Energy-Efficient Circuit and Architecture Designs for Intelligent Systems

by

Shihui Yin

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved October 2020 by the
Graduate Supervisory Committee:

Jae-sun Seo, Chair
Yu Cao
Sarma Vrudhula
Chaitali Chakrabarti

ARIZONA STATE UNIVERSITY

December 2020

ABSTRACT

In the era of artificial intelligent (AI), deep neural networks (DNN) have achieved accuracy on par with humans on a variety of recognition tasks. However, the high computation and storage requirement of DNN training and inference have posed challenges to deploying or locally training the DNNs on mobile and wearable devices. Energy-efficient hardware innovation from circuit to architecture level is required.

In this dissertation, a smart electrocardiogram (ECG) processor is first presented for ECG-based authentication as well as cardiac monitoring. The 65nm testchip consumes 1.06 μW at 0.55 V for real-time ECG authentication achieving equal error rate of 1.7% for authentication on an in-house 645-subject database.

Next, a couple of SRAM-based in-memory computing (IMC) accelerators for deep learning algorithms are presented. Two single-array macros titled XNOR-SRAM and C3SRAM are based on resistive and capacitive networks for XNOR-ACcumulation (XAC) operations, respectively. XNOR-SRAM and C3SRAM macros in 65nm CMOS achieve energy efficiency of 403 TOPS/W and 672 TOPS/W, respectively. Built on top of these two single-array macro designs, two multi-array architectures are presented. The XNOR-SRAM based architecture titled "Vesti" is designed to support configurable multibit activations and large-scale DNNs seamlessly. Vesti employs double-buffering with two groups of in-memory computing SRAMs, effectively hiding the write latency of IMC SRAMs. The Vesti accelerator in 65nm CMOS achieves energy consumption of $<$ 20 nJ for MNIST classification and $<$ 40 μJ for CIFAR-10 classification at 1.0 V supply. More recently, a programmable IMC accelerator (PIMCA) integrating 108 C3SRAM macros of a total size of 3.4 Mb is proposed.The 28nm prototype chip achieves system-level energy efficiency of 437/62 TOPS/W at 40 MHz, 1 V supply for DNNs with 1b/2b precision.

In addition to the IMC works, this dissertation also presents a convolutional neural

network (CNN) learning processor, which accelerates the stochastic gradient descent (SGD) with momentum based training algorithm in 16-bit fixed-point precision. The 65nm CNN learning processor achieves peak energy efficiency of 2.6 TOPS/W for 16-bit fixed-point operations, consuming 10.45 mW at 0.55 V.

In summary, in this dissertation, several hardware innovations from circuit to architecture level are presented, exploiting the reduced algorithm complexity with pruning and low-precision quantization techniques. In particular, macro-level and system-level SRAM based IMC works presented in this dissertation show that SRAM based IMC is one of the promising solutions for energy-efficient intelligent systems.

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my PhD advisor Dr. Jae-sun Seo for the continuous guidance and support through my PhD study. I have learned a lot from him in research skills, writing and presentation. This dissertation would not have been possible Without his mentoring and persistent help. I am also grateful to the rest of my dissertation committee: Dr. Yu Cao, Dr. Sarma Vrudhula and Dr. Chaitali Chakrabarti for their guidance in several of my research projects and insightful comments and suggestions on my dissertation. In addition, I am also grateful for Dr. Mingoo Seok (Associate Professor, Department of Electrical Engineering at Columbia University) and Dr. Bipin Rajendran (Associate Professor, Department of Electrical and Computer Engineering at New Jersey Institute of Technology) for their guidance and help in a number of collaborating research projects that I have participated in.

My sincere thanks also go to Dr. Jinwook Oh, Dr. Jungwook Choi, Dr. Kyuhyoun Kim, Dr. Naigang Wang, Dr. Sunil K Shukla and Dr. Kailash Gopalakrishnan for their valuable guidance and encouragement during my summer research internship at IBM TJ Watson Research Center.

I would also like to thank for my fellow colleagues at ASU for the help and encouragement in the past five years: Dr. Minkyu Kim, Dr. Deepak Kadetotad, Xiaoyang Mi, Usama Awais, Shreyas Venkataramanaiah, Sai Kiran Cherupally, Jyotishman Saikia, Jian Meng, Wangxin He, Han-sok Suh, Dr. Zihan Xu, Dr. Abinahs Mohanty, Xiaocong Du, Zheng Li, Dr. Yufei Ma, Dr. Pai-Yu Chen, Dr. Rui Liu, Xiaoyu Sun, Xiaochen Peng. In addition, I would like to thank Dr. Zhewei Jiang (Columbia University), Bo Zhang (Columbia University) and Dr. Shruti R Kulkarni (New Jersey Institute of Technology) for their inspiring discussions and persistent help in different projects that we have worked on together.

Last but not the least, I owe a lot of thanks to my parents and my sister who have always been giving me spiritual support and supporting all the decisions I made throughout my life. I dedicated this dissertation to my family.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION

Deep neural networks (DNNs) and convolutional neural networks (CNNs) have unprecedentedly improved the accuracies in large-scale recognition tasks [1–6]. However, the arithmetic complexity and memory access have limited the energy-efficiency and acceleration of DNN hardware [7–11]. Innovation in both algorithm and hardware design is required to address this issue.

On the algorithm side, two major methodologies have been proposed in the past decade to reduce the neural network complexity: **network pruning** and **network quantization**. By pruning a large number of less important parameters in the neural network without hurting the classification accuracy, we can reduce the memory footprint for storing the parameters and also greatly reduce the total number of operations [12–15]. Quantizing neural network activations and/or weights can also simplify the required computation complexity of DNNs. In recent algorithms, weights and neuron activations are binarized to +1 or -1 [16, 17] such that the multiplication between an weight and an activation becomes an XNOR operation and the accumulation of the XNOR operations becomes bitcount of those XNOR results. Although the initial XNOR-Net [16] showed a relatively large test accuracy degradation ($\sim$10-20%) for the ImageNet dataset, recent works that employ 2-bit precision [18] have shown 1–3% accuracy degradation for ImageNet.

On the hardware side, a number of digital ASIC designs [7, 8, 19, 20] have been presented to accelerate the inference or classification phase of DNNs with enhanced energy-efficiency. Below we briefly introduce the representative ASIC accelerators.

Eyeriss [7] proposed a new dataflow called Row Stationary (RS) on a spatial ar-

chitecture with 168 processing elements. RS dataflow reconfigures the computation mapping of a given shape, which optimizes energy efficiency by maximally reusing data locally to reduce expensive data movement, such as DRAM accesses. ENVISION [8] presented an energy-scalable CNN processor achieving efficiencies up to 10 TOPS/W, where the data precision of MAC units can be dynamically scaled from 4-bit to 16-bit, together with voltage/frequency scaling and body bias modulation. DNPU [19] demonstrated a reconfigurable processor that can efficiently map both CNN and Recurrent Neural Network (RNN) algorithms. It employed dynamic fixed-point precision with online adaptation via overflow monitoring. It also proposed a quantization table based matrix multiplication to reduce off-chip memory access. In [20], an accelerator for fully-connected DNNs is presented, featuring circuit and algorithmic error resilience with in-situ timing error detection and correction techniques (also known as Razor [21]).

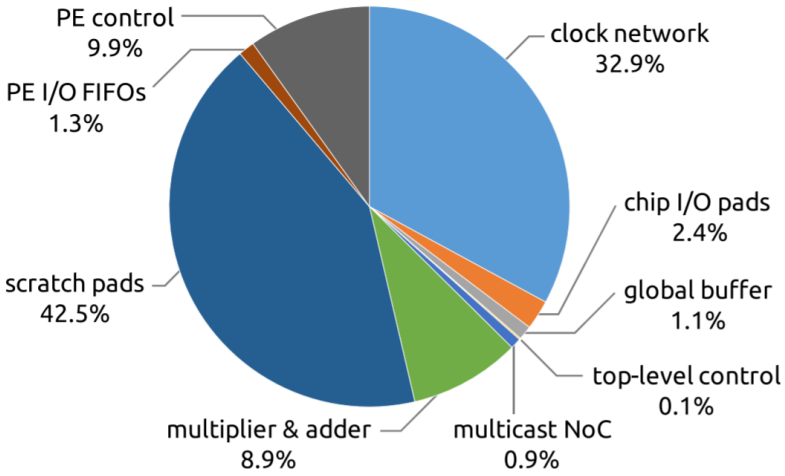


Figure 1.1: Power breakdown of Eyeriss chip [7] running the first convolution layer of AlexNet CNN [1]. Besides the clock network, most power is consumed by on-chip SRAM (scratch pads, global buffer).

In the Eyeriss chip results [7] of running AlexNet CNN [1], Fig. 1.1 shows that the

Arithmetic Logic Unit (ALU) operations only accounted for less than 10% of the total power. Memory access and data movement related components, including on-chip SRAMs, buffers, and network-on-chip, accounted for up to 45%. This confirms that memory access and data movement are actually significantly more energy consuming than ALU operations. Eyeriss chip consumes relatively high clock power (33% of total power), but the authors acknowledged that this can be considerably reduced with proper clock-gating techniques [7]. Therefore, we can conclude that the on-chip SRAMs indeed dominated on-chip power consumption, which infers that in-memory computing has a large potential to effectively reduce memory accesses and substantially improve the overall system energy efficiency of DNN hardware accelerators.

Taking advantage of the reduced computation complexity, dedicated hardware accelerators [22, 23] for CIFAR-10 dataset [24] have been proposed with digital or mixed-signal neuron array, achieving ∼86% test accuracy with all weights stored on a chip. It should be also noted that ternary precision have demonstrated better performance to binary precision [25], especially for large-scale datasets. To that end, implementing deep neural networks with ternary activation precision and binary weight precision is of a particular interest.

The arithmetic complexity reduction from the binary and ternary algorithms, however, makes row-by-row memory access dominating the speed and energy efficiency of DNN hardware [7]. Conventional on-chip static random-access memory, SRAM, requires row-by-row accesses, and fetching a very large number of weights in this manner consumes substantial energy and delay.

The advent of the binary-weight DNNs and CNNs opens a new possibility for SRAM-based in-memory computing, since each weight in those algorithms can be nicely stored in a single SRAM bitcell. By turning on multiple or all rows simultaneously, the input/activation values are applied as wordline voltages, which in turn in-

teract with the bitcells to perform MAC computation, typically in an analog manner. This can eliminate explicit memory access, which otherwise pose energy/performance bottlenecks in DNN/CNN hardware implementations. A number of works recently have demonstrated this type of in-SRAM computing [26–34], and we summarized representative works in Table. 1.1.

In [26], in-SRAM computing hardware in 130nm CMOS was demonstrated. This design employs binary weights, each of which is stored in a 6T bitcell. The 5-bit inputs are converted to analog voltages via Digital-to-Analog Converters (DACs) embedded in the address decoder, which drives the WordLines (WL). Each WL voltage modulates the resistance of access transistors of bitcells of that row. Depending on the weight stored in each bitcell, the bitcell either discharges or charges the BitLines (BL), making BL voltage proportionally grow with the MAC computation results. The bitline voltage is finally digitized into a binary value by a single sense amplifier in the column circuitry.

Table 1.1: Comparison of Recent in-SRAM Computing Hardware Demonstrations.

	Biswas et al. [28] ISSCC 2018	Khwa et al. [30] ISSCC 2018	Valavi et al. [31] Symp. VLSI 2018	XNOR-SRAM [32] Symp. VLSI 2018
Technology	65nm	65nm	65nm	65nm
SRAM bitcell	10T	6T	10T1C	12T
SRAM array size	256×64	512×256	192×192	256×64
Supply voltage	0.9-1.2V	1V	0.68-1.2V	0.6-1V
Column/row sensing	Integrating ADC	SA	SA	Flash ADC
DNN model	CNN (A: 7b, W: 1b)	FC layer of CNN (A: 1b, W: 1b)	CNN (A: 1b, W: 1b)	CNN/MLP (A:ternary, W:1b)
Energy Efficiency (TOPS/W)	28.1	55.8	658	403
MNIST accuracy	96%	95.1%	98.6%	98.3%
CIFAR-10 accuracy	N/A	N/A	83.3%	85.7%

This work employs 6T SRAM circuits, promising a compact silicon footprint. However, it cannot support mainstream DNN and CNN algorithms. Even though binarized neural network algorithms [16, 35] binarize the activation at the output of each layer, still the partial sum and accumulation needs to be performed with high precision. Since the partial sum result at each SRAM output is binarized prematurely in [26], if the neural network layer cannot fit in one SRAM array, the final neural network accuracy can be considerably degraded. By combining many weak classifiers (shallow neural networks), a boosting classifier is demonstrated, but only achieved 90% accuracy for MNIST dataset.

