent average energy efficiencies (111; 112). However, the CIM designs targeted for NAS-optimzed CNNs haven't been provided.

5.2.2 ReRAM CIM Accelerator Readout

Theoretically, the intrinsic advantage of ReRAM-based CIM accelerator with ReRAM array is parallel computing, which has the potential to achieve better energy-efficient performance than conventional row-wise/column-wise computing CIM works (97; 113). However, the degree of parallelism is largely affected by ReRAM nonlinearities and readout costs.

In terms of device variation, the standard deviation of ReRAM resistances is at least 10%, which is more than ten times the variance of conventional resistors and capacitors (114). Thus, the direct accumulation in the analog domain suffers from the ReRAM variances and results in less effective accuracy (115). Besides, the small resistance ratio (R ratio) between the ReRAM resistances at the high- (HRS) and low-resistive state (LRS) results in narrow signal margin. To clearly determine HRS and LRS state numbers, less than 20 MACs can be activated simultaneously for analog computing in one column (106; 116). The aforementioned nonlinearities reduce the activation per column. (103; 117).

In current-accumulation structures, computed MAC results would increase linearly with the LRS ReRAMs switching on. Thus, a large accumulated current generated from high column parallel results in large dynamic range for quantization, which imposes huge burdens on readout interface circuits, especially operational amplifier (OPAMP) and ADC. The power consumption of these circuits will account for more than 70% of the entire system (118).

Due to the large overhead of signal amplification and analog/digital conversion, the current from each column is not fed into this aforementioned readout circuit directly (103; 113). Instead, to reduce the overhead of ADC, these currents are initially either stored in sample and hold module or generated in sequence and then selected to be fed into readout circuit by a multiplexer in certain serial order (80; 113). However, such column order will not render any improvement on energy efficiency and instead degrade energy efficiency. Since the throughput is decreasing linearly with the reduction of ADC overhead, extra control modules are required by this column serial design, which causes an increment of total power.

To stabilize the voltage of BL, an OPAMP is commonly used at the end of each column of the ReRAM array, whose power consumption accounts for 85% of the overall readout circuits (118). To reduce OPAMP power, (119) developed a voltage clamp circuit to stabilize the voltage of BL with greatly reduced power consumption. In terms of ADC optimization, (103) uses an ADC with adjustable precision, but its ADC power consumption is still more than half of the overall system. (61) focuses

Table 5.2: Parameters of proposed ReRAM-based CIM accelerator

Parameters	Values	Description
u	1∼8	Bit mode
m	1	Row numbers of Macro cells in MAC
n	8	Column numbers of Macro cells in MAC
p	2048	Row numbers of MACs in PE
q	256	Column numbers of MACs in PE
K	3	Rows or column numbers of PEs
H	-	IFM width/height
C	-	IFM channel numbers
E	-	OFM width/height
M	-	3D filter numbers

on reducing processing necessaries by turning off certain ADCs to save power. To save total hardware costs, serial readout format has been employed with spatial-multiplexed circuits (80; 105). However, this method has to design additional MUXs with an asynchronous clock signal, which increases the design difficulties and cost on average.

5.3 Mixed-bit CIM Accelerator

5.3.1 Mixed-bit MAC

Figure 5.2 shows the block diagram of mixed-bit CIM system and 4-bit operations with corresponding input sequences. The figure shows the detailed structure of one MAC cell, which contains 8 ReRAM macro cells, one shared word line, and 8 consecutive bit lines. For each Macro cell, 1-bit *AND* operation is implemented. To enable accurate and flexible mixed-bit operation of 2.21, each macro needs to behave as an AND gate in digital circuit design.

For every 8-bit weight, each bit is stored in each macro cell of MAC cell from MSB bit to LSB bit. Meanwhile, the input bits are split and passed on the same WL one by one during each clock cycle and the columns will generate their corresponding bit-

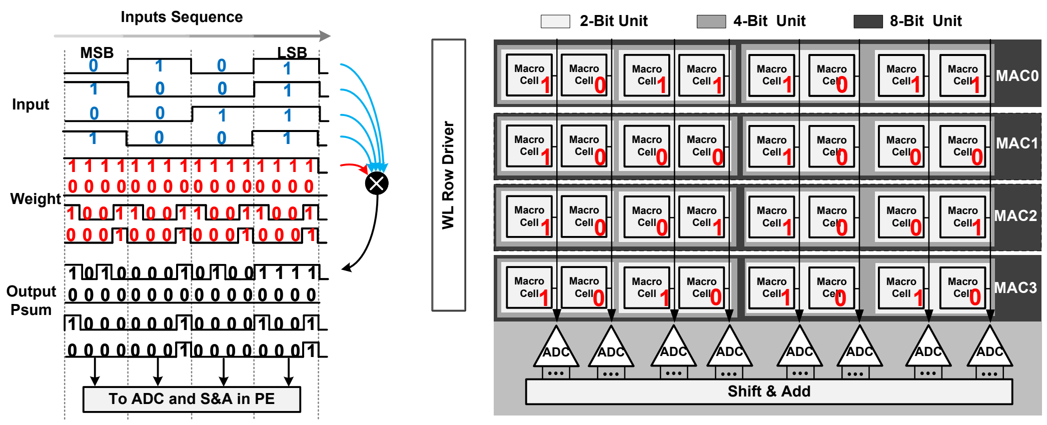


Figure 5.2: Mixed-bit MACs for 2-bit, 4-bit and 8-bit operations with corresponding input sequences.

Table 5.3: The True Table of Proposed Macro Cell

Input Feature Maps & Weights				Output
IFM	W	R_u	R_d	V_c
0	0	LRS	HRS	0
0	1	HRS	LRS	0
1	0	LRS	HRS	0
1	1	HRS	LRS	1

significance-wise analog partial sum in parallel. Thus, each MAC cell can implement one 8-bit× 1-bit operation in one clock cycle and an 8-bit× 8-bit operation in eight clock cycles. The MAC cell can also implement two 4-bit multiplication operations and four 2-bit multiplication operations by dividing the macro cells into groups with different compute cycles. Other operational bits like 3-bit, 5-bit and 6-bit could also be stored on mixed-bit MAC, but with certain degrees of utilization degradation. In the proposed work, 2's complement data format is used for both input feature maps (IFMs) and weights. The mixed-bit MAC cell operates purely in analog domain and the analog/digital converter as well as the precision reconfiguration module are introduced in the next hierarchical level of mixed-bit processing element (PE) in Section 5.3.2.

In the previous discussion, the non-linearity issue confines the activation per col-

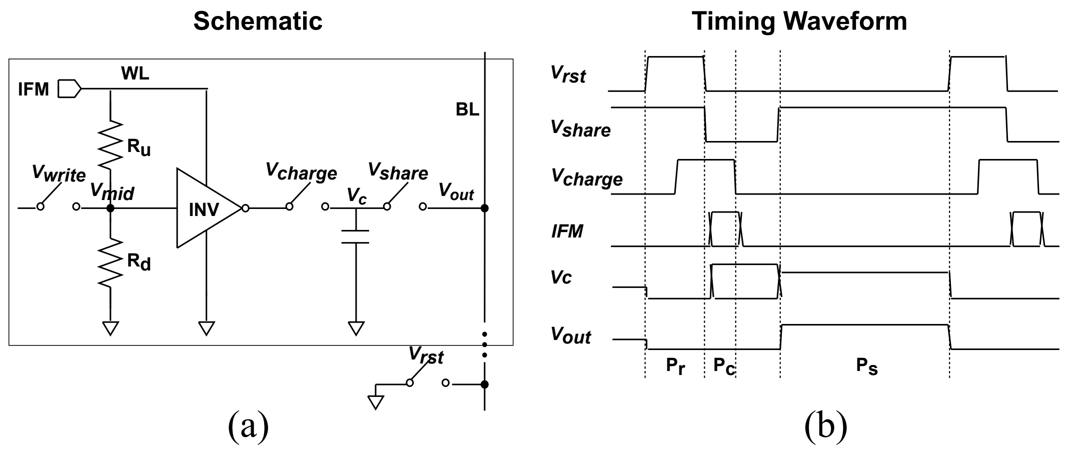


Figure 5.3: Macro cell in proposed ReRAM-based CIM accelerator: (a) Macro cell schematic and (b) timing waveform.

umn of current-accumulation ReRAM array. To reduce the ReRAM variation drawbacks and increase computing column parallel activation percentage, a charged-based macro cell is proposed. Within each cell, as shown in Figure 5.3, two ReRAM devices are connected in series in the first stage, which is used to store the weight signal. Then the voltage V_{mid} on the intermediate node is sent to an inverter which is used for variation isolation. The output is determined by V_{mid} and generated with rail-to-rail voltages, which will charge the following capacitor. Thus we isolate the side effects of ReRAM variance via voltage division and inversion, while only taking the variance of a small capacitor into consideration. Additionally, due to the voltage division-based functionality, the proposed charge-based macro could isolate the most impact of IR-drop commonly seen in 1T1R macro. An ReRAM of 1M/100K for HRS/LRS has a standard deviation of at least 10% (6; 114) and a capacitor of 1.3-fF in 28-nm process has normalized standard deviation of 0.8% in the PDK file.

As shown in Figure 5.3 (a) the working state of an inverter is determined by *IFM* value on wordline (*WL*) and middle voltage (*Vmid*) of serial-connected ReRAMs. If *IFM* equals 1, the inverter is enabled to reverse the *IFM* value and determines V_C on the top plate of the capacitor. Meanwhile, if weight equals 1, the upper ReRAM and lower ReRAM is set in HRS and LRS with R_{high} and R_{low} resistance, respectively. Due to voltage division, *Vmid* equals $R_{low}/(R_{high}+R_{low})\times V_{dd}$, which is within the pull-up region of a inverter. Then the inverter generates near V_{dd}

output and charges the capacitor. On the contrary, if the weight equals 0, the upper ReRAM and lower ReRAM is set to in LRS and HRS, respectively. The inverter generates near 0 output and discharges the capacitor. If *IFM* equals 0, the inverter is disabled and V_C keeps reset voltage regardless of weight values. In this way, the macro cell can realize *AND* function of *IFM* and *W*. The IFM and weight input combinations with corresponding output results are summarized in Table III.

The timing waveform of the macro cell is also provided in Figure 5.3 (b). The reset signal V_{rst} is used to connect BL to reset voltage 0. The signals V_{charge} and V_{share} are used to change the capacitor voltage V_C and share the charges with all the capacitors, respectively. As shown in the waveform, the capacitor is discharged to zero during the reset phase P_r. Then the capacitor voltage V_C is updated with *AND* result of *IFM* and *W* during the charging phase P_c. The capacitors are connected together with BL to generate output V_{out} and get quantized digital output during the sharing phase P_s. The switch controlled by V_{write} is employed to provide certain bias on V_{mid} and enable the Set/Reset ReRAM process.

5.3.2 Mixed-bit Operation

By optimizing by NAS algorithms (64; 104), the required CNN parameters can be greatly compressed to energy-saving low-precision formats, e.g., 2 bits and 4 bits. As such, NN models can be implemented on the CIM accelerators in one shot with fewer storage cells. However, NAS-optimized neural networks pose additional precision configuration challenges to ReRAM-based CIM accelerators. Herein, we present the corresponding mixed-bit operations in the proposed work from macro cell level to PE level.

The proposed mixed-bit PE is composed by $p \times q$ MAC cells with $q \times n$ ADCs as well as q shift-and-add (S&A) modules and supports various operational bits, namely 1∼ 8-bits. Before computing, the U-bit weights are mapped onto U consecutive charge-based macro cell of one MAC cell according to bit-significance. Compared with the rigid bit CIM design (120), (80), proposed mixed-bit MAC can flexibly select its operational mode in 1∼8 various bits. During the computing process, the IFMs are fed into the MAC array in serial, which will also be split into a

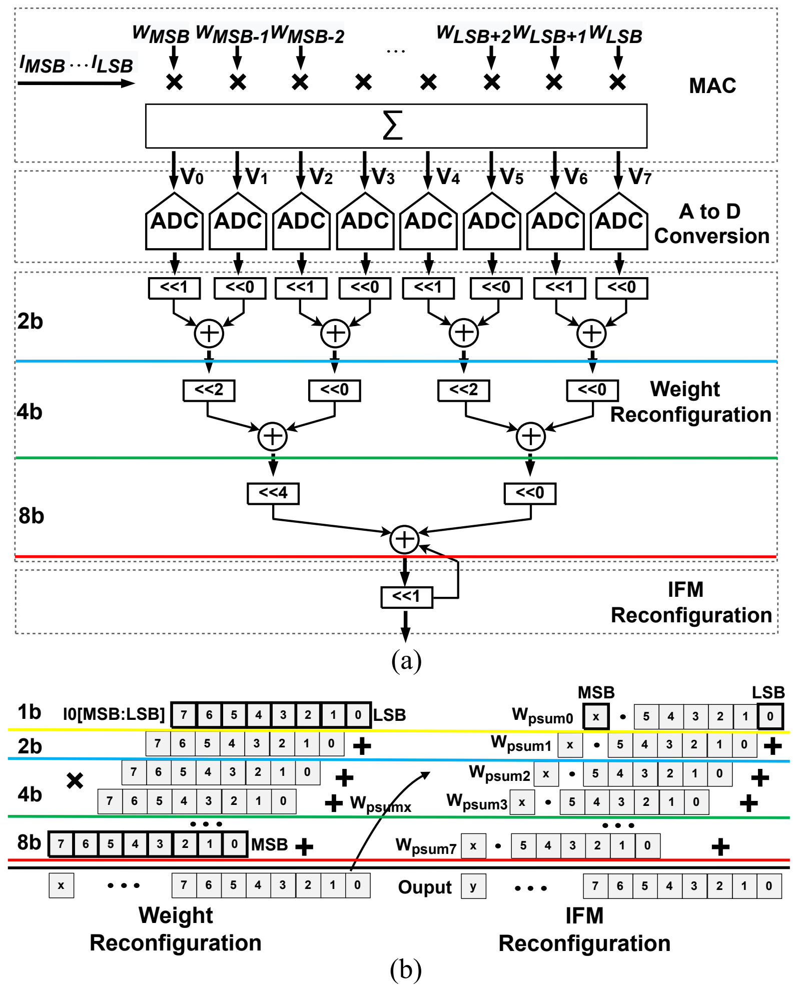


Figure 5.4: Mixed-bit operation method of the proposed system: (a) structure of mixed-bit PE and (b) mixed-bit dataflow.

bit-significance type in the same way as the weights. When computing, the WL controller generates a batch of voltages to the whole array. All of the input bits will then conduct "AND" operations with the embedded weight bits column-by-column via voltage division, and be summed up according to charge redistribution. After computing, each column of MACs will generate a (p-1)-state analog partial sum of the same weight bit-significance, which will be sent directly into subsequent readout peripheral and S&A module for precision configuration.

For proposed MAC, fixed column number n is 8. For weight bitwidths with lowest common multiple equal to 8, the MAC generates 8 un-configured multiplication-accumulation partial sums from 8, 4, 2, 1 pairs of 1, 2, 4, 8-bit weights and 1-bit IFMs each cycle, respectively. The accumulation cycle is determined by the IFM bitwidths. For these bitwidths, 100% utilization of MAC is achieved. However, the utilization performance is degraded when weights with unusual bitwidths are used. For other weight bitwidths with lowest common multiple not equal to 8, the MAC cannot be fully used. The MAC generates the partial sums from 2, 1, 1, 1 pairs of 3, 5, 6, 7-bit weights and 1-bit IFMs each cycle, respectively. The corresponding bit vacancies in each MAC are 2, 3, 2, 1, which results in utilization performance degradation with 12.5% 37.5% in the CIM array.

Figure 5.4 (a) shows the structure of column parallel readout and S&A module of one column of MAC cell inside mixed-bit PE. The digitized partial sum of MAC cells is fed into 4 pairs of 2-bit S&A module initially, then based upon the selection of operation bits, the partial sum will be selectively fed into the subsequent 4-bit and 8-bit S&A module. For 1-bit operations, the partial sum from ADCs will be sent directly into the OFM memory space without S&A. In summary, the un-configured weight partial sums of MAC are parallel fed into subsequent S&A module of mixed-bit PE in the spatial dimension and IFM partial sums are serially fed into the S&A module in the temporal dimension, as shown in Figure 5.4 (b).

5.4 Column Parallel Readout

5.4.1 Column Parallel Structure

Figure 5.5 shows the current-based CIM array of a low-throughput column serial readout scheme and the proposed charge-based CIM array for a high-throughput column parallel readout scheme. In order to minimize the influence of ReRAM device variation and large A/D peripheral overhead in high parallelism CIM architecture, both ratios of activation per column and column parallel have been confined to a certain level (80; 103), which not only limits throughput but also imposes additional control and signal multiplexing power.

In these aforementioned designs, analog signals from each column are initially amplified by an I/V operational amplifier (OPAM) to convert the signal from current to voltage and clamp the signal within an acceptable region due to current

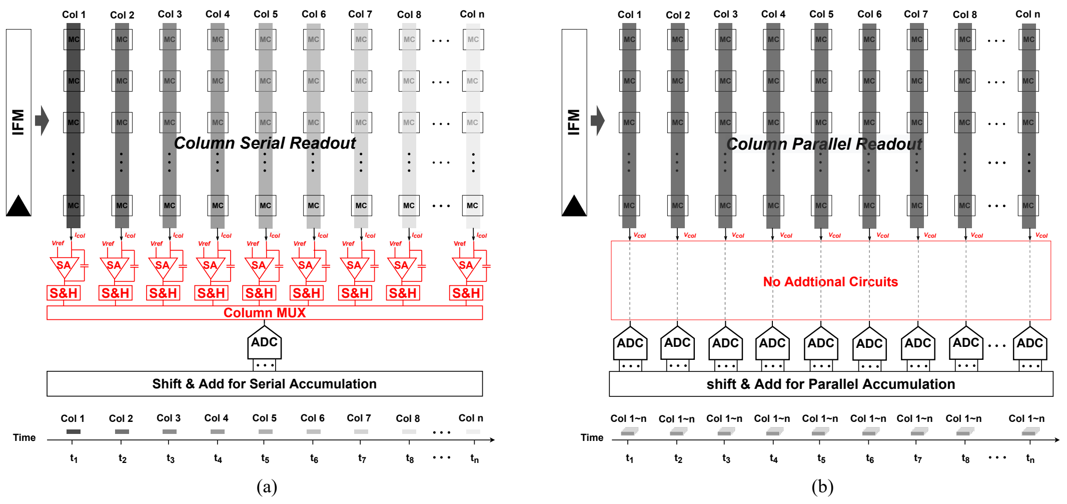


Figure 5.5: ReRAM-based CIM array comparison with different readout schemes: (a) conventional column-serial readout scheme; (b) proposed column-parallel readout scheme.

accumulation. The amplified voltages are then either stored in sample and hold (S&H) modules awaiting multiplexing and subsequent A/D conversion. Signal multiplexing would inevitably reduce throughput, hence some accelerators employ an asynchronous clock to drive the multiplexer and ADC or simply use a high macro frequency, e.g., 256 shared one multiplexer and one ADC which resulting an ADC frequency 255 times large than the main macro frequency (80) or 1GHz macro frequency (113). The column serial design indeed could retrench the area overhead of ADC by great a portion, but it would not render any improvement on energy efficiency. Because the power consumption is increasing with the reduction of ADC number linearly, which cancels out the increment on throughput of higher frequency, not to mention the extra control, amplification, and multiplexing power. Therefore, the column serial is area efficient design instead of an energy efficient design.

For the proposed CIM array, a column parallel readout method is employed. After storing the *AND* result on the capacitor of each macro cell, all the capacitors in the same column are connected together for sharing charges and generating accumulation voltage on BL. In this way, no voltage-clamp circuit is required and BL is connected to ADC directly. Besides, a sampling-and-hold (S&H) circuit in conventional ADC design is also eliminated. The voltage on BL can be stored during

ADC conversion and sent as input of the comparator. In the accumulator block, the ADC outputs from each column are shifted according to the weight factors and accumulated in parallel for high throughput performance. The shifting numbers are configurable to support mixed-bit operations.

In CIM architecture, the required margin of dynamic range decreases with activation per column, signal margins are much more likely to be overlapped in a small range due to variance caused by high activation per column and excessive signal margin overlapping makes computed outputs non-deterministic. There are some attempts to alleviate the problem of signal margin degradation by limiting activation per column to less than 20 row activation simultaneously (106; 116). Another merit of the proposed charge-based macro is that it could isolate most of the influence of ReRAM variance and only consider the impact of the capacitor with ultra low variance. To evaluate the impact of current-based macro and charge-based macro variance on signal margin, two types of macro are modeled as two distinct Gaussian distributions. Each macro is evaluated with 100 K-point Monte Carlo sampling. The signal margin standard deviation of current-based macro overlaps with dynamic range when activation per column is over 64, while standard deviation for charge-based macro almost declines with dynamic range and guarantee no computation error for different parallelisms (2 to 4096) (34).

Since the signal margin standard deviation of capacitor perfectly matches the requires dynamic ranges of high parallelism computing (up to 4096), we could further evaluate the efficacy of a proposed multi-bitwidth PE in different parallelism without concerning the computed error from analog domain. Due to error-less analog accumulation, an ADC with a resolution of $log_2(N)$ perfectly matches the required dynamic range for N parallelism computing (e.g., a 12-bit ADC for 4096 parallelism). However, in the proposed multi-bitwidth PE MAC, ADC's resolution is set to 8-bit for system overhead concerns. Indeed, a relatively low resolution ADC for high parallelism computing would result in a certain degree of loss in precision. Previous work (83) provides fundamental analysis for signal-quantization noise ratio (SQNR) of proposed multi-bitwidth PE with consideration of rounding effects of ADC. Here, we evaluate the degree of loss by considering the trade-off between parallelism and signal noise ratio (SNR) of typical multi-bitwidth reconfiguration module. The typical SNR is formulated as:

$$\mathcal{SNR} = \frac{P_{signal}}{P_{noise}} = \frac{\sum_{n=1}^{N} y_n^2}{\sum_{n=1}^{N}(y_n - \sum_{i=0}^{7}(y_{nq} << i))^2} \tag{5.1}$$

where y_n is the dot product result of two original float-point vector, y_{nq} is denoted as the dot product of two 8-bit ADC digitized bit-significance-wise partial sum of two orginal vector, symbol $<<$ represents the shift left procedure from LSB to MSB. When parallelism N equals 255, the calculated SNR is 48.4dB and perfectly matches with 8-bit ADC, meaning loss-less computing. Meanwhile, the typical SNR drops dramatically as parallelism goes up (rounding to 1). Compared with experiment result captured from (83) (minimum SQNR of 10dB at 2304 parallelism), the proposed multi-bitwidth PE with 8-bit ADC and enabling of 2048 parallel computing satisfies the minimum requirement for typical multi-bitwidth reconfiguration. To further explore the efficacy of the proposed charge-based macro, a comprehensive neural network robustness analysis under different variance settings is given in Section 6.3.2.

5.4.2 Low-Power Cycle-reduced ADC By Weight Preprocessing

When each weight is arranged on the ReRAM array as a binary number, there will be a large number of 0s distributed in it. If these 0s can be effectively used to reduce the power consumption of readout circuits, the computing energy efficiency can be effectively improved. In some previous work (61), a block-based design was used to make the ADC turn off when executing a block with all "0"s. In this way, a reduction in ADC power consumption is achieved. However, in order to maintain a good compression ratio of the network, the block cannot be very large, which leads to the problem of low parallelism. To solve these problems, as shown in Figure 5.6(b), the low power cycle-reduced ADC is designed.

When increasing the throughput, column parallel computing needs to employ a large number of peripheral readout circuits in parallel. Large power consumption poses an urgent requirement for energy-efficient readout circuits. The main mechanism of

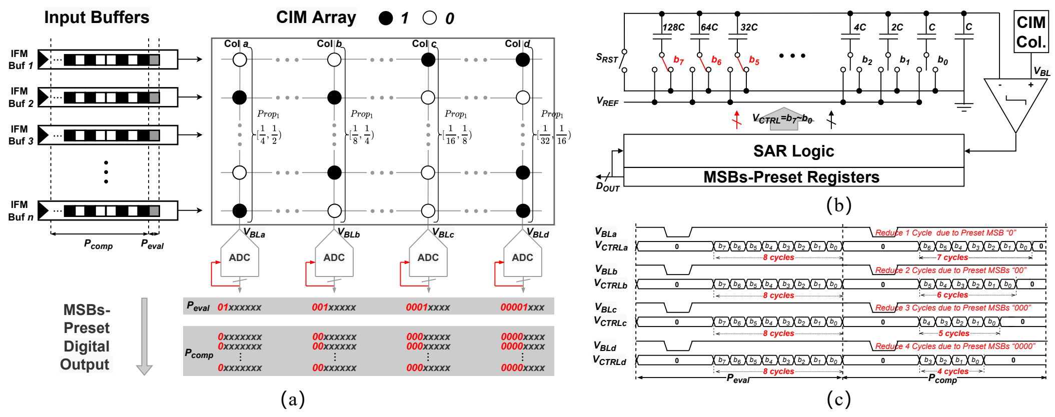


Figure 5.6: Low-power cycle-reduced ADC System: (a) working flow, (b) schematic and (c) waveforms.

optimized ADC is shown in Figure 5.6. The calculation process of the CIM system based on ReRAM is divided into two stages. The first step (set stage) occurs when the entire system is just powered on. In the set stage, each row in the ReRAM array will be entered with a "1" input. Next, a digital output SUM_0 will be generated below the ADC. It is stored in the register. In this way, SUM_0 represents how many "1"s there are in the weights stored in one column. For example, after the above process, the ADC output of a column is 0011_1111. This means that, regardless of the input, the column will produce output up to 0011_1111. Therefore, the first two bits of the output converted by the ADC are determined to be "0". Thus, the comparison operations of the first two bits are skipped in the following stage by ADC controller.In the calculation stage, ADC keeps the first two bits from flipping, so as to achieve the purpose of reducing the power consumption of the ADC. We can easily conclude that this method is effective only when there is more than 50 percent of the "0" data stored in a column in the ReRAM array. As the data of "0" in a column increases, the number of bits that are determined not to be flipped will increase, and the power saving will be more obvious.

This method is suitable for taking the advantage of sparse weights. After storing the weight values on macro cells, the zero numbers in each column are calculated for ADC cycle-reduction evaluation. Thanks to large proportion of 0 in weight data, normally at least one MSB comparison cycle can be skipped during calculation stage. The detailed power performance improvements under different sparsity rates are quantized in Section 6.3.

5.5 Systolic Dataflow

System-level design is developed to map neural networks onto the ReRAM accelerator to further boost energy efficiency. We define the PE array as an integrated module that can process a whole layer of convolutional operations. Thus, a full neural network can be directly executed from one layer to another. Each PE array contains multiple PEs, and the data flow between different PEs is in a systolic manner. The number of PE mainly depends on the size of the kernel used in the convolution calculation. In Figure 5.7, a $K{\times}K$ kernel requires $K{\times}K$ PEs to store the weights. The proposed CIM array employs three layers of horizontal systolic dataflow to reduce data bus and network for better data transmission efficiency(92). In conventional systolic design (93), TPU reduces the data transmission overhead, however, the weight data are still refreshed after finishing a batch of MAC operations.

In this systolic structure, since the mixed-bit MAC operations are completed within each PE, only the input data and partial sum data flow between PEs while weight data is stored in each PE, i.e. weight stationary. To make the best use of the proposed high activation per column charge-based macro, we map the weights on the same position of kernels across different channels into the same column of one macro cell array (92; 113; 121). As shown in Figure 5.7, input feature maps IFMs flow from left to right and partial sums flow from right to left, which are highlighted in red and blue lines, respectively. In this work, the correct partial sums are generated in two cycles after the corresponding IFM are fed in, which results in useless data within two cycles. To make the best use of computing cycles and boost throughput, two batches of uncorrelated IFMs are utilized in a time-interleaved manner. Specifically, if one kernel is sliding horizontally, the current IFMs need to be interleaved with IFMs that are sliding perpendicularly one stride from the current position. To enable a full operation of CNN, the proposed system also includes modules for accumulation, max pooling, sigmoid activation , and batch normalization.

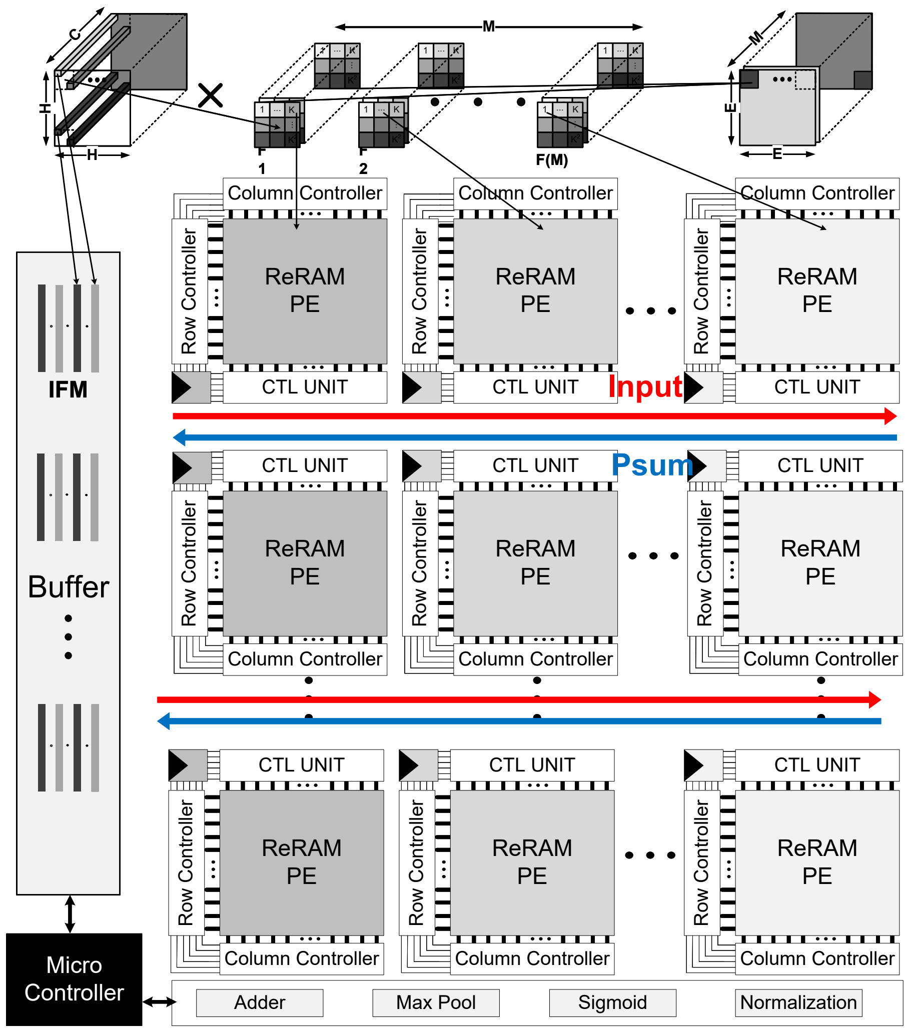


Figure 5.7: System structure with systolic dataflow and weight mapping strategy.

The pre-trained weights are loaded onto macro cells in advance. Assuming one PE has a size of $m \times n \times p \times q$. To match up with the systolic settings of $K \times K$ PEs within the CIM array, columns of each PE are loaded with weights in the same location of each filter for p of C different input channels in different bits significance, as illustrated in the bars from light grey to dark grey of Figure 5.7. The proposed mapping method is to map the 3D convolutional layers to PEs, where the size of the 3D IFM is $H \times H \times C$, and the PE array stores M 3D convolution kernels, and the size of each convolution kernel is $K \times K \times M$. In the PE array, k rows correspond to the width k of the convolution kernel, and k columns correspond to the height k of the convolution kernel. In PE, M rows correspond to the number of channels M of the convolution kernel, M columns correspond to the number M of convolution kernels, and IFM inputs C rows each time (113; 121). In short, our mapping method is to convert the 3D convolution kernel into a 2D convolution kernel, and also convert the 3D IFM to 2D IFM, and then perform the convolution operation. For IFMs, data will be split in bit significance as well and fed into the rows in sequence. When computing, each PE has a series of adders in charge of accumulating partial sums from ADCs within PE in systolic flow.

The corresponding experiment results of NAS optimized CNNs on proposed ReRAM-based CIM accelerator is organized in section 6.3.

Chapter 6

Experiment Results

6.1 Results on Analog-domain SRAM-based CIM Accelerator

In this section, the NAS-optimized network with layer-wise precision and sparsity is analyzed. The linearity performance of CIM array is evaluated to prove significant advantages. In the final, the chip results are measured and verified with comprehensive comparison with the state-of-the-art works.

Table 6.1: Results of Optimized ResNet-18 on CIFAR-10 Dataset

Strategy	**Input Bits**	**Weight Bits**	**Sparsity Rate**	**Comp Rate**	**Top-1 Acc**
Q[1]	8bits	8bits	N/A	1×	92.42%
	4bits	4bits	N/A	2×	90.23%
	2bits	2bits	N/A	4×	N/A
Proposed MQ&BS[2]	8bits	Mixed	96.10%	5.14×	92.17%
	4bits	Mixed	96.73%	5.25×	91.29%
	2bits	Mixed	97.12%	5.37×	90.55%

[1] Model-quantified in fixed precision mode.

[2] Model-quantified in layer-wise mixed-precision mode and bit-level sparsitified.

[3] Sparsity Rate = "0" bits in conv layers / total bits in conv layers.

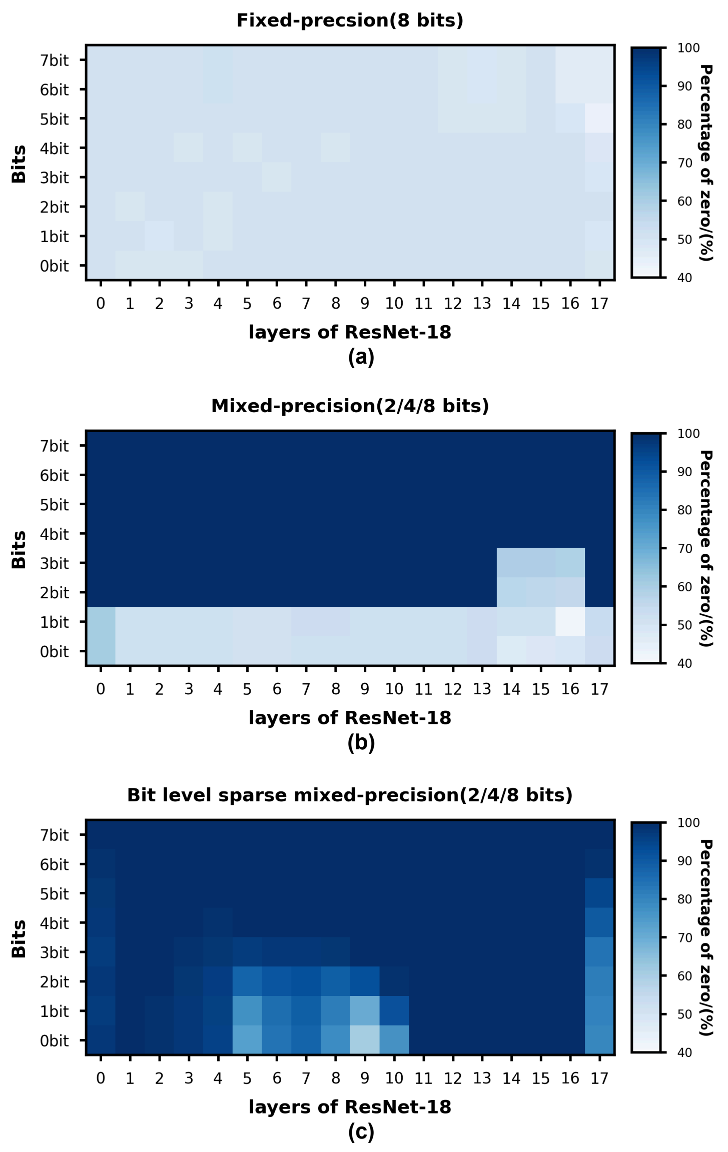


Figure 6.1: "0"s distribution of ResNet-18 networks: (a) fixed-precision, (b) mixed-precision, (c) bit-level sparse mixed-precision.