In-SRAM computing hardware was also presented in [27]. This was not for accelerating neural network algorithms but for Content Addressable Memory (CAM) and bit-wise AND/OR/XOR operations of two rows of SRAM. The main focus of the work was on the novel 6T bitcell design featuring portless write using N-well and decoupled differential read, scaling the lowest functional voltage down to 0.3V in a 55nm CMOS technology.

Conv-RAM [28] integrates digital-to-analog converters (DACs) for analog inputs, binary weights stored in SRAM, and analog-to-digital converters (ADCs) to convert the in-memory computation results back to digital values. In-memory computation in [28] targets convolution operation, which is accomplished by row-wise charge sharing of the SRAM bitcells in the same row. To perform this, local analog multiply-and-average circuits are added every 16 rows (out of the 256-row custom SRAM array). However, within a block of 16 rows, the in-memory SRAM still goes through row-by-row operation, and the integrating ADC exhibits slow speed. MNIST accuracy of 96% was reported, but only 100 test images was used for the accuracy calculation.

In-memory computing with on-chip training capability was presented in [29], where the weights were fine-tuned based on on-chip variability. The work reported an

accuracy of 96% on the MIT-CBCL dataset for relatively simple face detection tasks. In-memory computation in this work is limited to reading out multi-bit weights that are stored in different rows to a single analog voltage, instead of performing a MAC or convolution operation inside the memory.

In [30], the authors demonstrated MAC operations using a 4kb SRAM for fully-connected neural networks in edge processors. The paper proposed techniques to mitigate the challenges of excessive current, sense-amplifier offset, and sensing reference voltage optimization, arising due to simultaneous activation of multiple word lines in the in-memory computing scheme. However, similar to [26], each column output is binarized with a single sense amplifier, which limits the accuracy and scalability to arbitrary large DNNs. In addition, only fully-connected layers of DNNs are mapped onto in-SRAM computing, and the measured MNIST accuracy was limited to 95.1%.

Neural Cache [34] re-purposes the large last-level caches in microprocessors towards CNN acceleration with in-memory computing. However, the in-memory computing scheme only turns on *two* rows of SRAM in one cycle, which limits the parallelism.

PROMISE [33] is a programmable mixed-signal accelerator that supports diverse machine learning algorithms with a custom Instruction Set Architecture (ISA) and compiler support. However, they only demonstrated machine learning tasks on relatively simple benchmarks including MNIST and MIT-CBCL datasets.

A binarized CNN accelerator was presented in [31], which also performs a modified batch normalization with analog computation, and reported 83.27% test accuracy for CIFAR-10 dataset. However, since each column end is binarized with a single sense amplifier, it cannot naturally support ensuing high-precision operations such as max-pooling, and also lacks scalability for larger CNNs.

Twin-8T [36] employed two of the conventional 8T SRAM structures [37]. Supporting multi-bit CNNs, it achieves the accuracy of up to 90.42% for CIFAR-10. This design can simultaneously turn on 9/18 rows in single-/dual-channel mode, achieving energy efficiency of 37.5/72.1 TOPS/W. Compute SRAM [38] employs transposable SRAM [39] and implements bit-serial digital operations near the peripherals. The proposed 'digital' computing scheme avoids less robust analog computation, but allows to turn on only two rows of bitcells in one clock cycle, resulting in limited throughput and energy-efficiency.

To reduce the delay and energy associated with on-chip SRAM accesses, recent works have proposed SRAM-based in-memory computing (IMC) scheme, which performs computation on the bitline without reading out each row of bitcells [29–32, 36, 40, 41], demonstrating large improvement in energy efficiency and throughput. CONV-SRAM [41] integrates digital-to-analog converters (DACs) for analog wordlines, binary weights stored in SRAM, and analog-to-digital converters (ADCs) to convert the in-memory computation results back to digital values.

While prior in-SRAM computing works have made different design decisions, we focus on robust and scalable in-memory computing with the goal to further advance the trade-off of the DNN accuracy and energy-efficiency. Unlike some of the prior works [30, 31] that connect the drain/source of additional transistors directly to the SRAM storage nodes, we only connect the gate of additional transistors. This is critical to eliminate any write disturb when all rows are asserted. Also, unlike [30, 31] that prematurely binarize the analog bitline voltage with a single sense amplifier, we employ a multi-bit ADC. This enables scalability to arbitrary-sized DNNs.

In addition to DNN inference acceleration, on-device DNN training hardware acceleration has also attracted research interests from both academia and industry. With local training capability on mobile or edge devices, people can train the

DNN models with users own sensitive data. Unlike inference-only tasks, the back-propagation (BP) based training algorithm involves weight transpose for convolution and fully-connected (FC) layers. Maintaining two copies of weights (original and transposed) or employing transposable SRAM [42] incurs large area overhead or custom SRAM bitcell design.

The rest of this dissertation is organized as follows: Chapter 2 presents a low-power processor for real-time ECG-based biometric authentication and cardiac monitoring. This chapter explains the ECG processing data flow for real-time biometric authentication, arrhythmia and anomaly detection. In particular, lower power optimizations are proposed to reduce the ECG processor power including employing selective precisions in the hardware design, hardware optimization for Finite Impulse Response (FIR) filter, normalization module and inverse of square root module, network pruning based on Lasso regression. Measurement results are presented for a 65nm prototype chip of the proposed ECG processor.

Chapter 3 proposes a resistive in-memory computing SRAM macro named "XNOR-SRAM". The bitcell macro design of XNOR-SRAM is first presented. The resisitve voltage divider model is presented to explain the analog averaging operation for both binary and ternary activations. Design choices on the number of ADC levels, confined linear vs. nonlinear quantization scheme are also explained. The XNOR-SRAM macro is implemented in 65nm CMOS technology. Measurement results of a number of prototype chips are presented and analyzed in details. In addition, an ensemble-network based technique is proposed to improve the IMC classification accuracy. Comparison to a digital baseline and prior works is made to show the advantage of XNOR-SRAM macro in energy efficiency.

In Chapter 4, a capacitive in-memory computing SRAM macro named "C3SRAM" is presented. C3SRAM macro architecture is first presented followed by the bit cell

design. Double-samplng based self-calibrating single-ended comparators are used for the ADC operation. Signal switching order is explained to ensure correct functionality and highest throughput. Next, multi-bit activation precision support in C3SRAM is explained. The C3SRAM macro is implemented in 65nm CMOS. Finally, measurements on power/energy, the DNN accuracy and ADC varaiability are presented and analyzed.

Chapter 5 describes an in-memory computing architecture titled "Vesti", which is built on top of XNOR-SRAM macros. This chapter first presents a more detailed review of prior in-SRAM computing macros. Next, practical challenges of IMC accelerators are analyzed, based on which the Vesti accelerator design is proposed. Double buffering technique is explained for the Vesti architecture to hide the weight loading latency. A special activation memory storage and access pattern is presented for 3×3 convolution. Experiment results are finally presented and analyzed for different activation precision and DNN models.

Chapter 6 proposes a programmable in-memory computing architecture named "PIMCA" which consists of 108 C3SRAM macros. First, the PIMCA overall architecture and instruction set is explained. Next, storage and access pattern of the activation memory is explained. PE organization and mapping are explained for different layer types and weight precision, followed by the 256-way SIMD processor design explanation. Measurement results of a 65nm prototype chip on ADC variability, power/energy and DNN workload accuracies are finally presented.

Chapter 7 presents a fixed-point CNN learning processor. The chapter first reviews the commonly used stochastic gradient descent (SGD) based learning algorithm. Next, the CNN learning processor design and optimization are presented. In particular, the layer-layer instruction set, input feeder with two-level FIFO array, and dual-read-mode weight storage/access scheme are described. Finally, measurement

results of a 65nm prototype chip are presented.

Finally, chapter 8 concludes the dissertation.

Chapter 2

A SMART ECG PROCESSOR FOR REAL-TIME BIOMETRIC AUTHENTICATION AND PERSONAL CARDIAC MONITORING

2.1 Introduction

Wearable devices are becoming ubiquitous in our daily lives, many of which featuring ability to sense and process our physiological signals including photoplethysmogram (PPG), electrocardiogram (ECG), bio-impedance, etc. Smart watches such as Apple Watch Series 3 [43] and Samsung Gear S3 [44] can measure our heart rate by analyzing PPG signals. However, to obtain further cardiac health information beyond heart rate, ECG signals can be analyzed. By inspecting the rhythm/shape of ECG beats, initial assessment of cardiovascular diseases can be performed. Due to privacy concerns, instead of processing ECG signals on cloud servers, performing ECG-based cardiac monitoring on local wearable devices has gained a lot of attention.

Several custom ECG processors [45–47] have demonstrated arrhythmia detection by continuously monitoring heart rate variability or frequency spectrum of ECG signals. In addition, detection of abnormal ECG pulse shapes has been presented in [48]. In the commercial market, several wearable ECG recording devices are available for ambulatory monitoring such as Zio patch [49], KardiaBand [50], and Simband [51].

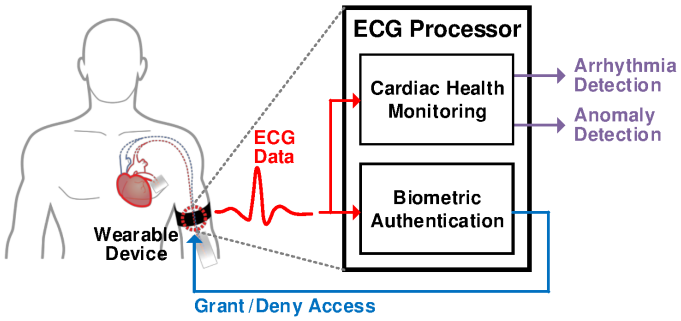
On the other hand, privacy concerns on personal health data necessitate enhanced security to access such wearable devices. Authentication based on biometrics such as fingerprint [52], iris [53], gesture [54], etc., is becoming increasingly popular for wearables. ECG signal has emerged as an attractive biometric modality, due to two key advantages compared to other existing methods: (1) ECG originates from the

electrical activity of the heart, thus providing intrinsic liveness proof, and (2) ECG authentication is difficult to spoof, since the ECG signal cannot be easily replicated. In recent years, many prior works proposed ECG-based authentication methods using various machine learning algorithms [55–59].

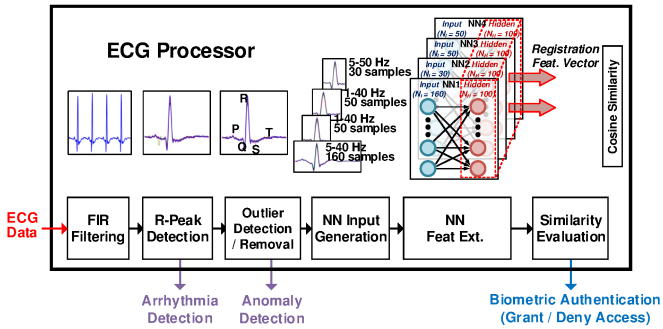
A few ECG-based authentication works have been reported in hardware. The authors of [57] implemented an ECG authentication deep neural network on FPGA, consuming ∼1 MB memory and 256 mW power. In [58], a cross-correlation based ECG authentication algorithm was implemented on the ARM microcontroller unit (MCU) in a wearable watch. However, both works evaluated authentication only on relatively small databases (90 subjects for [57] and 28 subjects for [58]). Furthermore, considering severe power constraints of wearable devices, compared to FPGAs/MCUs, energy-efficient ASIC processors are desired. Since both ECG authentication and cardiac monitoring share certain computation modules, it is natural to integrate both functionalities in a single chip.

In this chapter, we present an ultra-low-power smart ECG processor that performs both ECG-based authentication and cardiac monitoring including arrhythmia detection and anomaly detection [60]. Fig. 2.1(a) shows the high-level illustration of the proposed ECG processor. The ECG processor was designed for watch-type wearable devices such as smart watches or wristbands. We evaluated the ECG-based authentication performance on three ECG databases: MIT-BIH NSRDB [61] (18 subjects) and ECG-ID [62] (90 subjects) and an in-house ECG database (645 subjects). The arrhythmia detection performance was evaluated on MIT-BIH arrhythmia database [63], targeting atrial premature contraction (APC) or premature ventricular contraction (PVC). Our main contributions include:

1. This chapter presents, to the best of our knowledge, the first ECG processor that integrates both ECG-based authentication and cardiac monitoring func-



(a)



(b)

Figure 2.1: (a) The proposed ECG processor integrates two functionalities: cardiac health monitoring and biometric authentication. (b) Computation flow of the ECG processor.

tionalities.

2. A novel neural network based ECG feature extraction method is implemented and optimized for on-chip weight memory storage via data-driven Lasso regression. Aided by this, the ECG processor only consumes 1.06 μW at 0.55 V for real-time ECG authentication.