6.1.1 NAS-optimized Network

In this work, the ResNet-18 network is optimized for precision and sparsity using the proposed NAS method and evaluated on the CIFAR-10 dataset. Table 6.1 shows that with both inputs and weights quantized to 4 bits, accuracy drops by nearly 2% with 4× network compression. The bit-level sparse mixed-precision quantization method determines the optimal bit width for each layer and trains more bit-level "0"s in the weights, resulting in less than 1% accuracy loss and higher compression compared to traditional methods. Figure 6.1(a) and (b) shows that fixed-precision

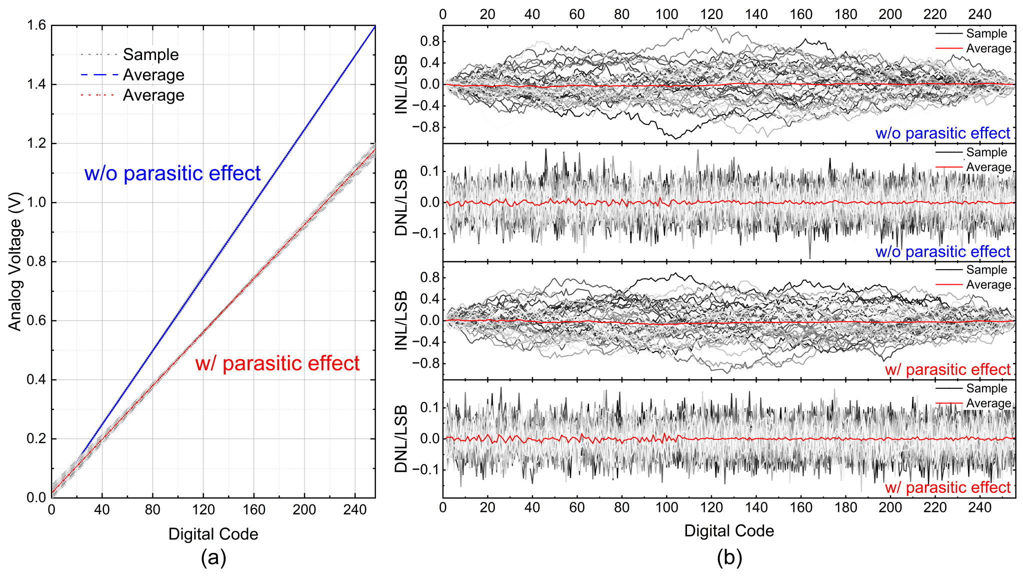


Figure 6.2: Linearity performance of bootstrapped CIM array under Monte-Carlo simulation @100MHz: (a)array analog output voltage curve, (b)INL and DNL.

(8-bit) and traditional mixed-precision methods have around 50% ”0”s per bit in weights, while the bit-level sparse mixed-precision method significantly increases bit-level ”0”s, which makes it possible to optimize the ADC performance at the bit level.

6.1.2 CIM Array with Bootstrapped SRAM Cell

To validate the functionality and evaluate the bootstrapped SRAM performance, the charge-domain CIM SRAM array with bootstrapped SRAM cells is constructed in 28-nm process through Cadence Virtuoso platform. A stepped analog MAC output voltage curve is obtained by sequentially inputting thermometer-code data into the CIM array, while a linearity analysis of the output voltage is performed with evaluation result shown in Figure 6.2.

The optimal quantization range for the ADC is analyzed to determine the impact of the input voltage range on quantization accuracy. As shown in Figure 6.3, for 8-bit SAR ADC, along with the input voltage decreasing, the dynamic performance of the ADC becomes progressively worse, and the non-linear characteristics of the ADC are also affected, resulting in a degradation of the static performance. As the

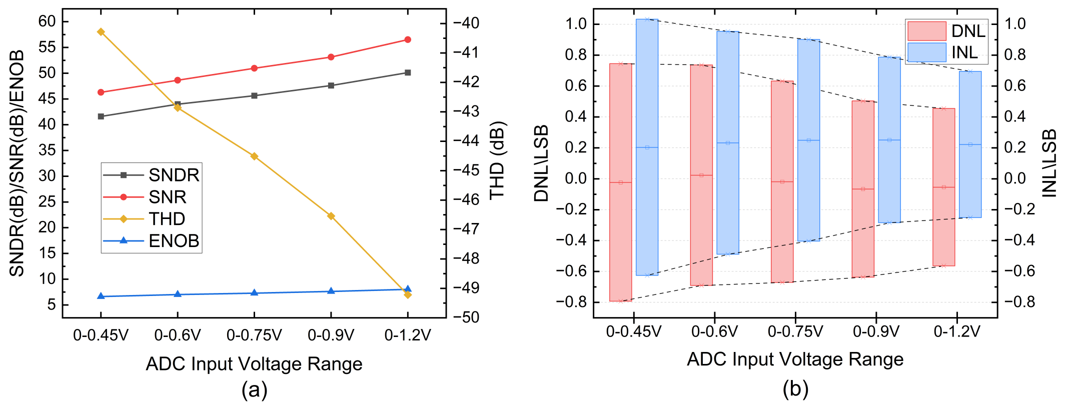


Figure 6.3: Linearity performance of ADC under input voltage range variation: (a)dynamic performance, (b)static performance.

input voltage range increases from 0.6V to 1.2V, the signal-to-noise-distortion ratio (SNDR) increases by approximately 6.12 dB, the effective number of bits (ENOB) increases by approximately 1.012, the integral non-linearity (INL) and differential non-linearity (DNL) range decreases by approximately 0.495 LSB and 0.408 LSB, respectively. 1.5× INL and 1.4× DNL improvements are achieved.

6.1.3 Sparsity-aware ADC

To evaluate the ADC performance under different sparsity conditions, the 8-bit sparsity-aware SAR ADC for 1152-row CIM array is designed under TSMC 28-nm process. The one-hot code generation module is added for control signals.

Power: The proposed sparsity-aware ADC gates the corresponding logic at different sparsity rates to reduce power consumption. Simulations show that when the sparsity rate is below 77.78%, the ADC operates in full-conversion mode with a power consumption of 41 μW. As shown in Figure 6.4, increasing sparsity reduces ADC conversion cycles, with DAC array power consumption decreasing exponentially and other logic decreasing linearly. Compared to traditional ADCs, the proposed design adds only 2 μW of power in full-load mode and less than 5% additional area. In bit-level sparse networks, most ADCs donot work in full-conversion mode, significantly saving system power and improving energy efficiency.

Energy Efficiency: The utilization of the array when mapping the CNNs layer-by-

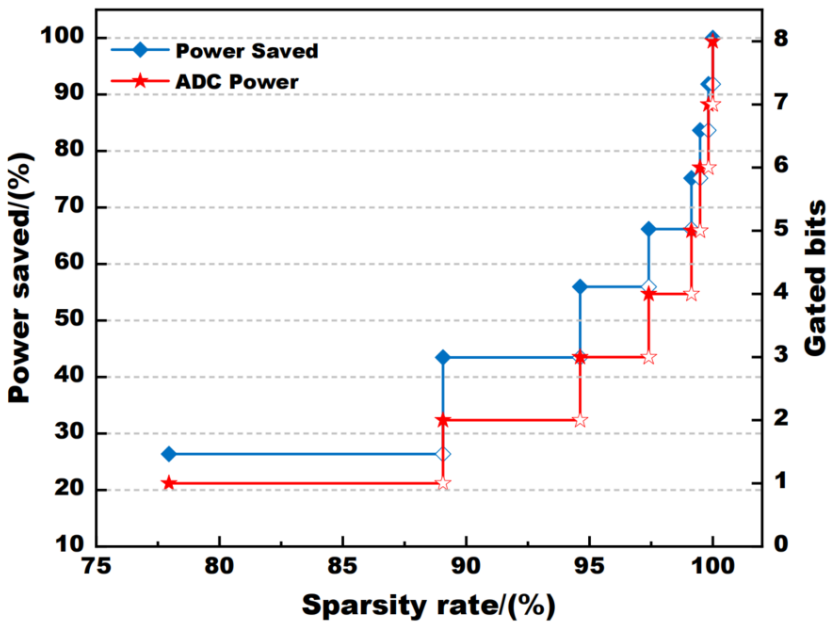


Figure 6.4: ADC power consumption evaluation with different gated bit numbers.

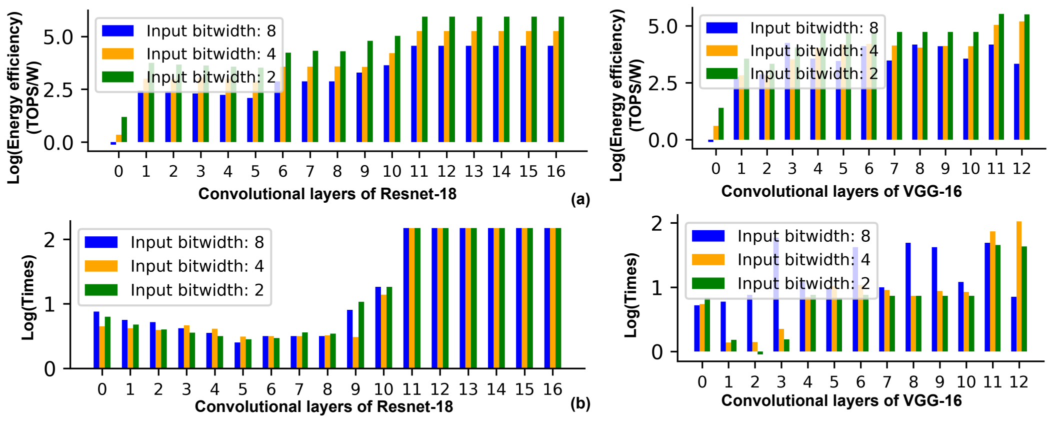


Figure 6.5: (a) Layer-by-layer energy efficiencies for ResNet-18 and VGG-16 with input bit width 2/4/8. (b) Layer-by-layer energy efficiency improvement for ResNet-18 and VGG-16 with input bit width 2/4/8.

layer is considered when performing layer-by-layer energy efficiency calculations. Besides, the performance is compared to the conventional scheme without the sparsity-aware ADC and bit-level sparsity-quantified method. The results are shown in Figure 6.5. The peak energy efficiency can reach 185.16 and 99.78 TOPS/W when the input bit width is 2 bits in ResNet-18 and VGG-16 networks with up to 9.20× and 8.72× improvements, respectively.

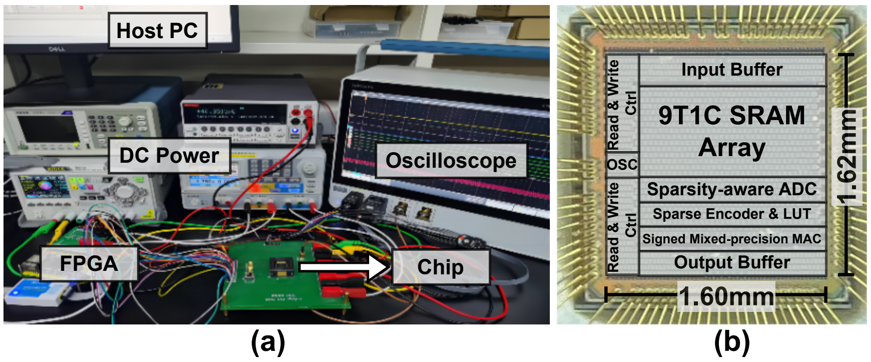


Figure 6.6: (a) Experiment setup. (b) Die photograph.

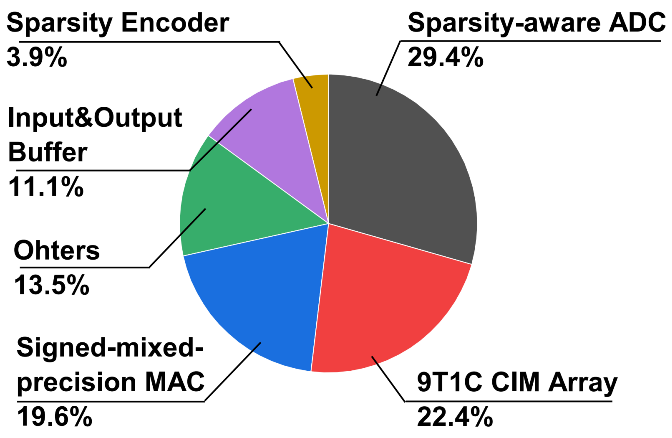


Figure 6.7: Chip power breakdown.

6.1.4 System Evaluation

The accelerator system with 1152×128 CIM array was measured in weight stationary scheduling. Figure 6.6 shows the experimental setup and die photogragh. Figure 6.7 shows that the CIM array, the sparsity-aware ADC and the digital processing block occupies 22.4%, 29.4% and 19.6% of total on-chip power, respectively. Table 6.2 shows the result of the proposed work with an operational frequency ranging from 160MHz to 340MHz under the ResNet-18 benchmark. Benefiting from the high throughput of the column-parallel yet low-power sparsity-aware ADC, as well as the signed-mixed-precision MAC, a peak energy efficiency of 302.78 TOPS/W are achieved under 2-bit configuration, surpassing all state-of-the-art analog CIM works (13; 59; 61; 122–124). Moreover, this work also achieves energy efficiency of 75.69 TOPS/W and 18.91 TOPS/W under 4, 8-bit configurations, respectively. Moreover, with 0.75% loss in accuracy compared to floating point 32-bit configuration, this

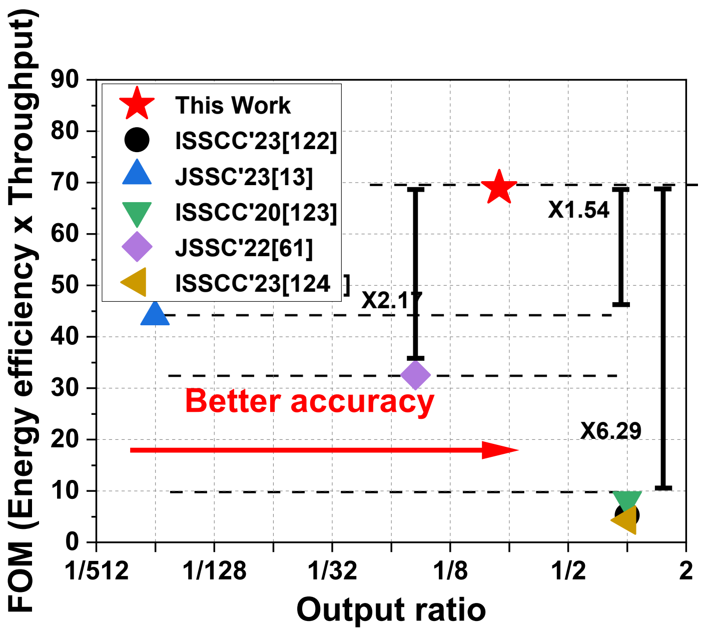


Figure 6.8: FOM comparison between this work with NAS optimized NN and previous CIM macros (normaized to 28nm and 4-bit configuration).

Table 6.2: Measurement Result and Comparison Table

Design	**ISSCC'23 (122)**	**JSSC'23 (13)**	**ISSCC'20 (123)**	**ISSCC'21 (59)**	**JSSC'22 (61)**	**ISSCC'23 (124)**	**This work**	
Technology	22nm	22nm	28nm	16nm	65nm	28nm	28nm	
Precision Type	Fixed-precision		Multi-precision				Mixed-precision	
Precision (Bits)	A/W=8	A/W=8	A/W=4,8	A/W=4,8	A/W=4-8	A/W=2∼8	A/W=2,4,8	
Cell Type	9T1C	9T	6T	8T1C	8T	8T	9T1C	
Area (mm^2)	1.88	0.25	N/A	25	5.66	1.3	2.59	
Macro Capacity (Mb)	0.125	0.125	0.0625	4.5	0.0156	0.0156	0.14	
Supply Voltage (V)	0.72∼0.82	0.7∼1.1	0.7∼0.9	0.8	0.9∼1.05	0.54∼0.9	0.9∼0.95	
Frequency (MHz)	280∼360	145∼240	119∼243	200	25∼100	20∼230	160∼340	
Peak Energy Efficiency^1 (TOPS/W)	172.16	244.8	143.724	218.52	71.6	130	302.78	
Neural Networks	ResNet-20	LeNet-5	ResNet-20	VGG-11	ResNet-18	ResNet-18	ResNet-18	Mixed ResNet-18
Classification Datasets	CIFAR-10	MNIST	CIFAR-10	CIFAR-10	CIFAR-10	CIFAR-10	CIFAR-10	CIFAR-10
Accuracy Loss(%)^2	N.A.	-2.35(8b)	-6.18(8b)	-5.99(4b)	-5.72(4b)	-1.95(8b)	-3.22(4b) -1.75(8b)	-2.16 (Mixed)
Reported Energy Efficiency (TOPS/W)	21.38(3b)	32.2(8b)	68.44(4b) 16.63(8b)	121(4b) 30.25(8b)	6.91(4b) 1.78(8b)	102(8b)	75.69(4b) 18.91(8b)	135.19 (Mixed)
Normalized Energy Efficiency (TOPS/W)^2	10.75(8b)	15.30(8b)	68.44(4b) 16.63(8b)	54.6(4b) 13.65(8b)	6.91(4b) 1.78(8b)	67(4b) 36.68(8b)	75.69(4b) 18.91(8b)	135.19 (Mixed)

1: Normalized to 28nm at 0.9V operational power (Reported EE×(Node/28nm)×(Voltage^2/0.9V^2)).