3. ECG authentication accuracy is verified on a large 645-subject database considering temporal variability of ECG signals, and achieves a low equal error rate (EER) of <2.5%, demonstrating its practical feasibility.

4. For cardiac monitoring, we achieve 93.13% arrhythmia detection sensitivity with a specificity of 89.78% for 42 subjects in the MIT-BIH arrhythmia database.

2.2 Smart ECG Processor Operations

The input to the ECG processor is single-lead raw digitized ECG signal sampled at 250 Hz with 13-bit precision, which goes through FIR filtering, R-peak detection, outlier detection/removal, normalization, neural network based feature extraction and cosine similarity evaluation. Fig. 2.1(b) illustrates the computation flow of these modules. The proposed ECG processor supports two ECG functionalities with shared hardware: (1) authentication and (2) cardiac health monitoring.

In the authentication mode, the ECG processor works in two stages: registration and identification. In the registration stage, 30 continuous ECG beats of the user are acquired, among which the outliers are detected and removed. The feature vectors (FVs) of all remaining valid ECG beats are extracted by neural networks (NNs), and the mean FV is registered. In the identification stage, the ECG processor acquires 4 valid (outlier-free) ECG beats, and the FVs of the 4 ECG beats are extracted using the same NNs. If the mean of the 4 newly extracted FVs is sufficiently similar to the registered mean FV, then the user is accepted; otherwise, the user is rejected. Acquiring 4 valid ECG beats for identification achieves a good balance between the identification time and authentication accuracy, where EER is 1.1% in software simulation using NNs with floating-point precision and without Lasso compression. If we use only 3 and 2 valid beats for authentication, we observed that the EER noticeably degrades from 1.1% to 2.0% and 3.9%, respectively.

In the cardiac monitoring mode, R-peak detection and outlier detection modules are used to detect arrhythmia (irregular ECG rhythm) and anomaly (abnormal ECG shape), respectively. Fig. 2.3 shows the overall architecture and operations, which

are described further in the following subsections. We explored the design space with different choices of the FIR filtering frequency bands, data segmentation, network topology, activation function, similarity metric, etc., using a forward greedy and backward greedy algorithm [64]. In the forward greedy algorithm, we evaluated each candidate model (a neural network and its preprocessing) on the first half of ECG records of the in-house 645-subject ECG databases, and incrementally augment the pool of models with a model that achieves the highest accuracy when combined with other ones in the pool. In the backward greedy algorithm, we incrementally remove a model such that the remained models in the pool achieve the best accuracy. Four final models remained in the pool were selected as the final choice.

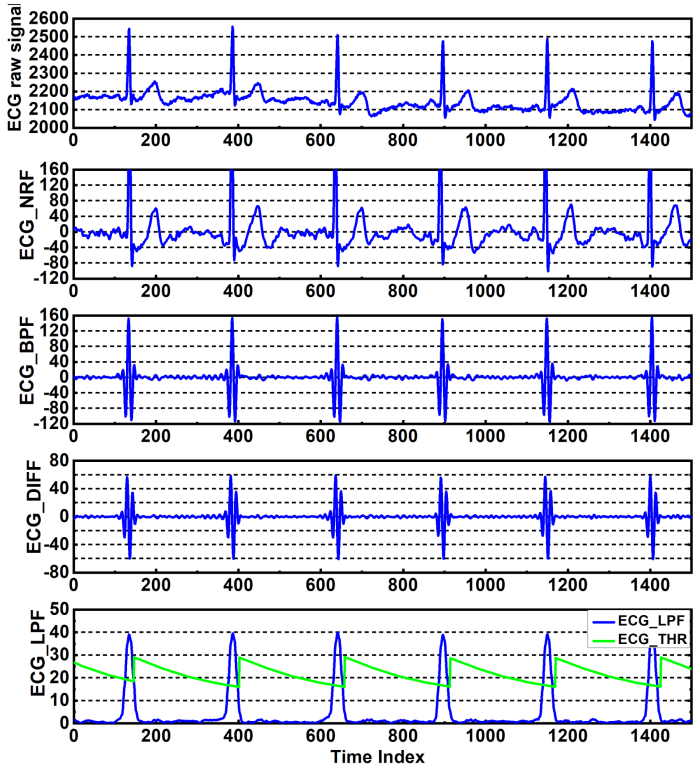


Figure 2.2: Representative ECG waveforms. From top to bottom: ECG raw signal, 256-tap FIR NRF output, 42-tap FIR BPF output, differentiator output, and 11-tap FIR LPF output (green line: dynamic threshold).

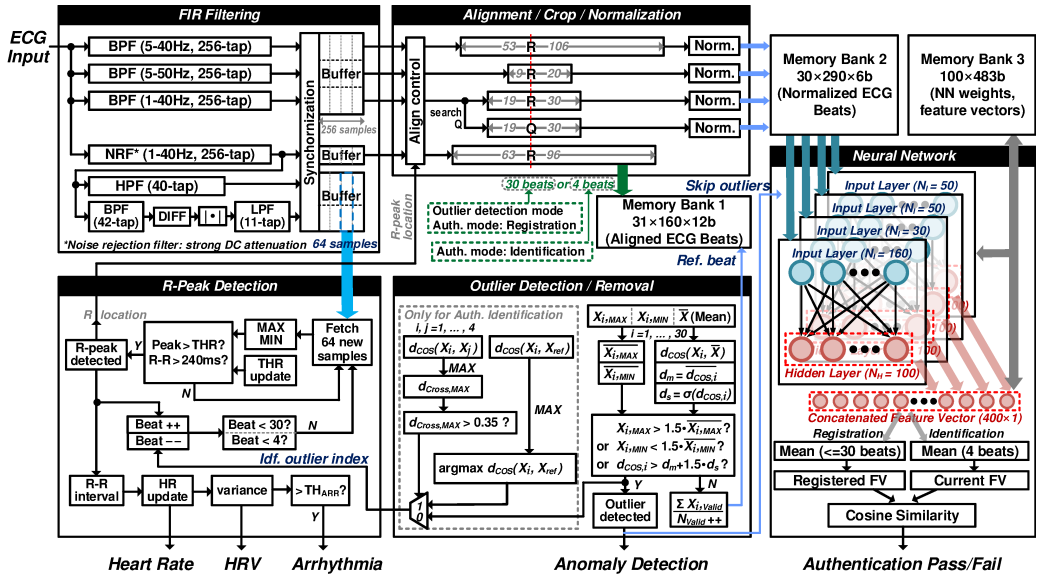


Figure 2.3: The overall architecture, dataflow, and computations of the proposed ECG processor.

2.2.1 FIR Filtering

A 256-tap FIR noise rejection filter (NRF) with cutoff frequency of 1-40 Hz is designed to reject both high frequency noise as well as DC wandering of the raw ECG signal. The NRF output ("ECG_NRF") is further filtered by a 40-tap FIR high pass filter (HPF), as well as a cascade of 42-tap FIR band pass filter ("ECG_BPF"), differentiator ("ECG_DIFF") and a 11-tap FIR low pass filter ("ECG_LPF"). Simulated waveforms of these filter outputs are shown in Fig. 2.2. In parallel, we have three additional 256-tap FIR BPFs with cutoff frequencies of 5-40 Hz, 1-40 Hz, and 5-50 Hz. These filters extract ECG information in different frequency ranges, which will be later used for 4 separate NNs. Let us denote these three filters as BPF_5_40, BPF_1_40, and BPF_5_50, respectively.

2.2.2 R-peak Detection

The outputs of LPF, HPF and four 256-tap FIR BPFs will be buffered in different but time-aligned consecutive 64-sample windows. The HPF output is used to accurately determine the maximum/minimum peaks in a 64-sample window. The LPF output is compared with a dynamic threshold ("ECG_THR") to detect the R-peak of ECG beats within a window [65], as shown in Fig. 2.2 (bottom). When a valid R-peak is detected, we extract a 160-sample segment from the buffer for the outputs of NRF and BPF_5_40 aligned at R-peak, two 50-sample segments from the buffer for the output of BPF_1_40 (one aligned at R-peak, the other aligned at Q-point), and a 30-sample segment from the buffer for the output of BPF_5_50 aligned at R-peak. The aligned 30 or 4 ECG beats after NRF are stored in on-chip memory for outlier detection.

2.2.3 Arrhythmia Detection

Among the various types of arrhythmia that concern cardiac health, we focus on the detection of sudden change in the R-R interval. This can be caused by premature contractions such as atrial premature contraction (APC) and premature ventricular contraction (PVC), for which example ECG waveforms from the MIT-BIH arrhythmia database [63] are shown in Fig. 2.4. We first estimate the instantaneous heart rate as the inverse of R-R interval. Then, the instantaneous heart rate variability (HRV) is estimated by computing the standard deviation of the past three consecutive heart rates. If the instantaneous HRV is above a threshold, we define that an arrhythmia is detected. If needed, the stored ECG beats can be exported for further diagnosis by cardiologists. Our arrhythmia detection algorithm is similar to that in [47], where HRV in a non-overlapping 10-second time window is estimated. In our ECG proces-

sor, HRV in a overlapping window of four ECG beats is estimated, which can more precisely locate the ECG beat when arrhythmia occurs.

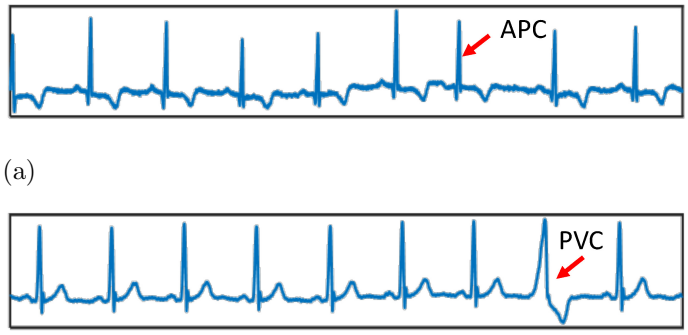


Figure 2.4: ECG waveforms of (a) atrial premature contraction (APC) and (b) premature ventricular contraction (PVC) are shown, from record 100 and 116 of [63], respectively.

2.2.4 Outlier Detection/Removal

Following the R-peak and arrhythmia detection module, the outlier detection module is present. It is notable that we employ the same outlier detection hardware for both authentication and cardiac monitoring modes, but in an opposite manner. For authentication, the objective is to detect the outliers and remove them, to form a representative (outlier-free) set of ECG beats for personal ECG feature extraction; for cardiac monitoring, we actually detect the outliers and save them, since they are considered as potential anomalies for the individual's cardiac health. Further details of the outlier detection/removal module for both operation modes are described below.

Authentication mode: In the authentication mode, after a certain number of ECG beats are obtained, we find outliers among the collected ECG beats and discard them, in order to acquire an outlier-free, representative ECG beat set for each

individual. The outlier removal algorithm is adopted from the techniques proposed in [66]. An ECG beat is defined as an outlier when any of the following three conditions is satisfied:

1. $X_{max} > 1.5\widetilde{X}_{max}$,
2. $X_{min} < 1.5\widetilde{X}_{min}$,
3. $d_{cos} > \bar{d}_{cos} + \gamma \cdot \sigma$,

where X_{max}/X_{min} are the maximum/minimum value of the ECG beat, respectively, and $\widetilde{X}_{max}/\widetilde{X}_{min}$ are the median of maximum/minimum values of all acquired ECG beats. d_{cos} is the cosine distance of the ECG beat to the mean ECG beat, $\bar{d}_{cos}$ is the mean of the cosine distances of all ECG beats to the mean ECG beat, and σ is the standard deviation of the cosine distances. γ is a configurable parameter in our hardware design, whose value can be 0.25, 0.5, 1 or 2. Smaller γ represents more stringent outlier removal. Depending on the scale and noise level of the target ECG data, different γ values might be desired. In our experiments, 0.5 is found to be the optimal γ value for all three ECG databases we investigated. The cosine distance between two ECG beats is defined as:

$$d_{cos}(\mathbf{X}_1, \mathbf{X}_2) = 1 - \frac{\mathbf{X}_1^T \mathbf{X}_2}{\|\mathbf{X}_1\|_2 \|\mathbf{X}_2\|_2}, \tag{2.1}$$

where $\mathbf{X}_1$ and $\mathbf{X}_2$ are two 160-sample ECG beats.

To ensure that sufficient ECG beats are used in the identification stage, in case an outlier is detected and discarded, new ECG beats are continuously read in until four ECG beats are collected. With outlier removal, EER improves from 2.60% to 1.70% for the in-house 645-subject ECG database.