2: Compared to software simulation result of floating-point configuration.

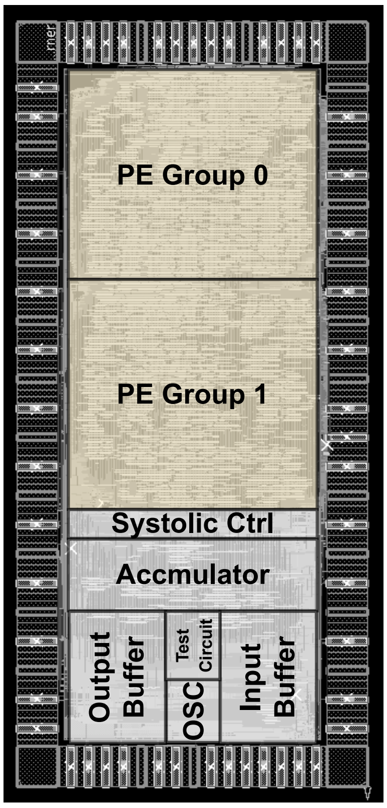
work delivers the highest average energy efficiency of 135.19 TOPS/W under the mixed-precision configuration. As such, the mixed-precision network can achieve an equivalent accuracy at 4-bit while maintaining a high energy efficiency at 2-bit. These measured results indicate that the proposed CIM macro realizes great balance between accuracy and power consumption. In this work, the Figure-of-Merit (FOM) comparison table is provided in Figure 6.8. The FOM value is based energy efficiency multiplying throughput performance for comprehensive evaluation. As shown, the proposed work achieved best FOM performance with at least 1.54× than other works. Additionally, the output ratio (61; 123) is the quotient of the actual ADC output resolution and the number of analog levels preceding the ADC. A higher output ratio correlates with improved accuracy.

6.1.5 Conclusion

The first work proposes a bootstrapped-SRAM CIM accelerator with a wide output voltage range to support precision and sparsity-optimized CNNs. The design includes a NAS method for optimizing precision and sparsity, a bootstrapped SRAM CIM array for fully-parallel computing and wide output range, and sparsity-aware ADC circuits for better energy efficiency. The design is verified on NAS-optimized networks VGG-16 and ResNet-18 using CIFAR-10 dataset, which could achieve an equivalent accuracy at 4-bit of 68.68% while maintaining a high energy efficiency at 2-bit of 135.19TOPS/W by measurements.

6.2 Results on Digital-domain SRAM-based CIM Accelerator

In this section, we analyze the power consumption and accuracy of the binarized BERT-Base/Tiny network on various datasets, implemented using our proposed digital-domain CIM accelerator. We compare our full-digital CIM design to state-of-the-art solutions and demonstrate significant advantages. Finally, we present chip results to validate our approach.



Technology	28nm CMOS
CIM Array Size	2KB
On-Chip SRAM Capacity	36KB
Macro Size	3.12um^2
Die Size	1.995mm^2
Supply Voltage	0.60~0.99V
Frequncy	60~400MHz
Power	62.96~419.74mW
Peak Throughput[1]	6.55TOPS
Area Efficiency	3.28TOPS/mm^2
Peak Energy Efficiency[2]	27.24TOPS/W
Average Energy Efficiency[3]	20.98TOPS/W

1: Measured at SS corner under 0.8V supply voltage.
2: Measured at TT corner with toggle rate equal to 50% under normalized 0.65V supply voltage.
3: Measured at TT corner with BERT-Tiny under normalized 0.65V supply voltage.

Figure 6.9: Layout photograph and chip summary.

6.2.1 Experiment Setup

The proposed CIM macro was designed and characterized using Cadence Virtuoso and Liberate MX. A latch-like SRAM RTL behavioral model was employed as a software baseline benchmark. The remaining accelerator components were implemented in RTL Verilog HDL, leveraging Memory Compiler-generated SRAM. Accuracy performance was evaluated using Synopsys PrimeSim AMS, incorporating SDF/DSPF

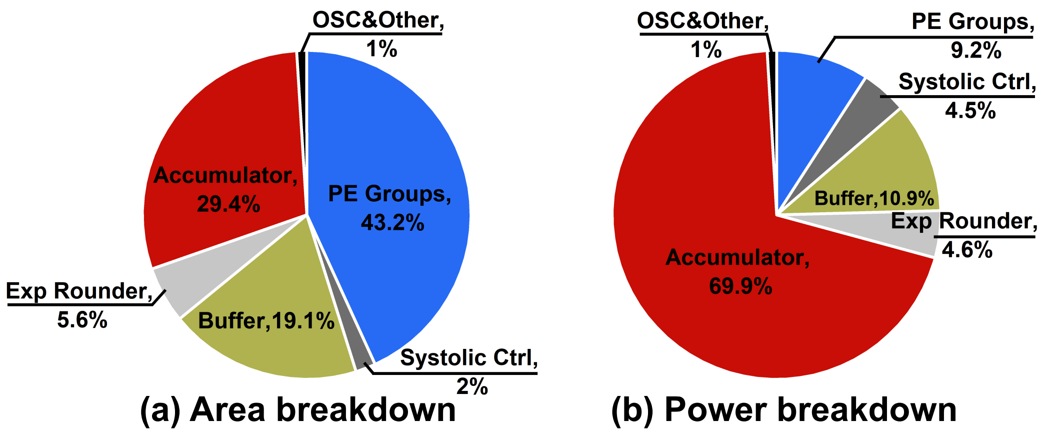


Figure 6.10: Area and power breakdown of proposed CIM accelerator. (a). Area breakdown. (b). Power breakdown.

RC file back-annotation. Area and power consumption were assessed through place-and-route in Cadence Innovus and timing analysis in Synopsys PrimeTime, respectively, targeting a 28nm technology node. The clock frequency was set to 400 MHz, resulting in a total post-layout area of 2 mm^2, as illustrated in Figure 6.9.

To avoid unfair comparison, the counterparts' energy efficiency is normalized to 28nm technology under 0.65V supply voltage. The scaling methodology is deployed from (125). CIM implementation (45) with 8-b computing space and layer-wise mantissa shifter is designed in accuracy comparison. Two weight binarized BERT models, e.g., BERT-Base and BERT-Tiny with four different datasets, e.g., SQuAD 1.1/2.0, STT-2 and Natural Questions (NQ) are used as the benchmarks for evaluation.

6.2.2 System Performance Evaluation

We evaluate the performance focusing on three aspects: BERT Binarization Evaluation, Binarized BERT Performance on CIM Accelerator, and System Comparison with existing designs.

The comprehensive evaluation demonstrates our design's advantages across multiple dimensions. In terms of model compression, our TWS binarization achieves 32×

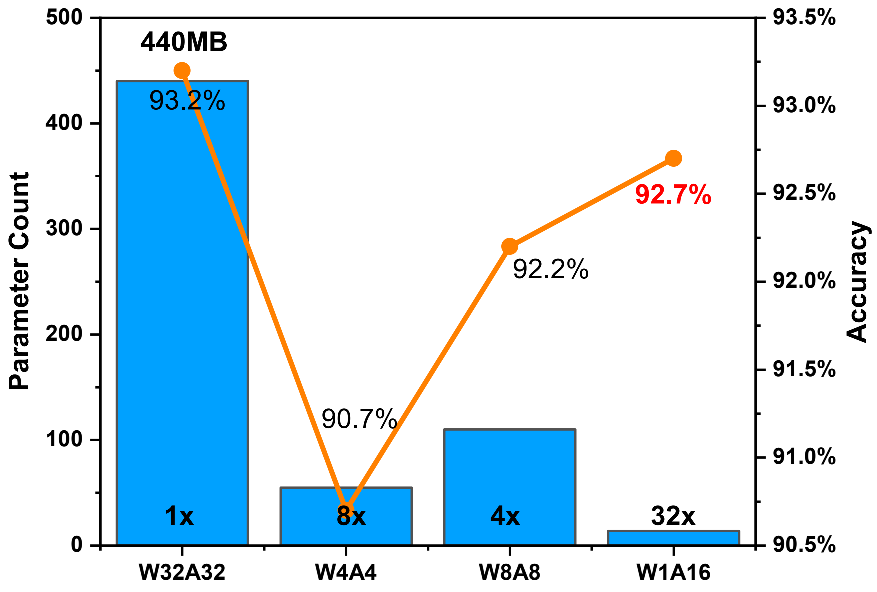


Figure 6.11: Comparison of compression ratio and accuracy for various quantization configurations on the SST-2 Dataset with (126).

reduction in model size while maintaining competitive accuracy. This is particularly significant for edge deployment, where both memory footprint and computational efficiency are critical constraints. The batch-wise processing strategy, combined with our group vector systolic architecture, enables efficient handling of different transformer operations while minimizing data movement overhead.

BERT Binarization Evaluation

The binarization of the BERT model plays a crucial role in optimizing the performance of the accelerator, particularly in terms of model accuracy and hardware efficiency. In comparison to previous works, the use of BF16x1 bit precision in our design yields superior results, as reflected in the accuracy scores for several benchmark datasets.

From Table 6.3, we observe that our model outperforms prior designs in terms of accuracy across all datasets. Specifically, on the SQuAD 1.1 dataset, our BF16x1 binarized BERT achieves 87.5% accuracy, surpassing VLSI'22 with 8-bit precision (87%), and significantly outperforming ISSCC'23 with FP-8 precision, which achieved only 92.2% on the same dataset. Similarly, on SST-2, our BF16x1 model achieves 93.1%

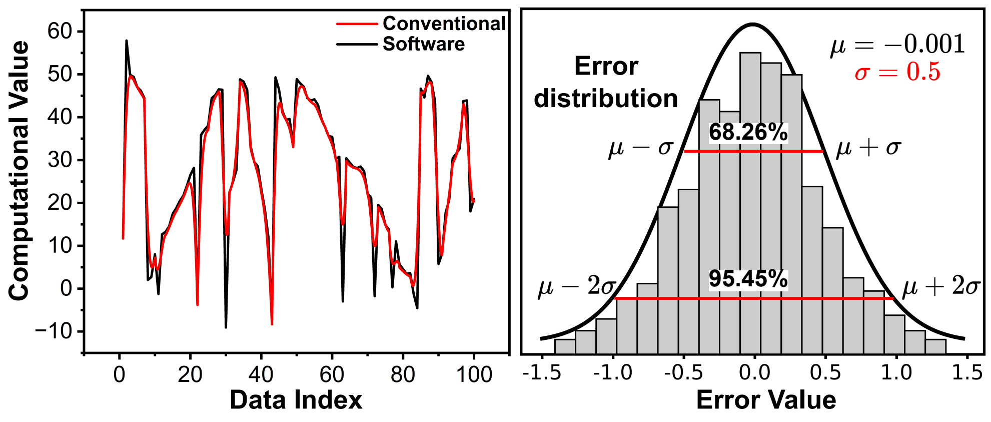


Figure 6.12: Conventional CIM implementation for floating point operation accuracy evaluation.

accuracy, which is higher than both ISSCC'23's FP-4 (90.7%) and VLSI'22's 8-bit (87%) results. On the SQuAD 2.0 dataset, our design achieves 88.3%, while JSSC'24's 16-bit BERT only reaches 45.73%. Our proposed configuration achieves a 32x compression ratio while maintaining 92.7% accuracy as shown in Figure 6.11, outperforming the AxW=FP4/8 configuration in (126). These results suggest that BF16x1 binarization strikes an optimal balance between precision and computational efficiency.

The key to this performance lies in the precision scaling of BF16, which maintains sufficient model fidelity while reducing hardware complexity. Unlike FP16 or FP32, BF16 focuses on mantissa reduction without losing essential information for transformer-based tasks, such as attention and context representation. This enables faster computation without significant accuracy degradation, making it ideal for resource-constrained environments such as edge devices. The bit-parallel computing structure and group vector systolic array further enhance the throughput, making BF16x1 particularly well-suited for tasks like SQuAD and SST-2, where both accuracy and speed are critical.

Binarized BERT Performance Evaluation on CIM Accelerator

The area and power breakdown are depicted in Figure 6.10. The systolic PE groups and Accumulator constitute the largest portions of area, accounting for 43.24% and

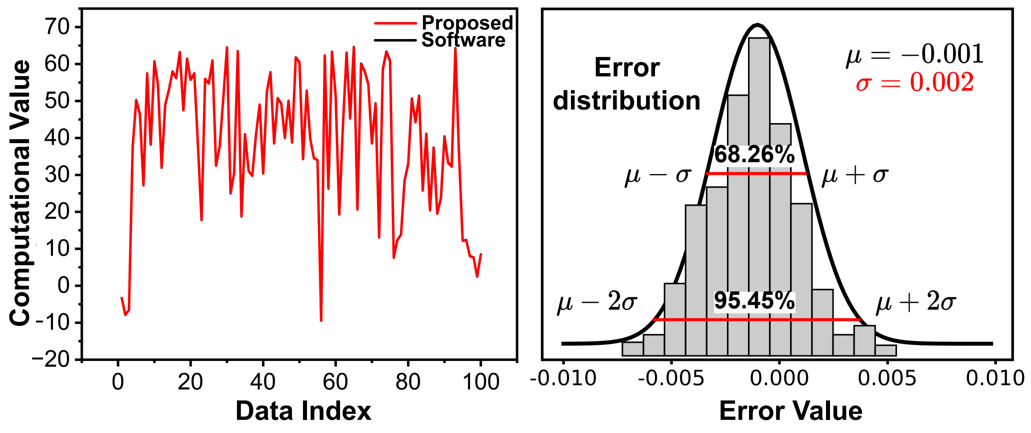


Figure 6.13: Proposed CIM accelerator for floating point operation accuracy evaluation

29.4%, respectively. The Accumulator and Buffer occupies the largest portions of power, accounting for 69.9% and 10.9%, respectively. The systolic control module only takes up 2% in both area and power consumption, which minimized data transaction power without substantially increase system burden. Figure. 6.12 shows the computed value comparison of conventional CIM implementation (45) with software results for 128×128 matrix-matrix multiplication. The simulated result indicates a computational error with mean of -0.001 and standard deviation of 0.5. Under the same settings, Figure. 6.13 demonstrates that computed value of proposed CIM accelerator coincides with software results. Benefiting from proposed batch-wise mantissa shifter and exponent rounder, proposed CIM architecture reduce the error standard deviation from 0.5 to 0.002 by two magnitude. Meanwhile, the exponent rounder only takes up 5.6% and 4.6% of area and power respectively, and thereby doesn't substantially burden the entire system.

System Comparison

The significant improvements in both area and energy efficiency stem from three key architectural decisions: (1) the bit-parallel computation paradigm that reduces cycle count and simplifies control logic, (2) the group vector systolic architecture that optimizes data movement and reduces register file requirements, and (3) the batch-wise mantissa shifting strategy that enables efficient floating-point operations

Table 6.3: Measurement results and comparison.

Design	VLSI'22 (127)	ISSCC'23 (126)	ISSCC'22 (128)	ISSCC'24 (129)	This work			
Technology	5nm	12nm	28nm	28nm	28nm			
Implementation	N.A.	N.A.	Digital CIM	Digital CIM	Digital CIM			
Precision (Bits)	AxW=4/8	AxW=FP4/8	AxW=8/16-b	AxW=8/16-b	AxW=BF16x1-b			
CIM Size (KB)	N.A.	N.A.	24	24	2			
On-chip SRAM (KB)	141	647	192	192	36			
Die Area (mm2)	0.153	4.6	6.83	6.98	1.995			
Supply Voltage (V)	0.46∼1.05	0.62∼1.0	0.6-1.0	0.6-1.0	0.65-1.0			
Frequency (Mhz)	152∼1760	77∼717	80∼240	80∼275	60∼400			
Power (mW)	N.A.	9∼122	27.04∼118.21	20.4∼119.7	63∼420			
Throughput(TOPS)	3.6(4-b) 1.8(8-b)	0.734(FP-4) 0.367(FP-8)	1.48(8-b) 0.37(16-b)	2.24(8-b) 0.56(16-b)	6.55			
Area Efficiency (TOPS/mm2)	23.3(4-b) 11.7(8-b)	0.16(FP-4) 0.08(FP-8)	0.22(8-b) 0.05(16-b)	0.32(8-b) 0.08(16-b)	3.3			
Model structure	BERT-Base	BERT-Base	BERT-Base	BERT-Tiny	BinaryBERT (BERT-Base)			BinaryBERT (BERT-Tiny)
Dataset	SQuAD 1.1	SST-2	NQ	SQuAD 2.0	SST-2	SQuAD 1.1/2.0	NQ	SQuAD 2.0
Accuracy Score(%)	87(4-b) 87.5(8-b)	90.7(FP-4) 92.2(FP-8)	73.6/54.4 (16-b)	48.94 (16-b)	93.1	88.3/76.6	69.25/55.8	45.73
Reported Energy Efficiency (TOPS/W)	95.6(4-b) 39.1(8-b)	18.1(FP-4) 8.24(FP-8)	20.5(8-b) 5.1(16-b)	15.71(16-b)	17.84	19.31/18.01	16.7	20.98
Normalized Energy Efficiency (TOPS/W)	8.55 (4-b) 3.5 (8-b)	7.76 (FP-4) 3.53 (FP-8)	15.28	8.61	17.84	19.31/18.01	16.7	20.98

1: Normalized to 28nm at 0.9V operational power (Reported EE×(Node/28nm)×(Voltage^2/0.9V^2)).

2: Compared to software simulation result of floating-point configuration.

with minimal overhead. These innovations work together to achieve superior performance metrics while maintaining high accuracy across different BERT variants and datasets. Our design's superior power efficiency is particularly evident in the normalized metrics, where we achieve 20.98 TOPS/W despite operating at a higher frequency than comparable designs. The power overhead from the batch-wise mantissa shifter and exponent rounder (4.6%) is more than compensated by the reduction in data movement and register file requirements, resulting in a net improvement in energy efficiency.

The proposed digital-domain CIM accelerator with 256×128 SRAM CIM array was measured with weight updating while computing in weight stationary scheduling. Table. 6.3 indicate proposed digital-domain CIM accelerator with an operational frequency ranging from 60MHz to 400MHz. Benefiting from high throughput BF16×1-b CIM macro, proposed CIM accelerator yield area efficiency of 3.28 TOPS/mm^2, surpassing (126) at 12nm technology node by 19× times, (128) by 13.9× times and (129) by 9.25× times. Though not optimized for high frequency computing, this work could realized equivalent area efficiency to (127) at 5nm technolgy node after scaling operation frequency to 400MHz. Moreover, owing to low power group vec-

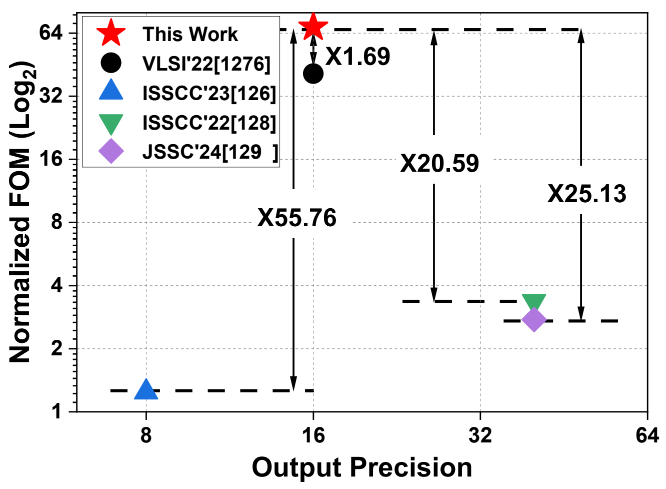


Figure 6.14: FoM comparison between this work and digital CIM implementations for BERT (126–129) (FoM=Normalized Energy efficiency × Area efficiency).

tor systolic CIM PE groups, this work achieves energy efficiency of 17.84 TOPS/W, 19.31/18.01 TOPS/W and 16.7 TOPS/W and 20.98 TOPS/W for BF16×1-b BERT-Based in SST-2, SQuAD 1.1/2.0, and NQ datasets respectively. For smaller networks of BF16×1-b BERT-Tiny in SQuAD 2.0, this work realized 20.98 TOPS/W in energ efficiency. This works exceed (127) in energy efficiency by 1.25× times, (126) by 1.29× times, (128) by 10% and (129) by 1.43× times. Although this work consume more power compared to SOTA counterparts for bit-parallel computing and low multiplexing ratio of adder tree, the throughput gaining still compensate power overhead increment and thus enhance efficiency.

As shown in Figure 6.14, our proposed accelerator surpasses state-of-the-art designs (126–129) in Figure of Merit (FoM), combining normalized energy efficiency and area efficiency. This improvement stems from minimized transaction power enabled by our group vector systolic CIM array and enhanced throughput through BF16×1-b macros. Table 6.3 provides a detailed comparison with state-of-the-art implementations.

6.2.3 Conclusion

The second work introduces a weight-binarized BERT model that can be highly efficiently mapped on a group vector systolic CIM accelerator. We employ knowledge distillation and TWS to obtain BF16x1-b BERT models, achieving a 0.5% to 2% accuracy improvement and a 4x to 8x compression ratio compared to INT4/8 quantization. To enhance efficiency, we propose high parallelism BF16x1-b CIM macro and group vector systolic CIM PE groups. Additionally, a batch-wise mantissa shifter and exponent rounder are introduced to boost computational accuracy. Post-layout results demonstrate that the proposed CIM architecture achieves an area efficiency of 3.3 TOPS/mm^2 and an energy efficiency of 20.98 TOPS/W for BERT-Tiny. Compared to state-of-the-art designs, our design achieves a 10.25x to 20.62x improvement in area efficiency and a 1.1x to 2.23x improvement in energy efficiency.

6.3 Results on Analog-domain ReRAM-based CIM Accelerator

In this section, the proposed methods are applied to accelerate the mixed-bits CNNs. Comparisons with previous works in terms of performance and energy efficiency are conducted.

6.3.1 Simulation Settings

To obtain the system performance of the proposed accelerator, digital modules includes S&As, adders, registers, and control logic are designed by Verilog HDL and synthesized by using Synopsys Design Complier in 28-nm process. ReRAM device model is adopted from (114). The ADCs are designed and evaluated on Cadence virtuoso in 28-nm process.

Hardware Settings

The ReRAM device in MAC is based on (114) with HRS of 1MΩ and LRS of 100kΩ. The ReRAM variation is ranged from 5% to 30% (97). MAC cell size is fixed to be 8×1. The column number is set to 8 to support 1∼8 bitwidth. Each PE is composed of 4096×256 MACs. The peripherals of control logic and precision reconfiguration are written in Verilog and synthesized in the DC compiler. 8-bit SAR-ADC is designed and evaluated in Virtuoso. The operation frequency of ReRAM main core is 20MHz considering the read latency of ReRAM (114). All modules are integrated on Cadence AMS for simulation.

Settings of NAS-optimized CNN benchmarks

To further investigate the performance of the proposed ReRAM-based NAS accelerator, the proposed accelerator is tested on NAS-optimized CNN (64) and summarized in Table I. The mixed-bit CNN benchmarks are the typical distributions of 2/4/8-bits weight in NAS-optimized CNN, under the same bit-width assignment for weight

Table 6.4: CIM Macro Cell Specification Summary

Designs	SRAM(34)	SRAM(98)	ReRAM(101)	ReRAM(44)	ReRAM(102)	This Work
Structure	8T1C	12T2C	1T1R	1T1R	2T1R	5T2R1C
Nonvolatile Storage	No	No	Yes	Yes	Yes	Yes
Fully-parallel Computing	Yes	Yes	No	No	No	Yes
Technology Node	65nm	28nm	130nm	55nm	14nm	28nm
Cell State Level	2	2	15	2	2	2
Functionality Domain	Charge	Charge	Current	Current	Current	Charge
Capacitance (fF)	1.2	1 & 0.5	-	-	-	1.3
R$_{OFF}$ (Ω)	-	-	~1.6M	-	~2M	1M
R$_{ON}$ (Ω)	-	-	133K	-	20K	100K
R$_{OFF}$/R$_{ON}$ Ratio	-	-	12	-	100	10
Area (μm^2)	1.8	2.06	1.69	0.2025	0.679	0.7
Operational Voltage (V)	0.94	1.1	5	1	0.5	0.8

and activation for each layer, and with the complexity cost model designed as DNAS (64) on ImageNet with a maximum compression rate of 2.31 and accuracy of 69.84%. Previous works of (6; 113; 116; 118–120) are used as benchmarks for throughput, and energy efficiency comparison.

6.3.2 Performance Summary

Macro Cell Performance Comparison

In Table 6.3.2, the performance specifications of several state-of-the-art ReRAM and SRAM macro cells are provided for comparison with the proposed one. When compared with SRAM macro cells, the ReRAM ones have nonvolatile and area-efficient advantages (44; 101; 102). However, most ReRAM designs cannot support fully-parallel computing, which can be guaranteed by the SRAM ones for high throughput performance (34; 98). In this work, the proposed ReRAM structure employs voltage-dividing, amplification and charge-sharing methods by serial-connected ReRAMs, an inverter and a capactior for device variation isolation and fully-parallel computing. In terms of area overhead, the capacitor, ReRAMs and transistors can be laid in 3D structure. The total area of proposed macro cell is determined by largest area, which is occupied by the capacitor. From TableTable 6.3.2, the area is slight larger than conventional ReRAM ones, but throughput performance is improved. When compared the SRAM ones, the area of proposed work is smaller with less transistors

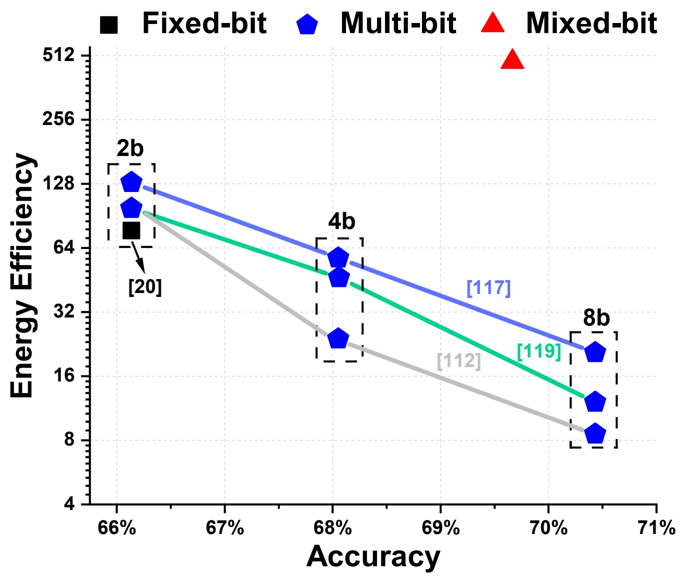


Figure 6.15: Energy efficiency evaluation in mixed-bit ResNet-18.

while has the advantage of nonvolatile storage.

NAS-optimized Mixed-Bit CNN Performance Evaluation

The proposed accelerator supports a flexible computational precision ranging from 1 bit to 8 bits, which is suitable for NAS-optimized mixed-bit CNN implementation. As discussed in previous sections, we can draw a conclusion that the throughput and energy efficiency of 4-bit mode and 2-bit mode can be greatly improved when compared to the pure 8-bit mode.

Figure 6.15 illustrates the theoretical energy efficiency enhancement result on accelerating mixed-bit CNN. We first quantize the ResNet-18 on ImageNet to 2-bit, 4-bit, and 8-bit. When compared with our NAS-optimized (2-bit prioritized) ResNet-18, only 8-bit quantized network shows competent accuracy with 0.75% enhancement. However, due to the relatively complex computing procedure of 8-bit operation, the highest energy efficiency among the current state-of-the-art multi-bit CIM architectures (113; 118; 120) could hardly reach 32TOPS/W, which is 14.18× smaller than that of the proposed accelerator.

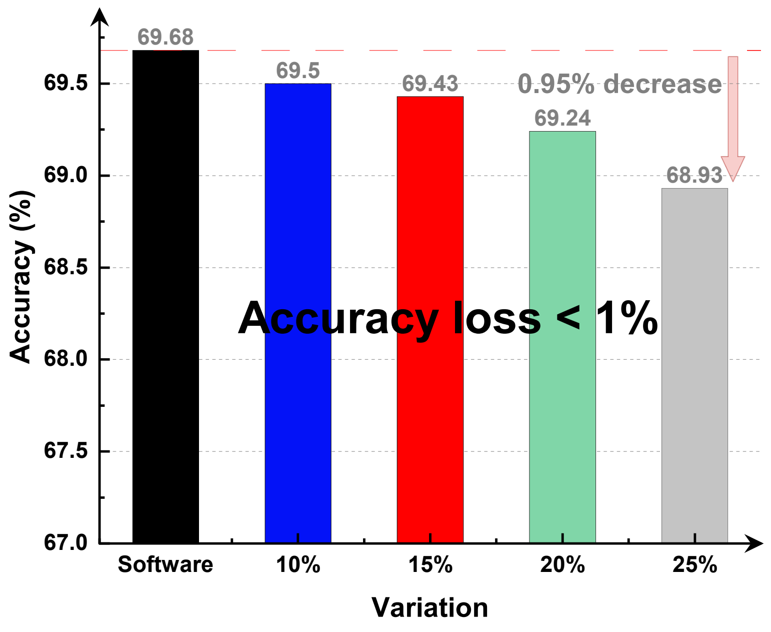


Figure 6.16: ResNet-18 inference accuracy with ImageNet datasets. Different configurations for variation sets are examined.

Column Parallel Readout Performance Evaluation

CNN Inference Accuracy To verify the inference accuracy of the proposed charge-based macro, we adopt a Pytorch-based simulator named Memtorch (130) to evaluate the proposed ReRAM accelerator with ImageNet datasets on a ResNet-18 convolutional neural network. The network accuracy is evaluated via the varying MAC result state of one column, which reflects the accumulated variance of ReRAM. The ReRAM models are set with a constant effective R-ratios (10) and four degrees of resistance variation (10%, 15%, 20% , and 25% for both LRS/HRS) are applied to the proposed charge-based macro to find the accuracy trend under different settings of variation. The ADC resolution is set to 8-bit with full column-parallel readout.

When testing the ImageNet test dataset, the inference accuracy of the ResNet-18 model is 69.68% for software simulation. The influence of ReRAM variation is shown in Figure 6.16. When the variation increases from 10% to 20% with a constant R-ratio of 10, less than 0.1% accuracy drop is observed. Even with the highest variation of 25%, the loss of inference accuracy is still less than 1%, demonstrating the robustness of the proposed macro-to-device variation.

Table 6.5: ADC Power Consumption under Cycle-reduced ADC

Total Weights SUM_0	ADC Results SUM_1	Cycle Reduction Numbers	ADC Power (uW)	Power Reduction Ratio
[0,8)	0	8	0	100%
[8,16)	1	7	1.08	93.8%
[16,32)	[2,4)	6	2.18	87.6%
[32,64)	[4,8)	5	3.34	81.0%
[64,128)	[8,16)	4	4.65	73.5%
[128,256)	[16,32)	3	6.25	64.4%
[256,512)	[32,64)	2	8.45	52.0%
[512,1024)	[64,128)	1	11.82	32.7%
[1024,2048)	[128,256)	0	17.56	0

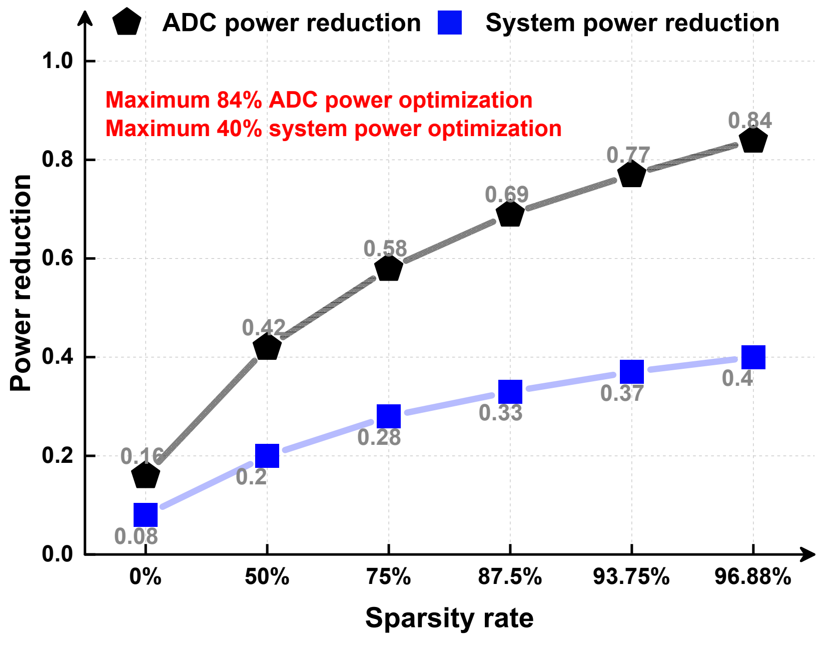


Figure 6.17: ADC and system power reduction under different sparsity rates.

ADC Power Optimization As shown in Table 6.5, in this design, the data of 2048 lines are turned on at the same time, and the 8bit ADC is used to quantize the data. SUM_1 here is the data after truncating two LSBs after 8bit ADC. When the value of SUM_0 is different, the ADC can skip the comparison stage of some data, thereby saving power consumption. For example, when the number of 1s in a column of weights is within the range of [32, 64), the result obtained after ADC is between [4, 8). At this time, ADC can skip the first 5 stage comparisons of MSB. In extreme cases, when the number of 1s in a column is between [0, 4), the ADC can be directly turned off at this time. In an ADC, the power consumption of the comparator and SAR logic is proportional to the number of comparisons, and the power consumption of the DAC array is proportional to the power of 2 comparisons. A summary of the power consumption of the ADC is shown in Table 6.5.

To evaluate the advantages of cycle-reduced ADCs, we explore the reduction ratio of ADC power consumption in the overall system at different sparsity rates. As shown in Figure 6.17, the cycle-reduced ADC can achieve up to 84% ADC power reduction and 40% system power reduction when the sparse rate is 96.88%. As the ratio of sparseness increases, the more bits the ADC can skip, the more advantageous this scheme achieves. Moreover, the sparse rate reported here refers to the ratio of the number of 0 (8 bits) in the network to the total number. It is worth noting that the 0 here is also represented by an 8-bit binary number. However, in our design, we consider whether a certain bit position is 0, so the sparse rate reported here is smaller than the actual number of binary 0s as a proportion of the total number of bits. This is why Cycle-reduced ADCs can also show some reduction in power consumption with a sparsity rate of 0.