Cardiac monitoring mode: In the cardiac monitoring mode, in addition to detecting abnormal heart rate, we can also detect abnormal ECG pulse shape. As

aforementioned, outliers among acquired ECG beats will be detected and removed in the authentication mode for better feature extraction quality. By reusing this identical outlier detection module, outliers in every 30 ECG beats will be continuously detected and reported in the cardiac monitoring mode. The ECG beats detected with abnormal ECG pulse shapes might be classified as an anomaly for cardiac health, as shown in Fig. 2.4(b). As needed, all 30 ECG beats can be exported for further analysis by cardiologists. Similar to the authentication mode, cosine distance threshold parameter γ can be varied to adjust the criteria for anomaly detection in the cardiac monitoring mode.

2.2.5 Normalization

Before the four ECG streams are sent to four corresponding NNs, we normalize the data such that NN input values are bounded within a certain range. Normalization is performed in two steps. First, the aligned ECG beats are normalized to zero-mean and unit-variance across different beats. Second, we normalize the ECG beats by the mean and standard deviation of ECG beats used for NN training. Then, ECG beats become normalized in both inter-beat and intra-beat dimensions.

2.2.6 Neural Network Based Feature Extraction

As shown in Fig. 2.5, four parallel NNs with input layer, one hidden layer, and output layer are designed to extract features from different frequencies and alignment (e.g. aligned at Q versus R). For each NN, there are 100 hidden layer neurons and 1,146 output layer neurons. The number of input neurons varies from 30 to 160, depending on the number of samples. The activation function of the hidden layer is $tanh(x)$. The NNs were trained on the first half of ECG records of the in-house 645-subject ECG database with 10-fold cross-validation. Each subject has a different

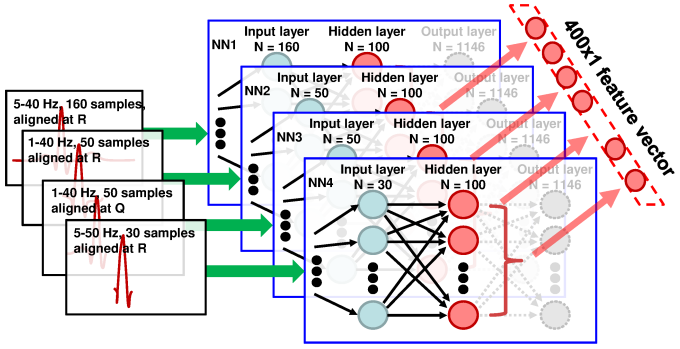


Figure 2.5: Four parallel neural networks for ECG feature extraction. Four 100×1 FVs are concatenated to form a 400×1 FV.

number of ECG records acquired over different time, and the total number of ECG records is 1,146. We could map different ECG records from the same subject to the same output neuron or to different output neurons. We tried both mapping schemes and found that the latter one leads to better EER, as it could extract ECG features in the hidden layer more effectively when we spread out each ECG record's data to different neurons. Therefore, we employed 1,146 output neurons in the NNs.

Training: We first pre-train a two-layer deep belief network as the initial weights values of NN [67]. Then, we use the identity labels of samples as the supervision information for fine tuning. After training is done, the intention is to use the hidden layer output as the feature descriptor.

As illustrated in Fig. 2.6, two loss functions are collectively employed to improve the accuracy: identification loss function and verification loss function [64]. The identification loss function maximizes the difference of ECG features from different users. The cross entropy value after a softmax layer is evaluated for the identification loss. On the other hand, the verification loss function minimizes the ECG feature variation from the same user, considering the temporal variation in ECG signals from the same user. Pairs of ECG data $\boldsymbol{x}_i, \boldsymbol{x}_j$ are fed to the NNs. If they are from the same subject,

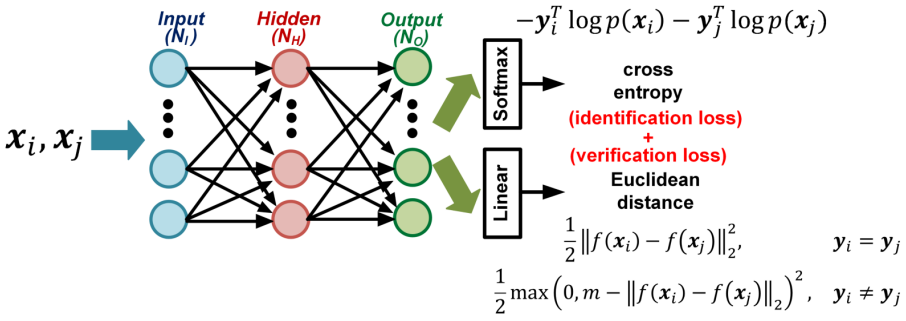


Figure 2.6: Identification and verification loss function used for training the neural networks.

the Euclidean distance between their network outputs is minimized; otherwise, the Euclidean distance is maximized to be greater than a specified threshold m.

Classification: Classification is performed with feedforward propagation using the trained NNs. Four 100×1 FVs are extracted from the hidden layer of four NNs and concatenated to form a 400×1 FV. The average 400×1 FV over all valid beats is considered as the final FV. Since the hidden layer is directly used for FVs, weights between hidden and output layers are not required for classification (only used in training).

2.2.7 Cosine Similarity Evaluation

In identification stage, cosine similarity is computed between the current (identification) FV and the registered FV:

$$cos_{sim} = \frac{FV_{new}^T FV_{reg}}{\|FV_{new}\|_2 \|FV_{reg}\|_2}. \tag{2.2}$$

When cos_{sim} is above the pre-defined threshold, the user will be accepted; otherwise, the user will be rejected.

2.3 Low-power Design Optimization

Power consumption is critical for our ECG processor due to the limited power budget of wearable devices. The ECG processor employs extensive clock gating based on module-level activity, and is optimized from software and hardware perspectives to substantially reduce the power consumption.

2.3.1 Selective Precision in Hardware Design

To reduce the power and area of the ECG processor, we adopted fixed-point precision representation. The input raw digitized ECG exhibits 13-bit precision. However, there is no need in keeping this precision throughout the entire signal processing flow. To optimally reduce the power without hampering accuracy, we selectively reduced the precision for different modules to the lowest before accuracy degradation occurs. The final precision values optimized for FIR filter coefficients, FIR filter signals, R-peak detection, normalization, NN based feature extraction, and similarity evaluation modules are 8-bit, 13-bit, 13-bit, 11-bit, 12-bit, 6-bit and 9-bit, respectively. Note that the 32-bit floating-point NN weights are quantized to only 6-bit, reducing the weight storage by 5.3×.

2.3.2 FIR Filter Design

We adopted the systolic architecture [68] for our direct form FIR filters design to improve the throughput. Since our FIR filters are linear-phase, the coefficients are always symmetrical around the center coefficient. Exploiting this, we used pre-adder to sum the samples associated with two symmetric coefficients, halving the number of necessary multipliers. In addition, we reduced the precision of the FIR coefficients to 8-bit. Consequently, a portion of the heading and ending coefficients were quan-

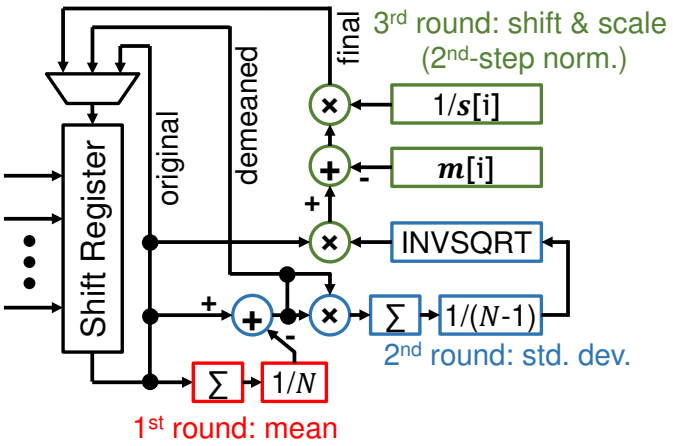


Figure 2.7: Normalization module based on three rotation rounds.

tized to zeros, which effectively reduced the orders of the FIR filters. Nonetheless, no significant impact was found on the filtering effect by discarding these coefficients. Therefore, we implemented reduced-order FIR filters to reduce area and power. For example, the 256-tap NRF, BPF_5_40, BPF_1_40 and BPF_5_50 were simplified to 148-tap, 178-tap, 178-tap and 178-tap, respectively. Delay lines were added to synchronize signals across different channels.

2.3.3 Normalization Module Design

As mentioned in Section II.E, the normalization of ECG beats consists of two steps. First, we compute the mean and standard deviation of an ECG beat. Then, we subtract the ECG beat by the mean and divide by the standard deviation. After this first step normalization, each ECG beat becomes zero-mean and unit-variance. Finally, we subtract the normalized ECG beat by a global mean vector and divide by a global standard deviation vector. We could have processed all samples in an ECG beat in parallel to reduce the normalization latency to a few clock cycles. However, since normalization is performed while reading in new ECG beats and the ECG beat

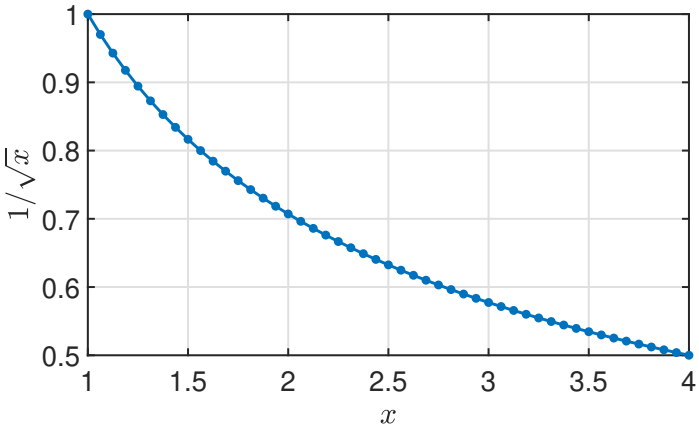


Figure 2.8: Piecewise linear approximation of $1/\sqrt{x}$ with 48 uniformly spaced segments on [1, 4).

rate is relatively low (60-100 bpm), the normalization module can tolerate higher latency (e.g. a few hundred clock cycles).

We designed a rotation-based normalization module (Fig. 2.7), which is time-multiplexed for three rounds of computation. The ECG beat is fed into a shift register from QRS alignment module. In the first round of rotation, the mean of the ECG beat is computed, and in the second round, the standard deviation of the ECG beat is computed. In the third round of rotation, the second-step normalization is completed. By reusing the data path for all samples in an ECG beat, only three multipliers are required for the normalization of one channel.

2.3.4 Inverse of Square Root Module Design

When we compute the cosine distance in outlier detection module, the standard deviations in normalization and arrhythmia detection module or the cosine similarity, we need to evaluate the inverse of square root (INVSQRT) function ($1/\sqrt{x}$), which is a nonlinear function. We approximated this nonlinear function with a piecewise linear function. In particular, we pre-computed the values (offset and slope) for 48 uniform-

length segments between [1, 4) as shown in Fig. 2.8. The input of the INVSQRT function has 14-bit precision. The six most significant bits (MSBs) of the input are used for segment index. The INVSQRT value for the input range beyond [1, 4) is obtained by bit-shifting. For example, if the input is 0.2, we first left-shift the input by 4 bits (i.e. $0.2 * 2^4 = 3.2$), to make the input fall within the valid range of [1, 4), then we need to left-shift the INVSQRT output by 2 (= 4/2) bits. In general, the amount of shifting at the output is half of the amount of shifting at the input, due to the fact that:

$$1/\sqrt{x} = 1/\sqrt{x \times 2^{2k}} \times 2^k. \tag{2.3}$$

2.3.5 Lasso Regression Based Sparsification of Neural Networks

As mentioned in Section II.F, the output layers of four parallel NNs are discarded as we extract the ECG features from the hidden layers, which reduces the number of weights by 16×. The input dimension of each NN is 160, 50, 50 and 30, respectively. The hidden layer dimension is 100 for all four NNs. The remaining fully-connected weight matrices have $(160 + 50 + 50 + 30) \times 100 = 29,000$ weights, which could still exceed on-chip storage capacity of wearable devices.