System Comparison

The detailed performance metrics including throughput, power, and energy efficiency of different works are tabulated in Table 6.6. Here we do not consider weights updating, since we focus on real-time inference operation only. For a fair comparison, the peak operational bit of compared multi-bits and proposed mixed-bit CIM design (8-bit for proposed accelerator) energy efficiency of all objectives are normalized to 1-bit, i.e., energy efficiency $\times$ operational weight bits $\times$ operational IFM bits. The proposed ReRAM accelerator reveals the highest energy efficiency of 479.37 TOPS/W on a NAS-optimized ResNet-18 and a normalized peak performance of 2490.32 TOPS/W. Compared with fixed-bit implementations(6; 118), the

Table 6.6: System Performance Evaluation

Design		ISSCC (6)	TCAS-I (118)	ISSCC (120)	DAC (113)	TVLSI (119)	ISSCC (116)	This work
Technology		130nm ReRAM	45nm ReRAM	22nm ReRAM	40nm ReRAM	45nm ReRAM	40nm PCM	**28nm ReRAM**
Precision type		Fixed-bits		Multi-bits				**Mixed-bits**
Capacity(Mb)		0.155	0.0625	4	0.0625	0.0625	2	**36**
Precision Bits		1	8	2 ,4, 8	2, 4, 8	2, 4, 8	2,4,8	**1∼8**
Frequency(MHz)		20	25	66.67-200	1000	16.7	-	**20**
Activation Per Column		1	1	1/256	1/256	1	1/32	**1**
Power(mW)		39	199.68	2.99	6.83	12.6	5.43	**60**
Peak Performance (TOPS/W) (1b Norm)		156.8	39.04	391.4	544	2976	1436	**2490.32**
Average* (ResNet-18)	Precision Bit	2bit	8bit	2bit 4bit 8bit	2bit 4bit 8bit	2bit 4bit 8bit	2bit 4bit 8bit	**Mixed**
	Throughput (TOPS)	3.12	0.121	0.209 0.099 0.0035	0.256 0.064 0.004	2.184 1.092 0.397	1.95 0.829 0.475	**14.51**
	Accuracy	66.13%	70.43%	66.13% 68.06% 70.43%	66.13% 68.06% 70.43%	66.13% 68.06% 70.43%	66.13% 68.06% 70.43%	**69.68%**
	Energy Efficiency (TOPS/W)	78.4	0.61	97.85 47.26 11.91	97.32 24.33 11.14	744 199.2 31.56	130.5 57.6 20.5	**479.37**

* Accuracy are obtained via software simulation.

proposed accelerator shows much superior energy efficiency owing to large activation per column and elimination of OPAMP. Work (118) suffers from the huge overhead imposed by OPAMP for signal clamping, which consumes more than 90% of the power consumption of the CIM core. Compared to previous designs (6; 118), our design achieves much better peak energy efficiency, 14.88 × and 62.8 × higher, respectively. Among the multi-bit (8-bit) RRAM-based implementations, mixed-bit operation proposed work also presents the best energy efficiency, which is 42.03 ×, 22.38 ×, 14.18 × and 39.25 × better than (113; 116; 119; 120). (116) shows the second largest energy efficiency by optimizing the power consumption of OPAMPs and minimizing their usage, which also proves the importance of OPAMP elimination of the proposed accelerator. As shown in Table 6.6, (118) shows the highest peak energy efficiency for its one-shot capacitor-based analog reconfiguration design, however, the rigid reconfiguration capacitor prevents the system from supporting a mixed-bit operation. Work (116) implements a mixed SLC-MLC PCM design to boost area efficiency and shows the third highest energy efficiency. However, such a mixed design imposes extra power and its low-resolution ADC design increases the latency of high-bit signal processing as well as the power of signal multiplexing. Unlike analog CIM design, Work (113) uses a bit-counter to achieve digital Boolean logic of weight and IFM. The weight bits are read from the ReRAM array via a sensing amplifier one by one and then multiplied with IFM from a nearby buffer. Such digital design could eliminate a large ADC, however, it renders the system with low throughput and increases the possibility of inaccurate ReRAM read due to the high frequency of 1GHz. Due to low-resolution read-out peripheral design and precision discarding, (120) reveals smallest power consumption. However, a conservative strategy results in low activation per row and the smallest throughput in comparison with the second largest capacity. As a result, we believe this work is much more suitable for the practical AI applications on edge.

6.3.3 Conclusion

For the last work, an ReRAM-based multi-bitwidth systolic accelerator with large column parallel computing is proposed to support NAS-based CNNs with various 1∼8 bitwidths. Experiments have been implemented on NAS-optimized CNNs and demonstrated substantial energy efficiency improvement. Benchmarks show that the proposed ReRAM accelerator can achieve peak energy efficiency of 2490.32 TOPS/W and average energy efficiency of 479.37 TOPS/W with evaluating NAS-optimized multi-bitwidth CNNs, which has 14.18× improvement when compared with existing approaches(119).

Chapter 7

Conclusion and Future Work

7.1 Conclusion

To conclude, this thesis has shown a thorough study on analog-domain and digital-domain CIM architectures towards machine learning applications from system level perspective, architecture level perspective to macro level perspective.

From the system level perspective: Firstly, we propose a novel approach that combines Neural Architecture Search (NAS) with bit-level sparse quantization to optimize the precision bitwidth for each layer of a Convolutional Neural Network (CNN). This method induces a higher proportion of bit-level zeros during training, exceeding 50% on the CIFAR-10 dataset when training ResNet-18 and VGG-16. This increased sparsity enables significant opportunities for hardware-level optimization. Secondly, we propose a technique that combines Ternary Weight Splitting (TWS) with knowledge distillation to obtain binarized weights for BERT models. This approach achieves higher accuracy than INT8x1-b or INT4x1-b binarization at the same compression rate. Furthermore, unlike integer operations, floating-point operations on CIM offer promising avenues for hardware optimization, a domain that remains relatively unexplored.

From the architecture perspective: We have developed an analog-domain charge-based SRAM CIM accelerator, a digital-domain SRAM CIM accelerator and an analog-domain charge-based ReRAM CIM accelerator which can be concluded as follows:

- We propose a sparsity-aware ADC within the analog-domain charge-based SRAM CIM accelerator. This ADC dynamically adjusts its configuration based on network sparsity, disabling unnecessary logic to optimize power consumption while maintaining an impressive ENOB of 7.51. The ADC incurs an additional power consumption of only 2μW and an area overhead of less than 5% for sparsity-awareness, demonstrating effective energy efficiency optimization for bit-level sparse CNNs. Secondly, a hardware-efficient signed mixed-precision shift-and-accumulate unit is introduced. This unit supports weight data of 2, 4, or 8 bits, eliminating unnecessary hardware overhead. With a power consumption of only 0.26 mW, accounting for just 19.6% of the total system power, it significantly improves system energy efficiency. Finally, the analog-domain charge-based SRAM CIM accelerator was fabricated in 28nm technology and verified on NAS-optimized VGG-16 and ResNet-18 networks using the CIFAR-10 dataset. The accelerator achieved an equivalent accuracy of 68.68% at 4-bit precision while maintaining a high energy efficiency of 135.19 TOPS/W at 2-bit precision.

- We propose a BF16$\times$1-bit CIM PE within the digital-domain SRAM CIM accelerator. This CIM PE comprises 256 CIM macros organized in two rows. The batch-wise mantissa shifter and exponent rounder within each PE group, reducing computational error by two orders of magnitude (from 0.5 to 0.002) with only a $\sim$5% increase in area and power overhead. Furthermore, we employ a group vector systolic CIM architecture to balance register file cost and latency. By standardizing the data frame, we eliminate tensor reshaping overhead, further improving energy efficiency and utilization. Post-layout results in 28nm technology demonstrate that the proposed digital-domain SRAM CIM accelerator achieves an area efficiency of 3.3 TOPS/mm^2 and an energy efficiency of 20.98 TOPS/W for BERT-Tiny. Compared to state-of-the-art designs, our design achieves a 10.25x to 20.62x improvement in area efficiency and a 1.1x to 2.23x improvement in energy efficiency.

- We propose a mixed-bit MAC within an analog-domain charge-based ReRAM

CIM accelerator to support NAS-optimized mixed-precision networks with 1 to 8-bit precision. A cycle-reduced SRA ADC is proposed to leverage sparsity in computation, achieving a 32.7% to 84% reduction in ADC power consumption and a 20% to 40% reduction in system power consumption. The proposed ReRAM accelerator was evaluated on the ResNet-18 convolutional neural network using the ImageNet dataset, employing the PyTorch-based Memtorch simulator. It demonstrated outstanding robustness to PVT variations with less than 1% accuracy loss compared to software benchmarks. Benchmark results show that the proposed charge-based ReRAM accelerator achieves a peak energy efficiency of 2490.32 TOPS/W and an average energy efficiency of 479.37 TOPS/W when evaluating NAS-optimized multi-bitwidth CNNs, representing a 14.18x improvement over existing approaches.

From the CIM macro perspective: We have developed charge-based bootstrap 9T1C SRAM macro, digital-domain BF16×1-b SRAM CIM macro, charge-based 5T2R1C ReRAM CIM macro which can be concluded as follows:

- We propose an innovative charge-based bootstrap 9T1C SRAM CIM macro. By incorporating three additional transistors, our design effectively isolates the read signal from the storage node during read operations, surpassing traditional 6T SRAM cells. This enhancement not only stabilizes data reading but also substantially improves the macro's tolerance to read noise. Moreover, the proposed macro mitigates the impact of parasitic capacitance and doubles the signal margin, leading to a 6.12 dB increase in Signal-to-Noise and Distortion Ratio (SNDR) and a 1.012-bit improvement in Effective Number of Bits (ENOB). Leveraging the robustness of our bootstrap 9T1C SRAM CIM macro, we achieve a high degree of parallelism, reaching 1152.
- We propose a digital-domain BF16×1-b SRAM CIM macro. By sharing binary weights and duplicating 4T-XOR logic gates, this macro enables bit-parallel computing, reducing the compute cycle from 12 to 1 compared to bit-serial approaches. Additionally, the 4T-XOR logic gate design optimizes efficiency by reducing peripheral circuitry. This macro achieves a 12x throughput improvement with a 9.81x area overhead compared to previous CIM macros.

Furthermore, the additional 4-bit shifting space within the BF16×1-b SRAM CIM macro significantly reduces the computing error (standard deviation) by two orders of magnitude.

- We propose an innovative charge-based 5T2R1C ReRAM CIM macro. By leveraging the voltage division property of inverters, we effectively isolate the impact of ReRAM device variations, improving the accuracy of MAC computations at the unit-cell level. Measured results demonstrate highly consistent output characteristics across trials, with standard deviations of 0.0788mV for logic "0" and 0.3156mV for logic "1" under a 1V supply voltage. The exceptional linearity performance of our proposed ReRAM macro enables high parallelism of up to 4096, surpassing most existing CIM designs.

The proposed research comprehensively addresses Computing-in-Memory (CIM) across multiple hierarchies. On the architectural level, it encompasses both charge-based and digital-domain paradigms. On the device level, it investigates SRAM-based and ReRAM-based implementations. These accelerators are designed for modern deep learning workloads, notably layer-wise mixed-precision Convolutional Neural Networks (CNNs) and binarized BERT models, each exhibiting unique trade-offs in precision, noise immunity, throughput, and energy/area efficiency.

A central conclusion from this work is that no single CIM architecture is universally optimal. The design must be tailored to its operational context:

- Transformers & High-Accuracy Scenarios: Digital-domain, SRAM-based CIM is most suitable, prized for its operational robustness, programmability, compatibility with integer and floating-point operations, and seamless CMOS integration. SRAM's high endurance is critical for applications requiring frequent model updates.

- Ultra-Low-Power Edge Inference (e.g., Keyword Spotting, TinyML): Analog-domain, ReRAM-based CIM provides superior energy efficiency and storage density. Its non-volatility enables instant-on functionality and eliminates idle power, making it ideal for always-on systems, while its accuracy remains sufficient for binary or ternary networks.

- High-Throughput CNN Acceleration: Charge-based CIM, leveraging either SRAM or ReRAM arrays, is recommended due to the robustness of capacitive computing and its support for large row activations. Within this category, ReRAM offers higher density, whereas SRAM affords lower latency.

7.2 Future Work

While this research has made significant strides and achieved a series of notable results, there is still ample room for improvement and optimization due to limitations in design cycles and experimental conditions.

One promising future direction involves enabling the coupling of multiple CIM cores to accelerate large-scale neural networks. Our current accelerator designs, limited by area constraints, incorporate only a single in-memory computing array.

The proposed charge-based SRAM/ReRAM CIM core is primarily suited for CNN architectures such as VGG and ResNet, owing to its relatively rigid convolution kernel configuration. Meanwhile, the digital-domain SRAM-based CIM accelerator, though capable of supporting more complex models like Vision Transformer (e.g., ViT-base), suffers from significantly longer computational cycles due to its restricted array size and limited on-chip buffer capacity.

These limitations hinder the accelerators' ability to efficiently process large networks or computationally intensive convolutional layers, often requiring frequent off-chip data transfers. Such memory shuttling contradicts the fundamental goal of in-memory computing—to minimize data movement.

With the advent of chiplet technology, a viable path forward is to integrate multiple CIM cores using high-bandwidth, on-chip interconnects. By co-optimizing data flow through inter-layer and inter-core pipelining, this multi-core architecture could significantly reduce off-chip access and help realize the full potential of in-memory computing.

A second potential area of research involves enhancing the high-frequency performance and optimizing the capacitor array structure of the proposed sparsity-aware or cycle-reduced ADC. While this research primarily focuses on bit-level sparsity, frequency optimization and capacitor array optimization have received relatively less attention. Consequently, the ADC frequency can become a bottleneck for system throughput, and the capacitor array can consume a significant portion of chip area and power. Future research could concentrate on improving ADC frequency through high-frequency system design techniques to enhance performance metrics and overall throughput. Additionally, exploring structure-level optimizations for the capacitor array, such as segmented capacitor arrays and C2C capacitor arrays, could improve area and energy efficiency.

A third potential research direction involves integrating a batch-wise mantissa shifting mechanism into the activation quantization procedure of the Binarized BERT model. Since the proposed digital-domain accelerator incorporates a 4-bit shifting space, information loss can be avoided as long as the maximum exponent is not more than 2^4 times larger than the minimum exponent within a batch of 128 activations. By implementing this mechanism, the binarized BERT model can ensure lossless computation and simplify the shifting procedure, leading to improvements in both accuracy and energy efficiency.

The fourth potential optimization involves enabling more fine-grained dataflow management in chip design. The massive scale of CIM computing arrays, especially analog CIM arrays, in accelerators poses significant challenges to the placement and routing of the digital top-level design, compromising its spatial utilization. By refining dataflow strategies, future designs can mitigate these challenges and enhance overall area efficiency.

The fifth promising direction involves integrating the CIM accelerator into a heterogeneous System-on-Chip (SoC), a step that promises substantial gains in energy efficiency and inference latency for matrix-intensive operations such as neural network inference. However, this integration introduces significant challenges, shifting the focus from pure computational efficiency to system-level data management and orchestration. Key bottlenecks include severe bandwidth and latency constraints in the network-on-chip (NoC) and memory hierarchy, which can undermine the CIM core's energy advantages by incurring costly data transfer delays; the intricacies of ensuring memory coherence between the CIM accelerator and host processors such as RISC-V; and the absence of a unified software stack and programming model for efficient workload partitioning, task scheduling, and coordinated execution across CPUs, GPUs, and the CIM unit. Successfully addressing these issues demands a holistic hardware-software co-design approach, encompassing the interconnect, memory architecture, and compiler tools, to ensure that system-level overhead does not dominate performance and erase the benefits of in-memory computing.

The sixth promising future research direction is the integration of computing-in-memory (CIM) and processing-near-memory (PNM) architectures. Currently, companies such as Samsung, SK Hynix, and AMD have introduced HBM-based near-memory computing accelerators utilizing 3D stacking technology. These accelerators improve energy efficiency and performance by reducing the distance between memory and computing units, yet they fundamentally remain based on the separated storage-and-computation architecture.

In contrast, computing-in-memory architectures integrate storage and computation, fundamentally addressing the power consumption issues caused by data movement. With simpler circuitry and more streamlined computation processes, CIM can achieve higher energy efficiency and lower latency compared to near-memory approaches. Additionally, near-memory architectures are limited by memory bandwidth, resulting in lower throughput than CIM architectures.

However, near-memory architectures support higher-precision computations such as FP16, BF16, and FP32, and offer greater programmability and generality. Moreover, benefiting from mature HBM technology, current near-memory architectures can leverage advanced 2.5D/3D stacking and more advanced manufacturing processes, providing advantages in area efficiency.

Therefore, future work could focus on integrating computing-in-memory and near-memory architectures to efficiently cover a wider range of application scenarios. In low-power mode, CIM architectures can handle simple neural network inference to ensure basic functionality, while near-memory architectures can compensate for the shortcomings of CIM in high-precision and high-performance computing applications.

The seventh feasible future research direction is based on more mature multi-bit emerging non-volatile devices such as FeFET and PCM. Currently, the computing-in-memory architectures designed in the paper are all based on single-bit devices. If replaced with multi-bit devices, the number of shift-and-add operations required for precision reconstruction on input features can be reduced, which could exponentially improve system throughput and energy efficiency.