To reduce the power/area of NNs, we propose to sparsify weight matrices by Lasso regression. Denote the original trained dense NN weight matrix between the input layer (m neurons) and the hidden layer (n neurons) as $\mathbf{W_{ori}}$ ($m \times n$). Given a sufficient number (p) of representative input samples $\mathbf{X}$ ($p \times m$) to the NNs, the weighted sum for the hidden layer will be:

$$\mathbf{Y} = \mathbf{X} \times \mathbf{W_{ori}}. \tag{2.4}$$

We seek a sparse weight matrix $\mathbf{W}^*$ such that

$$\mathbf{X} \times \mathbf{W}^* \approx \mathbf{Y}. \tag{2.5}$$

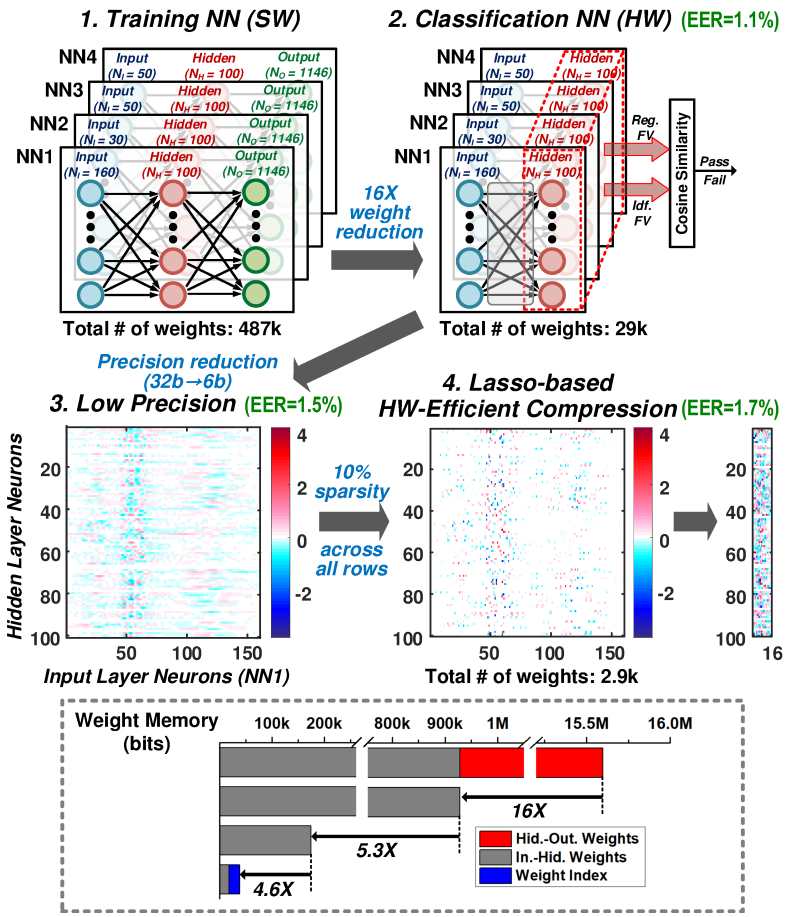


Figure 2.9: Summary of NN compression at different levels. (1) software model; (2) output layer removal; (3) quantization; (4) Lasso regression.

Lasso regression is a well-known algorithm [69] to find a sparse solution to (5). To sparsify the NN weights, we formulate n Lasso regression problems as follows:

$$\min_{\mathbf{w}_i^*} \|\mathbf{X} \times \mathbf{w}_i^* - \mathbf{X} \times \mathbf{w}_{ori,i}\|_2 + \lambda_i \|\mathbf{w}_i^*\|_1, i = 1, \ldots, n, \tag{2.6}$$

where $\mathbf{w}_i^*$ and $\mathbf{w}_{ori,i}$ are the i-th column of $\mathbf{W}^*$ and $\mathbf{W}_{ori}$, respectively, and λ_i is the regularization parameter in each Lasso regression problem. For the convenience of hardware implementation and efficient storage of sparse weight matrices, each $\mathbf{w}_i^*$ is preferred to have the same number of non-zero weights. Typically, the larger λ_i is, the sparser $\mathbf{w}_i^*$ will be.

For each Lasso regression problem, we conduct a binary search to find the smallest λ_i for each $\mathbf{w}_i^*$ column, so that the same target sparsity is achieved for all columns. For 10× weight compression, Lasso regression results in better EER of 1.70%, compared to the EER of 3.31% that is obtained when magnitude-based pruning is performed to reach the same sparsity. With 10× weight reduction, we achieved 4.6× memory compression (including overhead for weight indices) with only 0.2% EER degradation. Starting from the software NN model, Fig. 2.9 summarizes the overall NN weight reduction of 390×, collectively achieved by output layer removal, low-precision quantization, and Lasso-based compression.

The extracted features from four NNs are evaluated in parallel. The weighted sum of one hidden neuron is computed in one cycle. 160-to-16, 50-to-5, 50-to-5 and 30-3 multiplexers are used in the four NNs, to select corresponding partial inputs according to the sparse weight indices for each hidden neuron. In total, only 29 multipliers are required for the four NNs. After the bias is added, the weighted sum goes through a 128-entry look-up table based $tanh(x)$ module that only stores the positive parts, exploiting the rotational symmetry of $tanh(x)$.

2.4 Measurement Results

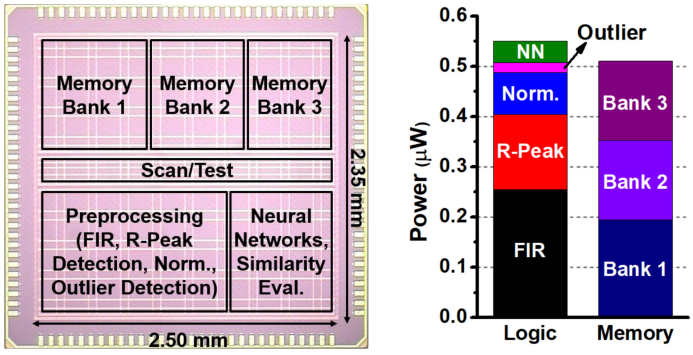


Figure 2.10: (a) Prototype chip micrograph and (b) power breakdown.

The prototype chip is implemented in 65nm LP CMOS (Fig. 2.10(a)). Total on-chip memory is 19.5 kB, out of which 4.6 kB is used for the NN weights. Measured power consumption of the ECG processor is 1.06 μW at 0.55 V supply and 2 kHz clock frequency, when it performs continuous real-time authentication. For cardiac monitoring, the ECG processor consumes 0.83 μW (at 1 kHz) for arrhythmia detection and 0.88 μW (at 3 kHz) for anomaly detection at 0.51 V supply. All measurements are performed at room temperature. Fig. 2.10(b) shows the power breakdown, based on module-level power percentages obtained from post-layout simulation.

2.4.1 Biometric Authentication

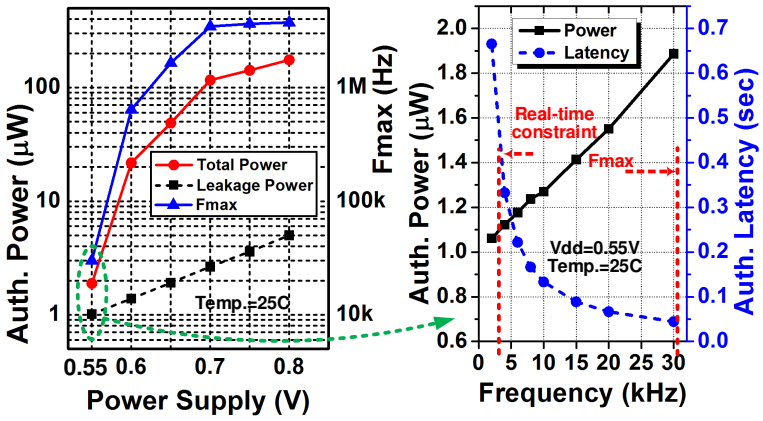


Figure 2.11: Measurement results of power, frequency and latency for authentication.

The maximum clock frequency scales down with lower supply voltages as shown in Fig. 2.11 (left). Fig. 2.11 (right) shows the tradeoff between authentication latency and power consumption at 0.55 V as we sweep the clock frequency. The minimum clock frequency for real-time authentication is 2 kHz. The combined latency of outlier detection, NN, and similarity evaluation modules dictates that the clock frequency

should not be too low; otherwise, user will need to wait for more than one second from the moment when the last ECG beat is acquired till the final authentication result comes out.

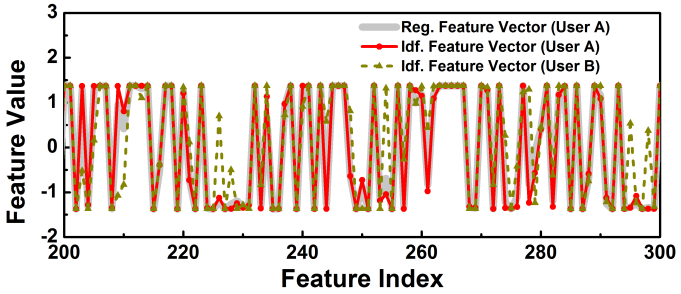


Figure 2.12: Measurement results of registration/identification FVs (features 201-300 are shown) from same and different users.

We tested the ECG processor with real-time ECG signals from a few volunteers. Fig. 2.12 shows the registration and identification FVs for one volunteer (user A), and the identification FV for another volunteer (user B). It can seen that there is a very small difference between the registration and identification FVs of user A, while there is sufficient difference between the registration FV of user A and identification FV of user B. As a result, only user A is authenticated during identification when user A's ECG data is registered.

We evaluated the authentication performance on three ECG databases. Note that clinical studies typically use 12-lead ECG signals to obtain spatial information of the heart's electrical activity. However, since we focus on wearable applications, which can practically integrate only single-lead ECG sensors, only single-lead or single-channel ECG signals are used for all three databases below. "Lead I" represents one of the 12-lead ECG orientations, namely right arm (-) to left arm (+).

- MIT-BIH normal sinus rhythm database (MIT-BIH NSRDB) [61, 70] includes 18 subjects (5 men, age 26-45, and 13 women, age 20-50) and ECG data of two

channels. We used the first channel in our testing.

- ECG-ID database [62] includes 90 subjects (44 men and 46 women, age 13-75) and data of one channel (lead I).

- In-house ECG database includes 645 subjects half men and half women. The ECG data has only one channel (lead I, electrodes on the wristband sense ECG

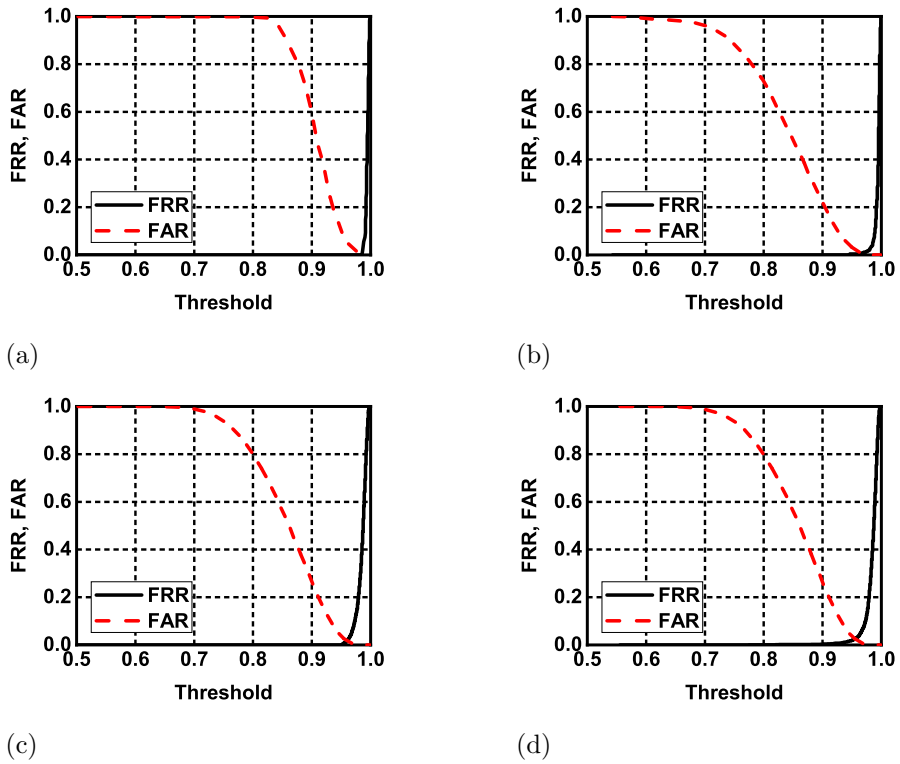


Figure 2.13: Measurement results of FAR and FRR of ECG authentication for three databases and different starting points are shown: (a) MIT-BIH NSRDB with 18 subjects (EER=0.10%), (b) ECG-ID database with 90 subjects (EER=0.74%), (c) in-house ECG database with 645 subjects with authentication starting at 4 seconds (results in best EER of 1.70%), and (d) in-house ECG database with 645 subjects with authentication starting at 1 second (results in worst EER of 2.48%).

signals from right index finger (-) to left wrist (+)), digitized by analog front end (AFE) chip ADS1292R [71] on a custom wristband. The subjects wearing the custom wristband sat calmly during ECG acquisition, and each ECG record is 2-minute long. The first half of the ECG records were used for NN training, and the second half of the ECG records were used for ECG authentication evaluation.

To evaluate the authentication accuracy, false acceptance rate (FAR), false rejection rate (FRR) and equal error rate (EER) are measured. FAR is the rate at which a wrong user is accepted; FRR is the rate at which a correct user is rejected; EER is the error rate when FAR and FRR are identical.

We perform registration once and identification three times for every subject. If a database has n subjects, we perform ECG authentication experiments for $n \times 3n$ times, where there are $3n$ same-user identification attempts and $3n \times (n-1)$ different-user identification attempts. By comparing $n \times 3n$ cosine similarity values with the threshold, we obtain FRR as the ratio of the number of the same-user cosine similarities that is less than the threshold, to the total number of the same-user cosine similarities ($3n$). FAR is obtained as the ratio of the number of the different-user cosine similarities that is greater than or equal to the threshold, to the total number of the different-user cosine similarities ($3n \times (n-1)$). By increasing the threshold, FAR decreases while FRR increases. Fig. 2.13 shows the FAR/FRR for a range of threshold values for MIT-BIH NSRDB, ECG-ID, and 645-subject in-house databases. The EER values are: 0.10% for MIT-BIH NSRDB, 0.74% for ECG-ID, and 1.70%/2.18%/2.48% (best/average/worst) for in-house ECG database. These measured EERs are comparable to EERs of recent fingerprint (0.8% [72]) and iris (0.82% [73]) based authentication algorithms.

Temporal variability exists in ECG signals for any individual, which poses a crit-

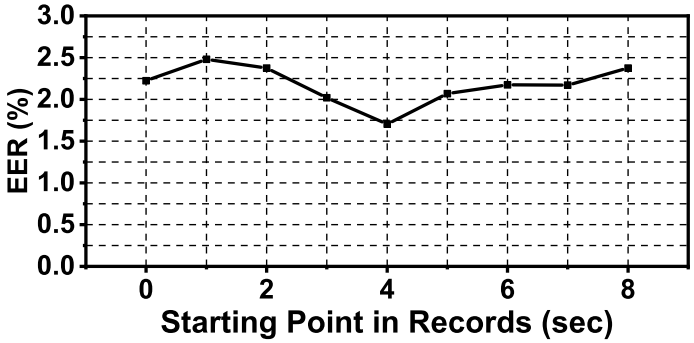


Figure 2.14: EER results on 645-subject in-house ECG database with varying starting points for registration and identification.

ical challenge for ECG authentication accuracy. To evaluate this, for the in-house ECG database, we varied the starting point of registration and identification among ∼8 seconds in the ECG records for 645 individuals, and measured EERs for these experiments. As shown in Fig. 2.14, the best EER is 1.70%, the worst EER is 2.48%, and the average EER for all 9 temporal experiments is 2.18%.

2.4.2 Arrhythmia Detection

We tested arrhythmia detection with ECG recordings from MIT-BIH arrhythmia database [63], which includes 48 half-hour excerpts of two-channel ambulatory ECG recordings from 25 men (age 32-89) and 22 women (age 23-89). We used the ECG data from channel 1. Fig. 2.15(a) shows successful arrhythmia detection for record 100 from [63]. When the PVC beat occurs, the HRV rises above the threshold (7.5 bpm) for a short period, which promptly turns on the detection signal.

In arrhythmia detection mode, the detection results are reported every heart beat based on the R-R intervals for the past 4 heart beats. Therefore, starting from the 4^{th} ECG beat, we can compare the arrhythmia detection results with annotations made by experts in the database to evaluate our arrhythmia detection accuracy. As mentioned in Section II.C, we focus on detection of APC and PVC, which are the

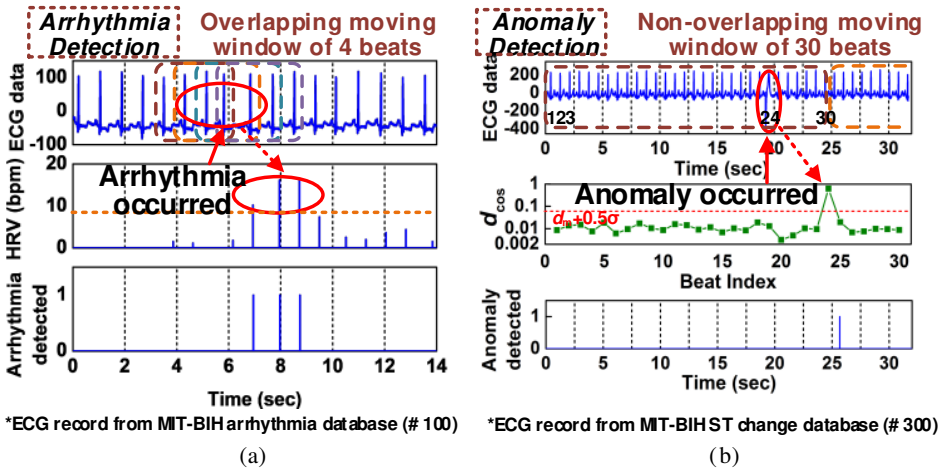


Figure 2.15: Measurement results of (a) arrhythmia detection and (b) anomaly detection.

two most common premature contractions [74, 75], and can be occasionally caused by heart diseases. The configurable HRV threshold can be a value between 0 and 15.9375 bpm.

Fig. 2.16 shows arrhythmia detection sensitivity (true positive rate) and specificity (true negative rate) measurement results for 48/42 subjects in the MIT-BIH arrhythmia database [63]. For all 48 subjects in [63], when the HRV threshold is 5 bpm, the sensitivity of APC or PVC detection is 80.65%, while the specificity is 88.98%. We noticed that 6 subjects (record 207, 208, 209, 213, 223, 232) contain hundreds of consecutive premature contraction beats that cannot be detected by our ECG processor, since HRV would be small in those cases. When these 6 subjects are excluded, the sensitivity for 42 subjects improve considerably, which shows that our ECG processor can well detect *sudden* HRV changes. At the HRV threshold of 6 bpm, the sensitivity is 93.13% and the specificity is 89.78%.

Table 2.1: Comparison with Prior ECG Processor Works.

	[46]	[47]	[48]	**This Work**	[57]	[58]
Technology	180nm	65nm	90nm	65nm	Artix-7 FPGA	Cortex-M MCU
Supply Voltage	0.7 V	0.4 V	0.5-1.0 V	0.55 V	1.0 V	Unknown
Include AFE	No	Yes	Yes	No	No	No
Power (Digital)	1.26 μW (Arr.)	45 nW (Arr.)	7-32.8 μW (M.I.)	1.06 μW (Auth.)	256 mW (Auth.)	Unknown
Clock Frequency	0.25-1 kHz	10 kHz	8-32 kHz	2 kHz	50 MHz	168 MHz
Memory Size	10.5 kB	3.7 kB	20.0 kB	19.5 kB	979.1 kB	Unknown
Arrhythmia Detection	Yes	Yes	Yes	Yes	No	No
Anomaly Detection	No	No	Yes	Yes	No	No
Authentication	No	No	No	Yes	Yes	Yes
Auth. EER/FAR/FRR	N/A	N/A	N/A	0.74% (ECG-ID) 1.70%-2.48% (645 subjects)	0.06% (ECG-ID)	FAR = 5.2% FRR = 1.9% (28 subjects)

Arr.: Arrhythmia, Auth.: Authentication, M.I.: myocardial infarction

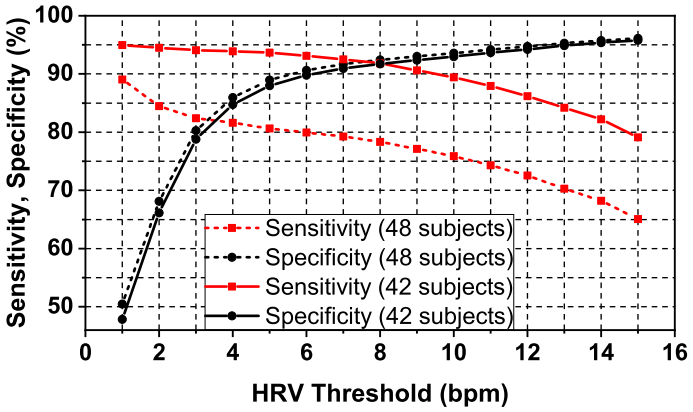


Figure 2.16: Measurement results of arrhythmia detection (APC or PVC) for subjects in MIT-BIH arrhythmia database [63].

2.4.3 Anomaly Detection

Anomaly detection is aimed at detecting abnormal ECG pulse shapes among normal ones, even if the heart rate is normal. We tested the ECG processor with ECG recordings with such abnormal pulse shapes. Fig. 2.15(b) shows the measurement results for an ECG recording (record 300) from MIT-BIH ST change database [76]. It can be seen that, although the heart rate is regular, the 24^{th} ECG beat appears to be very different in the ECG pulse shape, and its cosine distance (from simulation) to the mean ECG beat among the 30 beats is much higher than others and is higher than the threshold. Also, the minimum value of this 24^{th} is much lower than the 1.5 times the median minimum value. Therefore, this beat is detected as an outlier and reported by the ECG processor. The anomaly detection result will be updated every 30 ECG beats.

2.4.4 Comparison with Related Works

Table 2.1 provides the comparison of our proposed ECG processor against other related works. AFE is not included in our processor, so only the digital power of

prior works are reported in Table 2.1. While the cardiac monitoring part compares similarly to prior works, note that anomaly detection is much more complex than arrhythmia detection.

On the authentication side, our work is the first ASIC that implements ECG-based biometric authentication. The authors of [57] achieved a very low EER on the ECG-ID database. However, they trained different NNs for different users, while we used a fixed common NN for all users across different ECG databases. In addition, our ECG processor is tested on a much larger in-house database of 645 subjects, together with temporal variability in the ECG records. Only our ECG processor supports both cardiac monitoring and authentication in a single chip with $\sim$1 μW power for real-time operation.

2.5 Conclusion

We presented an ultra-low-power ECG processor that performs both biometric authentication and personal cardiac monitoring. Aided by output layer removal, low precision, and Lasso-based compression, a total of 390$\times$ NN memory is reduced compared to software. We achieved $<$2.5% EER for a 645-subject in-house ECG database, consuming 1.06 μW power for real-time ECG authentication. For arrhythmia detection, we achieved 93.13%/89.78% sensitivity/specificity for 42 subjects in MIT-BIH arrhythmia database. The proposed ECG processor enables secure access and cardiac monitoring in wearable devices with stringent power/area constraints.

Chapter 3

XNOR-SRAM: A RESISTIVE IMC SRAM MACRO

3.1 Introduction

Deep neural networks (DNNs) and convolutional neural networks (CNNs) have unprecedentedly improved the accuracies in large-scale recognition tasks [1–6]. However, the arithmetic complexity and memory access have limited the energy-efficiency and acceleration of DNN hardware [7–11].

To address this, in recent algorithms, weights and neuron activations are binarized to +1 or -1 [16, 17] such that the multiplication between an weight and an activation becomes an XNOR operation and the accumulation of the XNOR operations becomes bitcount of those XNOR results. Although the initial XNOR-Net [16] showed a relatively large test accuracy degradation (∼10-20%) for the ImageNet dataset, recent works that employ 2-bit precision [18] have shown 1–3% accuracy degradation for ImageNet, and this is an active research area in the machine learning community. Taking advantage of the reduced computation complexity, dedicated hardware accelerators [22, 23] for CIFAR-10 dataset [24] have been proposed with digital or mixed-signal neuron array, achieving ∼86% test accuracy with all weights stored on a chip. It should be also noted that ternary precision have demonstrated better performance to binary precision [25], especially for large-scale datasets. To that end, implementing deep neural networks with ternary activation precision and binary weight precision is of a particular interest.

The arithmetic complexity reduction from the binary and ternary algorithms, however, makes row-by-row memory access dominating the speed and energy efficiency

of DNN hardware [7]. Conventional on-chip static random-access memory, SRAM, requires row-by-row accesses, and fetching a very large number of weights in this manner consumes substantial energy and delay.

To reduce the delay and energy associated with on-chip SRAM accesses, recent works have proposed SRAM-based in-memory computing (IMC) scheme, which performs computation on the bitline without reading out each row of bitcells [29–32, 36, 40, 41], demonstrating large improvement in energy efficiency and throughput.