However, using multi-bit devices also introduces several challenges. First, in terms of device linearity, previously proposed techniques such as gate bootstrapping and error-isolation macro cells have limited effectiveness. Without special handling, significant computational errors may occur. Second, multi-bit devices cause an exponential increase in the number of output partial sum levels, drastically reducing the signal range between different levels and posing great challenges to the effective signal range supported by ADCs.

Therefore, future work can be divided into two directions. First, through hardware-software co-optimization, leveraging device characteristics and ADC circuit properties, use automated neural architecture search, quantization strategies, and pruning strategies to identify networks with high accuracy, high performance, and high robustness. Second, design highly robust macro cells based on multi-bit devices to mitigate accuracy degradation caused by poor device linearity.

References

[1] D. W. Otter, J. R. Medina, and J. K. Kalita, "A Survey of the Usages of Deep Learning for Natural Language Processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, pp. 604–624, 2021.

[2] W. Mao, K. Li, Q. Cheng, L. Dai, B. Li, X. Xie, H. Li, L. Lin, and H. Yu, "A Configurable Floating-Point Multiple-Precision Processing Element for HPC and AI Converged Computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 2, pp. 213–226, 2022.

[3] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[4] K. Feng, Z. Chen, F. Gao, Z. Wang, L. Xu, and W. Lin, "Post-Training Quantization for Vision Transformer in Transformed Domain," in *2023 IEEE International Conference on Multimedia and Expo (ICME)*, 2023, pp. 1457–1462.

[5] X. Dong *et al.*, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 4860–4874.

[6] Q. Liu, B. Gao, P. Yao, D. Wu, J. Chen, Y. Pang, W. Zhang, Y. Liao, C.-X. Xue, W.-H. Chen, J. Tang, Y. Wang, M.-F. Chang, H. Qian, and H. Wu, "A Fully Integrated Analog ReRAM Based 78.4TOPS/W Compute-In-Memory

Chip with Fully Parallel MAC Computing," in *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*, 2020, pp. 500–502.

[7] X. Si, J.-J. Chen, Y.-N. Tu, W.-H. Huang, J.-H. Wang, Y.-C. Chiu, W.-C. Wei, S.-Y. Wu, X. Sun, R. Liu, S. Yu, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, Q. Li, and M.-F. Chang, "A Twin-8T SRAM Computation-In-Memory Macro for Multiple-Bit CNN-Based Machine Learning," in *2019 IEEE International Solid-State Circuits Conference - (ISSCC)*, 2019, pp. 396–398.

[8] J. Song, X. Tang, H. Luo, H. Zhang, X. Qiao, Z. Sun, X. Yang, Z. Wu, Y. Wang, R. Wang, and R. Huang, "A 4-bit Calibration-Free Computing-In-Memory Macro With 3T1C Current-Programed Dynamic-Cascode Multi-Level-Cell eDRAM," *IEEE Journal of Solid-State Circuits*, vol. 59, no. 3, pp. 842–854, 2024.

[9] H. Zhang, S. He, X. Lu, X. Guo, S. Wang, Y. Du, and L. Du, "SSM-CIM: An Efficient CIM Macro Featuring Single-Step Multi-bit MAC Computation for CNN Edge Inference," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, no. 11, pp. 4357–4368, 2023.

[10] G. Shin, D. Seo, J. Kim, J. Rhe, E. Lee, S. Kim, S. Jeong, J. H. Ko, and Y. Lee, "A Charge-Domain Computation-In-Memory Macro with Versatile All-Around-Wire-Capacitor for Variable-Precision Computation and Array-Embedded DA/AD Conversions," in *ESSCIRC 2021 - IEEE 47th European Solid State Circuits Conference (ESSCIRC)*, 2021, pp. 123–126.

[11] D. Liu, H. Zhou, W. Mao, J. Liu, Y. Han, C. Man, Q. Wu, Z. Guo, M. Huang, S. Luo, M. Lv, Q. Chen, and H. Yu, "An Energy-Efficient Mixed-Bit CNN Accelerator With Column Parallel Readout for ReRAM-Based In-Memory Computing," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 12, no. 4, pp. 821–834, 2022.

[12] W. Mao, F. Li, J. Liu, R. Xiao, K. Huang, Y. Li, H. Yu, Y. Liu, and G. Han, "A Low-Power Charge-Domain Bit-Scalable Readout System for Fully-Parallel Computing-in-Memory Accelerators," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 71, no. 6, pp. 2916–2920, 2024.

[13] H. Wang, R. Liu, R. Dorrance, D. Dasalukunte, D. Lake, and B. Carlton, "A Charge Domain SRAM Compute-in-Memory Macro With C-2C Ladder-Based 8-Bit MAC Unit in 22-nm FinFET Process for Edge Inference," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 4, pp. 1037–1050, 2023.

[14] H. Zhang, Y. Du, and L. Du, "Linearity Analysis for Charge Domain In-memory Computing," in *2023 IEEE 15th International Conference on ASIC (ASICON)*, 2023, pp. 1–4.

[15] P.-C. Wu, J.-W. Su, Y.-L. Chung, L.-Y. Hong, J.-S. Ren, F.-C. Chang, Y. Wu, H.-Y. Chen, C.-H. Lin, H.-M. Hsiao, S.-H. Li, S.-S. Sheu, S.-C. Chang, W.-C. Lo, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, C.-I. Wu, and M.-F. Chang, "A 28nm 1Mb Time-Domain Computing-in-Memory 6T-SRAM Macro with a 6.6ns Latency, 1241GOPS and 37.01TOPS/W for 8b-MAC Operations for Edge-AI Devices," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 1–3.

[16] 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, D. Sun, M. E. Sinangil, Y.-H. Chen, T.-L. Chou, K. Akarvardar, H.-J. Liao, Y. Wang, M.-F. Chang, and T.-Y. J. Chang, "An 89TOPS/W and 16.3TOPS/mm2 All-Digital SRAM-Based Full-Precision Compute-In Memory Macro in 22nm for Machine-Learning Edge Applications," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 252–254.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[18] T. B. Brown, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[19] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2014, pp. 10–14.

[20] H. Wang, Z. Zhang, and S. Han, “Spatten: Efficient sparse attention architecture with cascade token and head pruning,” in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, pp. 97–110.

[21] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[23] A. Gholami, Z. Yao, S. Kim, C. Hooper, M. W. Mahoney, and K. Keutzer, “AI and Memory Wall,” *IEEE Micro*, vol. 44, no. 3, pp. 33–39, 2024.

[24] X. Dong, S. Chen, and S. J. Pan, “Learning to prune deep neural networks via layer-wise optimal brain surgeon,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS’17, Red Hook, NY, USA, 2017, p. 4860–4874.

[25] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, “Binary neural networks: A survey,” *Pattern Recognition*, vol. 105, p. 107281, 2020.

[26] H. Bai, W. Zhang, L. Hou, L. Shang, J. Jin, X. Jiang, Q. Liu, M. Lyu, and I. King, “BinaryBERT: Pushing the limit of BERT quantization,” in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Online, Aug. 2021, pp. 4334–4348.

[27] R. C. Minnick, “Cutpoint cellular logic,” *IEEE Transactions on Electronic Computers*, no. 6, pp. 685–698, 1964.

[28] W. H. Kautz, “Cellular logic-in-memory arrays,” *IEEE Transactions on Computers*, vol. 100, no. 8, pp. 719–727, 1969.

[29] D. Liu, H. Yu, and Y. Chai, "Low-power computing with neuromorphic engineering," *Advanced Intelligent Systems*, vol. 3, no. 2, p. 2000150, 2021.

[30] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature nanotechnology*, vol. 15, no. 7, pp. 529–544, 2020.

[31] H. You, W. Li, D. Shang, Y. Zhou, and S. Qiao, "A 1–8b Reconfigurable Digital SRAM Compute-in-Memory Macro for Processing Neural Networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 71, no. 4, pp. 1602–1614, 2024.

[32] J. Kim and J. Park, "The Quantitative Comparisons of Analog and Digital SRAM Compute-In-Memories for Deep Neural Network Applications," in *2022 19th International SoC Design Conference (ISOCC)*, 2022, pp. 129–130.

[33] W. Sun, J. Yue, Y. He, Z. Huang, J. Wang, W. Jia, Y. Li, L. Lei, H. Jia, and Y. Liu, "A Survey of Computing-in-Memory Processor: From Circuit to Application," *IEEE Open Journal of the Solid-State Circuits Society*, vol. 4, pp. 25–42, 2024.

[34] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-Tile 2.4-Mb In-Memory-Computing CNN Accelerator Employing Charge-Domain Compute," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 6, pp. 1789–1799, 2019.

[35] H. Oh, H. Kim, D. Ahn, J. Park, Y. Kim, I. Lee, and J.-J. Kim, "Energy-Efficient In-Memory Binary Neural Network Accelerator Design Based on 8T2C SRAM Cell," *IEEE Solid-State Circuits Letters*, vol. 5, pp. 70–73, 2022.

[36] Y. He, J. Yue, X. Feng, Y. Huang, H. Jia, J. Wang, L. Zhang, W. Sun, H. Yang, and Y. Liu, "An RRAM-Based Digital Computing-in-Memory Macro With Dynamic Voltage Sense Amplifier and Sparse-Aware Approximate Adder Tree," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 2, pp. 416–420, 2023.

[37] H. Kim, T. Yoo, T. T.-H. Kim, and B. Kim, "Colonnade: A Reconfigurable SRAM-Based Digital Bit-Serial Compute-In-Memory Macro for Processing

Neural Networks," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 7, pp. 2221–2233, 2021.

[38] L. Ni, H. Huang, Z. Liu, R. V. Joshi, and H. Yu, "Distributed In-Memory Computing on Binary RRAM Crossbar," *ACM Journal on Emerging Technologies in Computing System*, vol. 13, no. 36, pp. 1–18, 2017.

[39] W. Mao, Z. Xiao, P. Xu, H. Ren, D. Liu, S. Zhao, F. An, and H. Yu, "Energy-Efficient Machine Learning Accelerator for Binary Neural Networks," in *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, ser. GLSVLSI '20, New York, NY, USA, 2020, p. 77–82.

[40] J. Zhang, Z. Wang, and N. Verma, "In-Memory Computation of a Machine-Learning Classifier in a Standard 6T SRAM Array," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 4, pp. 915–924, 2017.

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

[42] M. E. Sinangil, B. Erbagci, R. Naous, K. Akarvardar, D. Sun, W.-S. Khwa, H.-J. Liao, Y. Wang, and J. Chang, "A 7-nm Compute-in-Memory SRAM Macro Supporting Multi-Bit Input, Weight and Output and Achieving 351 TOPS/W and 372.4 GOPS," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 1, pp. 188–198, 2021.

[43] S. Xie, C. Ni, A. Sayal, P. Jain, F. Hamzaoglu, and J. P. Kulkarni, "eDRAM-CIM: Compute-In-Memory Design with Reconfigurable Embedded-Dynamic-Memory Array Realizing Adaptive Data Converters and Charge-Domain Computing," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 248–250.

[44] C.-X. Xue, W.-H. Chen, J.-S. Liu, J.-F. Li, W.-Y. Lin, W.-E. Lin, J.-H. Wang, W.-C. Wei, T.-Y. Huang, T.-W. Chang, T.-C. Chang, H.-Y. Kao, Y.-C. Chiu, C.-Y. Lee, Y.-C. King, C.-J. Lin, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, and M.-F. Chang, "Embedded 1-Mb ReRAM-Based Computing-in-Memory Macro

With Multibit Input and Weight for CNN-Based AI Edge Processors," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 1, pp. 203–215, 2020.

[45] F. Tu, Y. Wang, Z. Wu, L. Liang, Y. Ding, B. Kim, L. Liu, S. Wei, Y. Xie, and S. Yin, "ReDCIM: Reconfigurable Digital Computing- In -Memory Processor With Unified FP/INT Pipeline for Cloud AI Acceleration," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 1, pp. 243–255, 2023.

[46] C. Man, C. Chang, C. Ding, A. Shen, H. Ren, Z. Guan, Y. Cheng, S. Luo, R. Zhang, N. Wong, and H. Yu, "RankSearch: An Automatic Rank Search Towards Optimal Tensor Compression for Video LSTM Networks on Edge," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2023, pp. 1–2.

[47] M. Huang, Y. Liu, C. Man, K. Li, Q. Cheng, W. Mao, and H. Yu, "A High Performance Multi-Bit-Width Booth Vector Systolic Accelerator for NAS Optimized Deep Learning Neural Networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 9, pp. 3619–3631, 2022.

[48] Z. He, A. Sheri, Q. Li, Q. Cheng, and H. Yu, "Agile Hardware and Software Co-design for RISC-V-based Multi-precision Deep Learning Microprocessor," in *2023 28th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2023, pp. 490–495.

[49] K. Li, H. Huang, M. Huang, C. Ding, L. Lin, L. Ni, and H. Yu, "A 29.12 TOPS/W and 1.13 TOPS/mm2 NAS-Optimized Mixed-Precision DNN Accelerator with Vector Split-and-Combination Systolic in 28nm CMOS," in *2024 IEEE Custom Integrated Circuits Conference (CICC)*, 2024, pp. 1–2.

[50] Y. C. H. L. Huanrui Yang, Lin Duan, "BSQ: Exploring Bit-Level Sparsity for Mixed-Precision Neural Network Quantization," in *International Conference on Learning Representations*, 2021.

[51] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, and L. Kaiser, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[53] B. Wu, Y. Wang, P. Zhang, Y. Tian, P. Vajda, and K. Keutzer, "Mixed precision quantization of convnets via differentiable neural architecture search," in *2019 International Conference on Learning Representations (ICLR)*, 2019.

[54] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[55] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[56] J. Zhang and D. Tao, "Empowering Things With Intelligence: A Survey of the Progress, Challenges, and Opportunities in Artificial Intelligence of Things," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 7789–7817, 2021.

[57] J. Yue, Z. Yuan, X. Feng, Y. He, Z. Zhang, X. Si, R. Liu, M.-F. Chang, X. Li, H. Yang, and Y. Liu, "A 65nm Computing-in-Memory-Based CNN Processor with 2.9-to-35.8TOPS/W System Energy Efficiency Using Dynamic-Sparsity Performance-Scaling Architecture and Energy-Efficient Inter/Intra-Macro Data Reuse," in *2020 IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 234–236.

[58] J.-W. Su, Y.-C. Chou, R. Liu, T.-W. Liu, P.-J. Lu, P.-C. Wu, Y.-L. Chung, L.-Y. Hung, J.-S. Ren, T. Pan, S.-H. Li, S.-C. Chang, S.-S. Sheu, W.-C. Lo, C.-I. Wu, X. Si, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, and M.-F. Chang, "A 28nm 384kb 6T-SRAM Computation-in-Memory Macro with 8b Precision for AI Edge Chips," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 250–252.

[59] H. Jia, M. Ozatay, Y. Tang, H. Valavi, R. Pathak, J. Lee, and N. Verma, "A Programmable Neural-Network Inference Accelerator Based on Scalable In-Memory Computing," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 236–238.

[60] X. Qiao, J. Song, X. Tang, H. Luo, N. Pan, X. Cui, R. Wang, and Y. Wang, “A 65 nm 73 kb SRAM-Based Computing-In-Memory Macro With Dynamic-Sparsity Controlling,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 6, pp. 2977–2981, 2022.

[61] J. Yue, Y. Liu, Z. Yuan, X. Feng, Y. He, W. Sun, Z. Zhang, X. Si, R. Liu, Z. Wang, M.-F. Chang, C. Dou, X. Li, M. Liu, and H. Yang, “STICKER-IM: A 65 nm Computing-in-Memory NN Processor Using Block-Wise Sparsity Optimization and Inter/Intra-Macro Data Reuse,” *IEEE Journal of Solid-State Circuits*, vol. 57, no. 8, pp. 2560–2573, 2022.

[62] J. Meng, S. K. Venkataramanaiah, C. Zhou, P. Hansen, P. Whatmough, and J.-s. Seo, “FixyFPGA: Efficient FPGA Accelerator for Deep Neural Networks with High Element-Wise Sparsity and without External Memory Access,” in *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*, 2021, pp. 9–16.

[63] J.-H. Kim, J. Lee, J. Lee, J. Heo, and J.-Y. Kim, “Z-PIM: A Sparsity-Aware Processing-in-Memory Architecture With Fully Variable Weight Bit-Precision for Energy-Efficient Deep Neural Networks,” *IEEE Journal of Solid-State Circuits*, vol. 56, no. 4, pp. 1093–1104, 2021.

[64] C. Gong, Z. Jiang, D. Wang, Y. Lin, Q. Liu, and D. Z. Pan, “Mixed Precision Neural Architecture Search for Energy Efficient Deep Learning,” in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019, pp. 1–7.

[65] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, “HAQ: Hardware-Aware Automated Quantization With Mixed Precision,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, USA. June, 2019.

[66] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” *5th International Conference on Learning Representations (ICLR)*, Toulon, France. April, 2016.

[67] Y. He, X. Zhang, and J. Sun, "Channel Pruning for Accelerating Very Deep Neural Networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 1398–1406.

[68] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "NISP: Pruning Networks Using Neuron Importance Score Propagation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, USA. June, 2018.

[69] W. Mao, L. Dai, K. Li, Q. Cheng, Y. Wang, L. Du, S. Luo, M. Huang, and H. Yu, "An Energy-Efficient Mixed-Bitwidth Systolic Accelerator for NAS-Optimized Deep Neural Networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 12, pp. 1878–1890, 2022.

[70] A. Kusupati, V. Ramanujan, R. Somani, M. Wortsman, P. Jain, S. Kakade, and A. Farhadi, "Soft threshold weight reparameterization for learnable sparsity," in *Proceedings of the 37th International Conference on Machine Learning*, ser. ICML'20, 2020.