In this chapter, we present an in-memory mixed-signal SRAM macro titled "XNOR-SRAM" that not only energy-efficiently computes ternary-XNOR-and-accumulate (XAC) in binary/ternary DNNs, but also supports the DNNs/CNNs of arbitary size with high accuracy. Our XNOR-SRAM performs a 256-input XAC without explicit memory readout, via analog accumulation of bitwise ternary-XNOR results on the read bitline (RBL) voltage of the SRAM array, and digitizes the RBL voltage (V_{RBL}) using a flash analog-to-digital converter (ADC) embedded in the periphery. XNOR-SRAM supports binary weights (+1, -1) and binary inputs (+1, -1) as well as ternary inputs (+1, 0, -1).

Our 65-nm prototype chip achieves 300X better energy-delay product (EDP) than a digital baseline in computing XAC. DNN classification using our XNOR-SRAM achieves 98.3%/98.8% accuracy for MNIST and 87.3%/88.8% accuracy for CIFAR-10 using binary/ternary precision, respectively.

3.2 XNOR-SRAM Macro Design and Optimization

3.2.1 XNOR-SRAM Bitcell Design

Fig. 3.1 presents the proposed XNOR-SRAM architecture, which can map convolutional and fully-connected layers of CNNs and multi-layer perceptrons (MLPs).

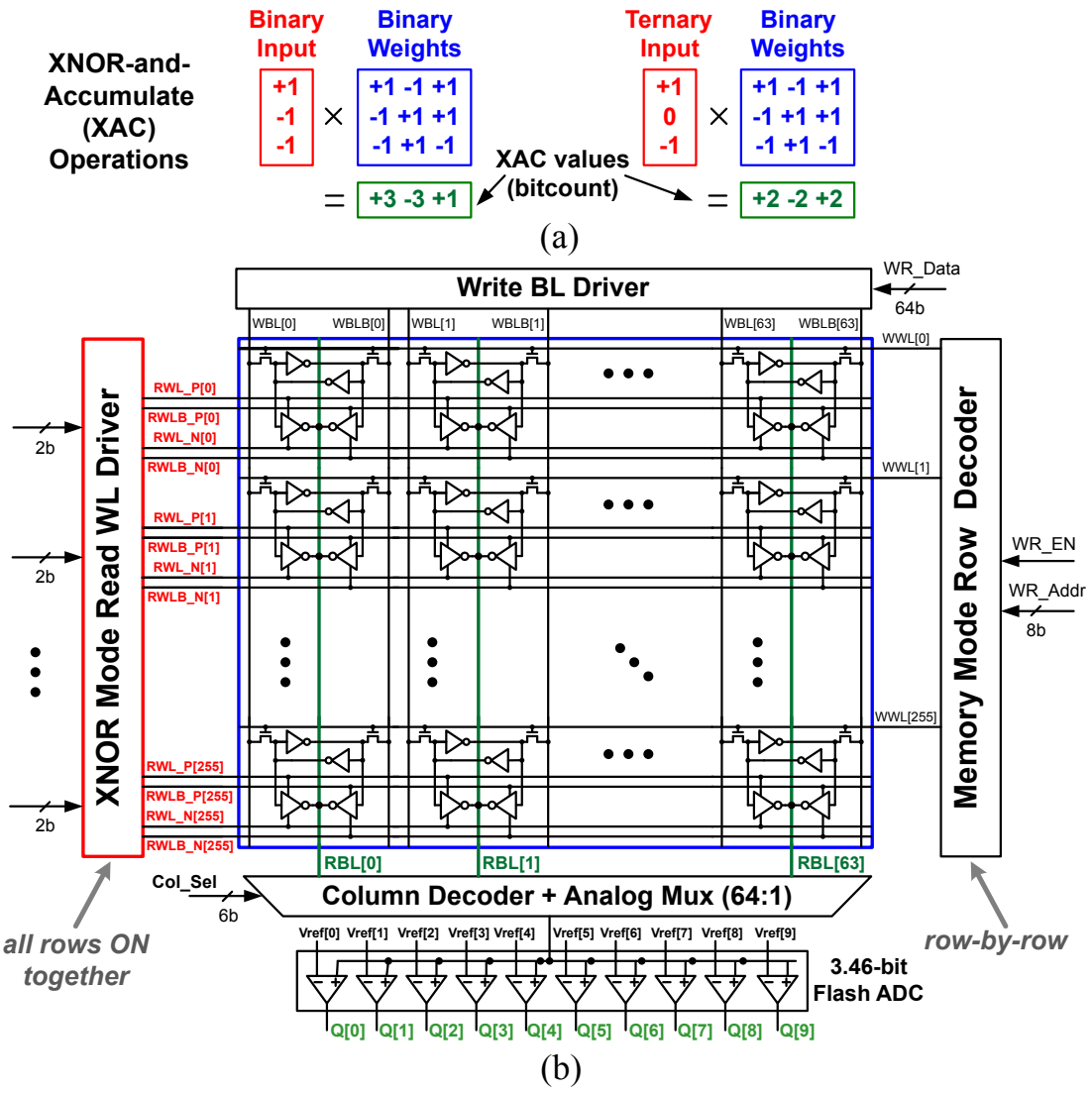
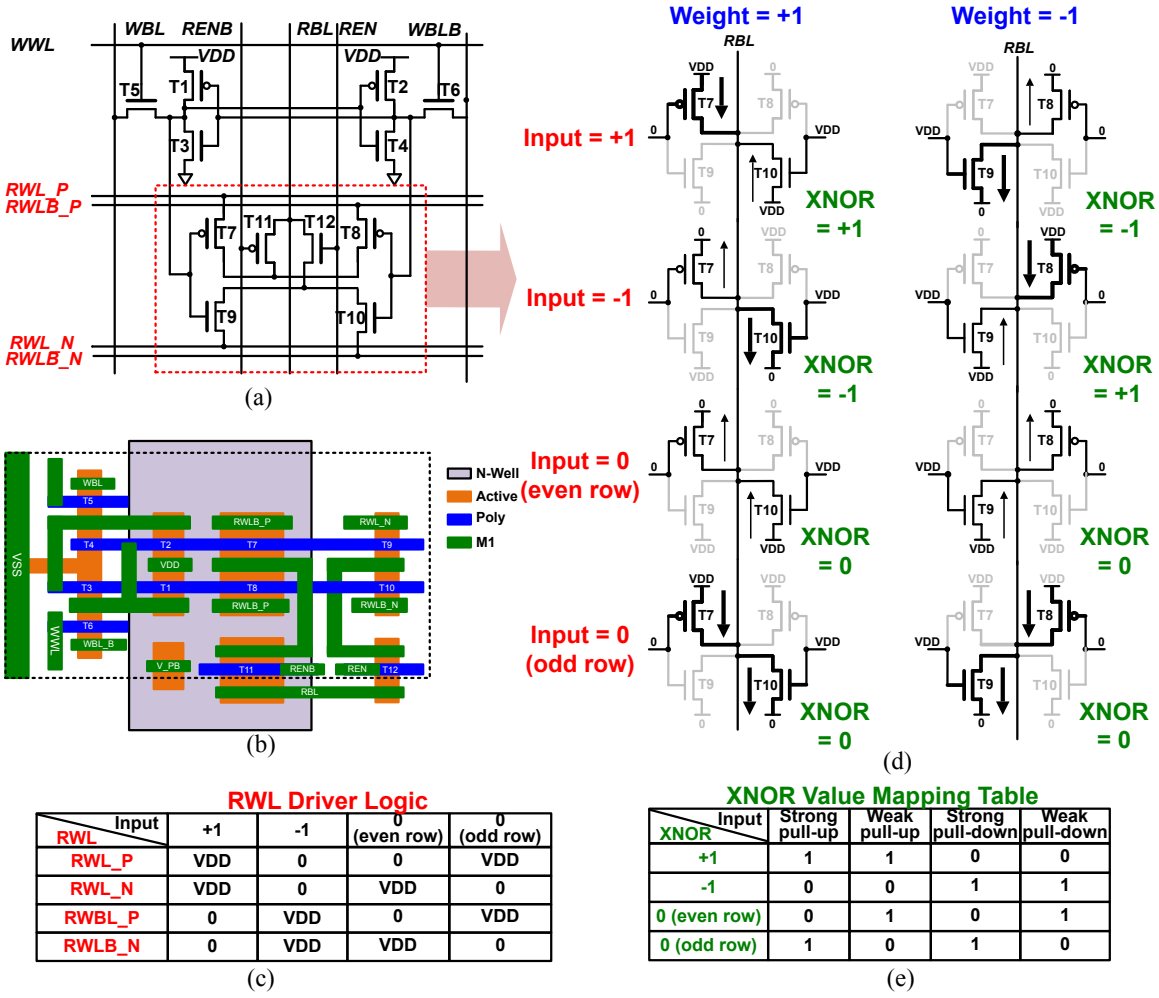


Figure 3.1: Overall architecture of XNOR-SRAM macro.

It consists of a 256-by-64 custom bitcell array, a row decoder, an XNOR-mode WL driver, and a column periphery including a 3.46-bit flash ADC. The XNOR-SRAM operates in either of two modes: memory mode and XNOR mode. In memory mode, it performs row-by-row digital read/write as regular SRAM. In XNOR mode, it performs in-memory XAC computation with all rows asserted simultaneously.

Fig. 3.2(a) shows the proposed 12T bitcell for XNOR-SRAM. T1 to T6 form a 6T cell; T7 to T10 form complimentary pull-up (PU) and pull-down (PD) circuits for



RWL Driver Logic

RWL \ Input	+1	-1	0 (even row)	0 (odd row)
RWL_P	VDD	0	0	VDD
RWL_N	VDD	0	VDD	0
RWBL_P	0	VDD	0	VDD
RWLB_N	0	VDD	VDD	0

(c)

XNOR Value Mapping Table

XNOR \ Input	Strong pull-up	Weak pull-up	Strong pull-down	Weak pull-down
+1	1	1	0	0
-1	0	0	1	1
0 (even row)	0	1	0	1
0 (odd row)	1	0	1	0

(e)

Figure 3.2: XNOR-SRAM bitcell design and XNOR-ACC operation with ternary inputs/activations and binary weights.

XNOR mode (and memory mode read); T11 and T12 power-gate the PU/PD circuits when the corresponding column is disabled. Fig. 3.2(b) shows the layout of the bitcell drawn in logic ground rules. The area is 3.915 μm^2 (2.7-by-1.45 μm). Except T7, T8, and T11, all transistors in the bitcell use the minimum size. We slighltly sized up the PMOS transistors T7, T8, and T11 to match its strength to their NMOS counterparts.

In XNOR mode, the read wordline (RWL) driver translates each ternary/binary

input activation to four RWLs according to Fig. 3.2(c). In the second half of a clock cycle, T11 and T12 in a selected column are turned on, and T7 to T10 perform ternary-XNOR operation between RWLs (activations of +1, 0, or -1) and the binary weight (+1 or -1) stored in the bitcell. The RBL voltage finally settles and is read by the flash ADC.

3.2.2 Proposed In-Memory Computing Operation and Analysis

Binary Activations and Binary Weights

For binary activations, the bitcell produces the XNOR output of '+1' with one strong PU by PMOS and one weak PU by NMOS. It produces the XNOR output of '-1' with one strong PD by NMOS and one weak PU by PMOS. This operation is summarized in the first two rows of the Fig. 3.2(d). The 256 bitcells in a column contribute such XNOR-output-controlled PU and PD circuits and essentially form a resistive voltage divider from the supply voltage to the ground, where RBL is the output. If PU and PD resistances are identical, the RBL voltage (V_{RBL}) will be a symmetric and monotonic function of the XAC value. In practice, they are different due to process variations. Our design is capable of correcting this non-ideality by tuning the PMOS body bias of the bitcell array, which is made as a separate pin in our prototype chip (more details in Fig. 3.14).

The first-order analysis on the relationship between XAC value and RBL voltage is as follows. If the number of rows is N, the range of XAC is from $-N$ to $+N$. Suppose that u is the number of PU cells among N cells in a column, and d is the number of PD cells. As shown in (3.1), we can represent N as the sum of u and d. Given that each PU and PD cell represents bitwise XNOR output of '+1' and '-1', respectively, the XAC result that accumulates all cells' XNOR outputs is formulated

as (3.2). As described in Fig. 3.2(c), the bitwise XNOR output of '+1' and '-1' results in two PU and two PD paths, respectively. This is illustrated in Fig. 3.3, and V_{RBL} can be represented as (3.3) with the resistive divider. Using (3.1) and (3.2), V_{RBL} can be formulated as (3.4), showing a linear relationship with XAC value. Note that V_{RBL} is not affected by the activation/weight patterns as long as they result in the same the XAC bitcount.

$$N = u + d, \tag{3.1}$$

$$XAC = u - d, \tag{3.2}$$

$$V_{RBL} = \frac{2u}{2u + 2d}, \tag{3.3}$$

$$V_{RBL} = \frac{XAC + N}{2N}, \tag{3.4}$$

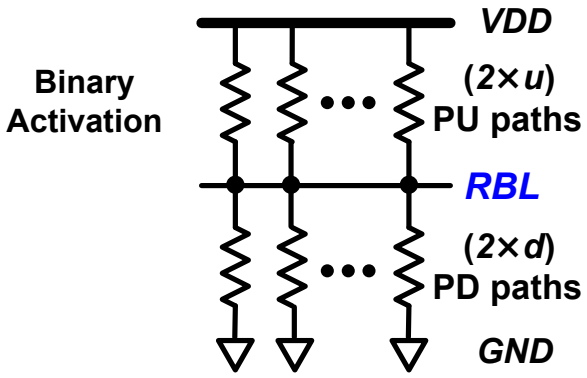


Figure 3.3: PU/PD paths for V_{RBL} with binary activations.