[71] V. Sehwag, S. Wang, P. Mittal, and S. Jana, "Hydra: Pruning adversarially robust neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19 655–19 666, 2020.

[72] A. Guo, X. Chen, F. Dong, J. Chen, Z. Yuan, X. Hu, Y. Zhang, J. Zhang, Y. Tang, Z. Zhang, G. Chen, D. Yang, Z. Zhang, L. Ren, T. Xiong, B. Wang, B. Liu, W. Shan, X. Liu, H. Cai, G. Sun, J. Yang, and X. Si, "A 22nm 64kb Lightning-Like Hybrid Computing-in-Memory Macro with a Compressed Adder Tree and Analog-Storage Quantizers for Transformer and CNNs," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 67, 2024, pp. 570–572.

[73] Y. Yuan, Y. Yang, X. Wang, X. Li, C. Ma, Q. Chen, M. Tang, X. Wei, Z. Hou, J. Zhu, H. Wu, Q. Ren, G. Xing, P.-I. Mak, and F. Zhang, "A 28nm 72.12TFLOPS/W Hybrid-Domain Outer-Product Based Floating-Point SRAM Computing-in-Memory Macro with Logarithm Bit-Width Residual

ADC," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 67, 2024, pp. 576–578.

[74] J. Bai, S. Sun, W. Zhao, and W. Kang, "CIMQ: A Hardware-Efficient Quantization Framework for Computing-In-Memory-Based Neural Network Accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 1, pp. 189–202, 2024.

[75] M. Ali, I. Chakraborty, S. Choudhary, M. Chang, D. E. Kim, A. Raychowdhury, and K. Roy, "A 65 nm 1.4-6.7 TOPS/W Adaptive-SNR Sparsity-Aware CIM Core with Load Balancing Support for DL workloads," in *2023 IEEE Custom Integrated Circuits Conference (CICC)*, 2023, pp. 1–2.

[76] W. Mao, Y. Li, C.-H. Heng, and Y. Lian, "A Low Power 12-bit 1-kS/s SAR ADC for Biomedical Signal Processing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 2, pp. 477–488, 2019.

[77] D. M. Mathew, A. L. Chinazzo, C. Weis, M. Jung, B. Giraud, P. Vivet, A. Levisse, and N. Wehn, "RRAMSpec: A Design Space Exploration Framework for High Density Resistive RAM," in *Embedded Computer Systems: Architectures, Modeling, and Simulation*, Cham, 2019, pp. 34–47.

[78] W. Zhang, L. Hou, Y. Yin, L. Shang, X. Chen, X. Jiang, and Q. Liu, "TernaryBERT: Distillation-aware ultra-low bit BERT," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, Nov. 2020, pp. 509–521.

[79] Z. Guan, H. Huang, Y. Su, H. Huang, N. Wong, and H. Yu, "APTQ: Attention-aware post-training mixed-precision quantization for large language models," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, New York, NY, USA, 2024.

[80] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 14–26.

[81] S. Shen, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, “Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, Apr. 2020, pp. 8815–8821.

[82] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat, “Q8BERT: Quantized 8Bit BERT,” in *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS).* IEEE, Dec. 2019.

[83] H. Jia, H. Valavi, Y. Tang, J. Zhang, and N. Verma, “A Programmable Heterogeneous Microprocessor Based on Bit-Scalable In-Memory Computing,” *IEEE Journal of Solid-State Circuits*, vol. 55, no. 9, pp. 2609–2621, 2020.

[84] A. Guo, X. Si, X. Chen, F. Dong, X. Pu, D. Li, Y. Zhou, L. Ren, Y. Xue, X. Dong, H. Gao, Y. Zhang, J. Zhang, Y. Kong, T. Xiong, B. Wang, H. Cai, W. Shan, and J. Yang, “A 28nm 64-kb 31.6-TFLOPS/W Digital-Domain Floating-Point-Computing-Unit and Double-Bit 6T-SRAM Computing-in-Memory Macro for Floating-Point CNNs,” in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, 2023, pp. 128–130.

[85] D. Wang, C.-T. Lin, G. K. Chen, P. Knag, R. K. Krishnamurthy, and M. Seok, “DIMC: 2219TOPS/W 2569F2/b Digital In-Memory Computing Macro in 28nm Based on Approximate Arithmetic Hardware,” in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 266–268.

[86] F. Tu, Z. Wu, Y. Wang, L. Liang, L. Liu, Y. Ding, L. Liu, S. Wei, Y. Xie, and S. Yin, “TranCIM: Full-Digital Bitline-Transpose CIM-based Sparse Transformer Accelerator With Pipeline/Parallel Reconfigurable Modes,” *IEEE Journal of Solid-State Circuits*, vol. 58, no. 6, pp. 1798–1809, 2023.

[87] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, “Large scale distributed deep networks,” *Advances in neural information processing systems*, vol. 25, 2012.

[88] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: a system for large-scale machine learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'16, USA, 2016, p. 265–283.

[89] G. Tagliavini, S. Mach, D. Rossi, A. Marongiu, and L. Benini, "A transprecision floating-point platform for ultra-low power computing," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2018, pp. 1051–1056.

[90] P.-C. Wu, J.-W. Su, L.-Y. Hong, J.-S. Ren, C.-H. Chien, H.-Y. Chen, C.-E. Ke, H.-M. Hsiao, S.-H. Li, S.-S. Sheu, W.-C. Lo, S.-C. Chang, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, and M.-F. Chang, "A 22nm 832Kb Hybrid-Domain Floating-Point SRAM In-Memory-Compute Macro with 16.2-70.2TFLOPS/W for High-Accuracy AI-Edge Devices," in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, 2023, pp. 126–128.

[91] T.-H. Wen, H.-H. Hsu, W.-S. Khwa, W.-H. Huang, Z.-E. Ke, Y.-H. Chin, H.-J. Wen, Y.-C. Chang, W.-T. Hsu, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, S.-H. Teng, C.-C. Chou, Y.-D. Chih, T.-Y. J. Chang, and M.-F. Chang, "A 22nm 16Mb Floating-Point ReRAM Compute-in-Memory Macro with 31.2TFLOPS/W for AI Edge Devices," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 67, 2024, pp. 580–582.

[92] H.-T. Kung, "Why systolic architectures?" *Computer*, vol. 15, no. 1, pp. 37–46, 1982.

[93] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le,

C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-datacenter performance analysis of a tensor processing unit," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, 2017, pp. 1–12.

[94] M. Huang, J. Luo, C. Ding, Z. Wei, S. Huang, and H. Yu, "An Integer-Only and Group-Vector Systolic Accelerator for Efficiently Mapping Vision Transformer on Edge," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, no. 12, pp. 5289–5301, 2023.

[95] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 367–379.

[96] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 292–308, 2019.

[97] N. Verma, H. Jia, H. Valavi, Y. Tang, M. Ozatay, L.-Y. Chen, B. Zhang, and P. Deaville, "In-Memory Computing: Advances and Prospects," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 3, pp. 43–55, 2019.

[98] E. Lee, T. Han, D. Seo, G. Shin, J. Kim, S. Kim, S. Jeong, J. Rhe, J. Park, J. H. Ko, and Y. Lee, "A Charge-Domain Scalable-Weight In-Memory Computing Macro With Dual-SRAM Architecture for Precision-Scalable DNN Accelerators," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 8, pp. 3305–3316, 2021.

[99] X. Peng and S. Yu, "Benchmark of RRAM based architectures for dot-product

computation," in *2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. IEEE, 2018, pp. 378–381.

[100] S. Yu, "Neuro-Inspired Computing with Emerging Nonvolatile Memorys," *Proceedings of the IEEE*, vol. 106, no. 2, pp. 260–285, 2018.

[101] W. Wan, R. Kubendran, C. Schaefer, S. B. Eryilmaz, W. Zhang, D. Wu, S. Deiss, P. Raina, H. Qian, B. Gao, S. Joshi, H. Wu, H.-S. P. Wong, and G. Cauwenberghs, "A compute-in-memory chip based on resistive random-access memory," *Nature*, vol. 608, no. 7923, pp. 504–512, 2022.

[102] L. Wang, W. Ye, J. Lai, J. Liu, J. Yang, X. Si, C. Huo, C. Dou, X. Xu, Q. Liu *et al.*, "A 14nm 100Kb 2T1R Transpose RRAM with>150X resistance ratio enhancement and 27.95% reduction on energy-latency product using low-power near threshold read operation and fast data-line current stabling scheme," in *2021 Symposium on VLSI Technology*. IEEE, 2021, pp. 1–2.

[103] C.-X. Xue, T.-Y. Huang, J.-S. Liu, T.-W. Chang, H.-Y. Kao, J.-H. Wang, T.-W. Liu, S.-Y. Wei, S.-P. Huang, W.-C. Wei, Y.-R. Chen, T.-H. Hsu, Y.-K. Chen, Y.-C. Lo, T.-H. Wen, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, and M.-F. Chang, "A 22nm 2Mb ReRAM Compute-in-Memory Macro with 121-28TOPS/W for Multibit MAC Computing for Tiny AI Edge Devices," in *2020 IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 244–246.

[104] Y. Yang, S. You, H. Li, F. Wang, C. Qian, and Z. Lin, "Towards Improving the Consistency, Efficiency, and Flexibility of Differentiable Neural Architecture Search," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, USA. June, 2021.

[105] B. Yan, J.-L. Hsu, P.-C. Yu, C.-C. Lee, Y. Zhang, W. Yue, G. Mei, Y. Yang, Y. Yang, H. Li, Y. Chen, and R. Huang, "A 1.041-Mb/mm2 27.38-TOPS/W Signed-INT8 Dynamic-Logic-Based ADC-less SRAM Compute-in-Memory Macro in 28nm with Reconfigurable Bitwise Operation for AI and Embedded Applications," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 188–190.

[106] L. Wang, W. Ye, C. Dou, X. Si, X. Xu, J. Liu, D. Shang, J. Gao, F. Zhang, Y. Liu, M.-F. Chang, and Q. Liu, "Efficient and Robust Nonvolatile Computing-In-Memory Based on Voltage Division in 2T2R RRAM With Input-Dependent Sensing Control," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 5, pp. 1640–1644, 2021.

[107] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, USA. June, 2019.

[108] A. Ren, T. Zhang, S. Ye, J. Li, W. Xu, X. Qian, X. Lin, and Y. Wang, "ADMM-NN: An Algorithm-Hardware Co-Design Framework of DNNs Using Alternating Direction Methods of Multipliers," in *2019 Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Providence, USA. April, 2019.

[109] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid, "Towards Effective Low-Bitwidth Convolutional Neural Networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[110] S. Yin, P. Ouyang, J. Yang, T. Lu, X. Li, L. Liu, and S. Wei, "An Energy-Efficient Reconfigurable Processor for Binary-and Ternary-Weight Neural Networks With Flexible Data Bit Width," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 4, pp. 1120–1136, 2019.

[111] K. Li, J. Zhou, Y. Wang, J. Luo, Z. Yang, S. Yang, W. Mao, M. Huang, and H. Yu, "A Precision-Scalable Energy-Efficient Bit-Split-and-Combination Vector Systolic Accelerator for NAS-Optimized DNNs on Edge," in *2022 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2022, pp. 730–735.

[112] L. Dai, Q. Cheng, Y. Wang, G. Huang, J. Zhou, K. Li, W. Mao, and H. Yu, "An Energy-Efficient Bit-Split-and-Combination Systolic Accelerator

for NAS-Based Multi-Precision Convolution Neural Networks," in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2022, pp. 448–453.

[113] Q. Zheng, Z. Wang, Z. Feng, B. Yan, Y. Cai, R. Huang, Y. Chen, C.-L. Yang, and H. H. Li, "Lattice: An ADC/DAC-less ReRAM-based Processing-In-Memory Architecture for Accelerating Deep Convolution Neural Networks," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[114] P.-Y. Chen and S. Yu, "Compact Modeling of RRAM Devices and Its Applications in 1T1R and 1S1R Array Design," *IEEE Transactions on Electron Devices*, vol. 62, no. 12, pp. 4022–4028, 2015.

[115] M. Pelgrom, A. Duinmaijer, and A. Welbers, "Matching properties of MOS transistors," *IEEE Journal of Solid-State Circuits*, vol. 24, no. 5, pp. 1433–1439, 1989.

[116] W.-S. Khwa, Y.-C. Chiu, C.-J. Jhang, S.-P. Huang, C.-Y. Lee, T.-H. Wen, F.-C. Chang, S.-M. Yu, T.-Y. Lee, and M.-F. Chang, "A 40-nm, 2M-Cell, 8b-Precision, Hybrid SLC-MLC PCM Computing-in-Memory Macro with 20.5 - 65.0TOPS/W for Tiny-Al Edge Devices," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 1–3.

[117] D. Kim, C. Yu, S. Xie, Y. Chen, J.-Y. Kim, B. Kim, J. P. Kulkarni, and T. T.-H. Kim, "An Overview of Processing-in-Memory Circuits for Artificial Intelligence and Machine Learning," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 12, no. 2, pp. 338–353, 2022.

[118] S. Zhang, K. Huang, and H. Shen, "A Robust 8-Bit Non-Volatile Computing-in-Memory Core for Low-Power Parallel MAC Operations," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 6, pp. 1867–1880, 2020.

[119] Y. Zhang, K. Huang, R. Xiao, B. Wang, Y. Xu, J. Fan, and H. Shen, "An 8-bit in resistive memory computing core with regulated passive neuron and

bitline weight mapping," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 4, pp. 379–391, 2022.

[120] C.-X. Xue, J.-M. Hung, H.-Y. Kao, Y.-H. Huang, S.-P. Huang, F.-C. Chang, P. Chen, T.-W. Liu, C.-J. Jhang, C.-I. Su, W.-S. Khwa, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, Y.-D. Chih, T.-Y. J. Chang, and M.-F. Chang, "A 22nm 4Mb 8b-Precision ReRAM Computing-in-Memory Macro with 11.91 to 195.7TOPS/W for Tiny AI Edge Devices," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 245–247.

[121] X. Peng, R. Liu, and 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.

[122] P. Chen, M. Wu, W. Zhao, J. Cui, Z. Wang, Y. Zhang, Q. Wang, J. Ru, L. Shen, T. Jia, Y. Ma, L. Ye, and R. Huang, "A 22nm Delta-Sigma Computing-In-Memory (ΔΣCIM) SRAM Macro with Near-Zero-Mean Outputs and LSB-First ADCs Achieving 21.38TOPS/W for 8b-MAC Edge AI Processing," in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, 2023, pp. 140–142.

[123] X. Si, Y.-N. Tu, W.-H. Huang, J.-W. Su, P.-J. Lu, J.-H. Wang, T.-W. Liu, S.-Y. Wu, R. Liu, Y.-C. Chou, Z. Zhang, S.-H. Sie, W.-C. Wei, Y.-C. Lo, T.-H. Wen, T.-H. Hsu, Y.-K. Chen, W. Shih, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, N.-C. Lien, W.-C. Shih, Y. He, Q. Li, and M.-F. Chang, "A 28nm 64Kb 6T SRAM Computing-in-Memory Macro with 8b MAC Operation for AI Edge Chips," in *2020 IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 246–248.

[124] Y. He, H. Diao, C. Tang, W. Jia, X. Tang, Y. Wang, J. Yue, X. Li, H. Yang, H. Jia, and Y. Liu, "A 28nm 38-to-102-TOPS/W 8b Multiply-Less Approximate Digital SRAM Compute-In-Memory Macro for Neural-Network Inference," in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, 2023, pp. 130–132.

[125] A. Stillmaker and B. Baas, “Scaling equations for the accurate prediction of CMOS device performance from 180nm to 7nm,” *Integration*, vol. 58, pp. 74–81, 2017.

[126] T. Tambe, J. Zhang, C. Hooper, T. Jia, P. N. Whatmough, J. Zuckerman, M. C. D. Santos, E. J. Loscalzo, D. Giri, K. Shepard, L. Carloni, A. Rush, D. Brooks, and G.-Y. Wei, “A 12nm 18.1TFLOPs/W Sparse Transformer Processor with Entropy-Based Early Exit, Mixed-Precision Predication and Fine-Grained Power Management,” in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, 2023, pp. 342–344.

[127] B. Keller, R. Venkatesan, S. Dai, S. G. Tell, B. Zimmer, W. J. Dally, C. Thomas Gray, and B. Khailany, “A 17–95.6 TOPS/W Deep Learning Inference Accelerator with Per-Vector Scaled 4-bit Quantization for Transformers in 5nm,” in *2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, 2022, pp. 16–17.

[128] F. Tu, Z. Wu, Y. Wang, L. Liang, L. Liu, Y. Ding, L. Liu, S. Wei, Y. Xie, and S. Yin, “A 28nm 15.59µJ/Token Full-Digital Bitline-Transpose CIM-Based Sparse Transformer Accelerator with Pipeline/Parallel Reconfigurable Modes,” in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 466–468.

[129] R. Guo, X. Chen, L. Wang, Y. Wang, H. Sun, J. Wei, H. Han, L. Liu, S. Wei, Y. Hu, and S. Yin, “CIMFormer: A Systolic CIM-Array-Based Transformer Accelerator With Token-Pruning-Aware Attention Reformulating and Principal Possibility Gathering,” *IEEE Journal of Solid-State Circuits*, vol. 59, no. 10, pp. 3317–3329, 2024.

[130] C. Lammie and M. R. Azghadi, “MemTorch: A Simulation Framework for Deep Memristive Cross-Bar Architectures,” in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, Seville, Spain. Oct, 2020.