Ternary Activations and Binary Weights

To support ternary activations, we have additionally considered the activation value of '0' and have derived the equations that are similar to (3.4). Suppose that the number of cells that exhibit the bitwise ternary-XNOR output of '0' as z, N would be the

sum of u, d and z ((3.5)). Since those z bitcells do not contribute to the XAC output, the equations for the XAC value are identical for the binary and ternary activation case (i.e., (3.2) and (3.6)). To maintain the same linear relationship between V_{RBL} and N as in (3.4), the z bitcells should contribute z PU and z PD circuits. This is illustrated in Fig. 3.4.

$$N = u + d + z, \tag{3.5}$$

$$XAC = u - d, \tag{3.6}$$

$$V_{RBL} = \frac{2u + z}{2u + 2d + 2z}, \tag{3.7}$$

$$V_{RBL} = \frac{XAC + N}{2N}, \tag{3.8}$$

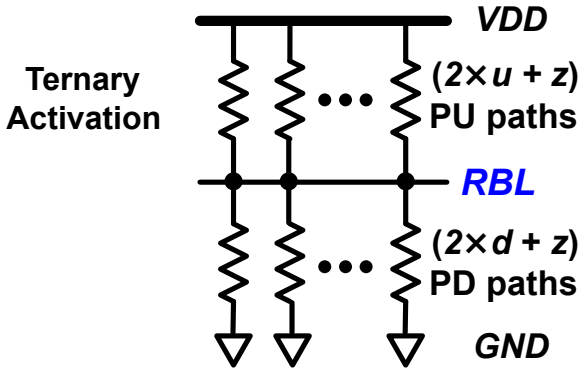


Figure 3.4: PU/PD paths for V_{RBL} with ternary activations.

Since each u and d cell leads to $2u$ PU paths and $2d$ PD paths (one strong plus one weak), each z cell should ideally yield the average strength of the u and d cells, or 0.5 strong PU + 0.5 strong PD + 0.5 weak PU + 0.5 weak PD. However, since T9-T10 (T7-T8) use (close to) minimum size, splitting T7-T10 transistors to support such half/full strengths will complicate and enlarge the XNOR-SRAM bitcell design by about 50% and double the current consumption. The bitcell-embedded ternary

XNOR computation and operation are summarized in Fig. 3.2(c). Note that having z cells to exhibit no PU and PD paths (i.e., turning off '0' activation rows) will make (3.8) deviate from equation (3.4), and introduce further difference in V_{RBL} depending on the number of '0' activation rows. Without changing the bitcell design that implements binary activations and weights, we propose to drive even '0' rows with weak PU/PD and odd '0' rows with strong PU/PD (Fig. 3.2(e)), to effectively support ternary activations. This design is based on the assumption that '0' activations are evenly distributed on even and odd rows. Deviation from this assumption, i.e., the number of even-row zeros and odd-row zeros are not equal, would cause V_{RBL} deviation. According to our post-layout simulation with parasitics annotated, the V_{RBL} variance caused by the mismatch in these two numbers in the ternary VGG-like and ResNet CNNs (for CIFAR-10) and MLPs (for MNIST) that we benchmarked is negligible compared to other variability sources such as transistor mismatch in the XNOR-SRAM cells.

3.2.3 Transfer Function and ADC Optimization

The ADC plays an important role in the computing throughput and accuracy. It digitizes the analog RBL voltage to the digital output. We chose to use the flash ADC using strong-arm comparators for the high-speed advantages. We shared the ADC among 64 columns via a 64-to-1 analog multiplexer for two reasons. First, the RWL drivers could be considerably large to support column parallel operation as the drivers need to supply the current flowing in the resistor dividers. Second, the 64 ADCs would incur a large overhead for the ADC area. As pointed out in [77], the column multiplexing scheme would not degrade the energy efficiency in the first order, since the amount of voltage switching on all the wordlines and bitlines are roughly the same for both column multiplexing and column parallel schemes. Nonetheless,

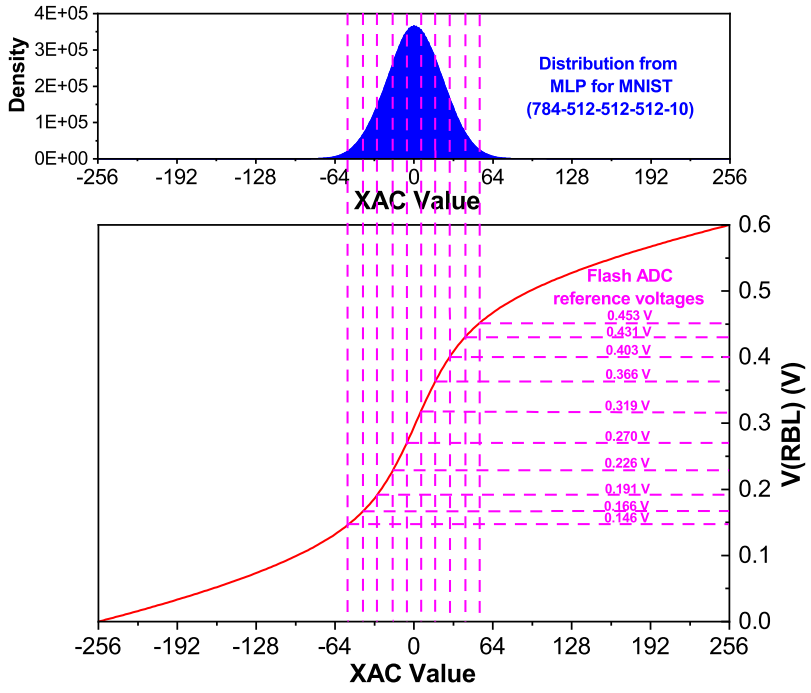


Figure 3.5: XAC is mapped to V_{RBL} with 11-level confined linear quantization.

the column multiplexing in our design hurts the throughput by roughly 64 times, compared to a fully column parallel design.

We investigate the distribution of XAC values. As the data distribution from MLP for MNIST shows (Fig. 3.5), the XAC value is highly concentrated around zero. Exploiting such statistics, we confined the quantization range to the region that covers most data (-60 to +60), within which we linearly divided the quantization levels with reference values. Each quantization reference in XAC value maps to a particular reference voltage (Vref) for the flash ADC (Fig. 3.5). Note that the non-linearity of the PD and PU resistance makes the V_{RBL} transfer function non-linear, placing Vrefs non-uniformly, as shown in Fig. 3.5.

We investigated the required ADC precision based on the MLP for MNIST (784-512-512-512-10) and the VGG-like CNN for CIFAR-10 (128C3-128C3-MP2-256C3-256C3-MP2-512C3-512C3-MP2-1024FC-1024FC-10FC). Fig. 3.6 shows the simula-

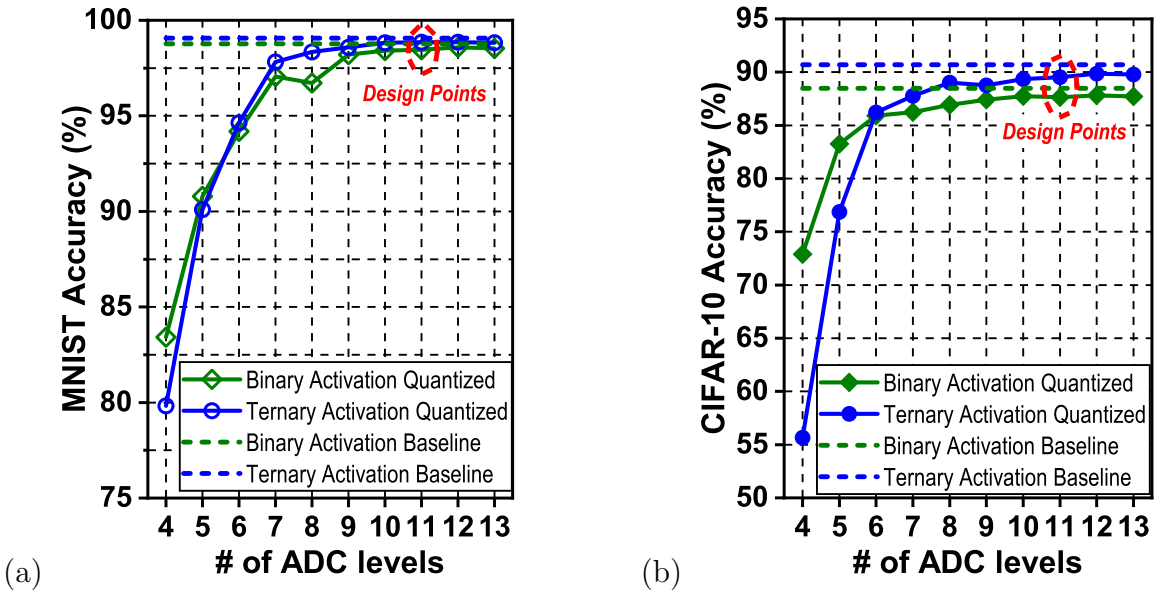


Figure 3.6: MNIST and CIFAR-10 accuracy as a function of ADC levels.

tion results across the different numbers of ADC levels. This simulation ignore analog non-ideality such as offset voltage and transistor variability. We have found that employing 11 quantization levels (i.e., 3.46 bits) results in satisfactory accuracy. In addition, the accuracy saturates for the ADC levels beyond 11. Based on these results, we have designed the 11-level flash ADC, which consists of ten strong-arm comparators.

Note that in [32], we employed a non-linear quantization scheme based on Llyod-Max algorithm [78]. This scheme produces finer grain reference levels where more data exists (i.e., XAC$\simeq$0). However, we have found that this made the difference between two adjacent Vrefs to be very small, increasing the error associated with ADC. In addition, the non-linear quantization scheme does not consider the difference of RBL voltage variance for different XAC values. We have found that near-zero XAC values have larger RBL voltage variance (more details in Fig. 3.13) and ideally require wider ADC quantization intervals. As illustrated in Fig. 3.7, the difference of

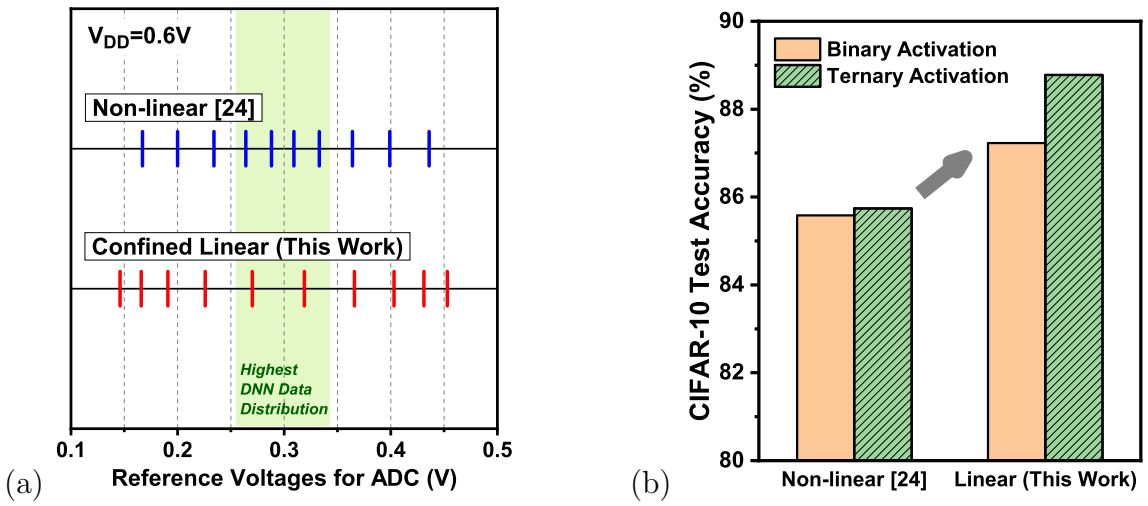


Figure 3.7: Comparison of non-linear quantization and confined linear quantization.

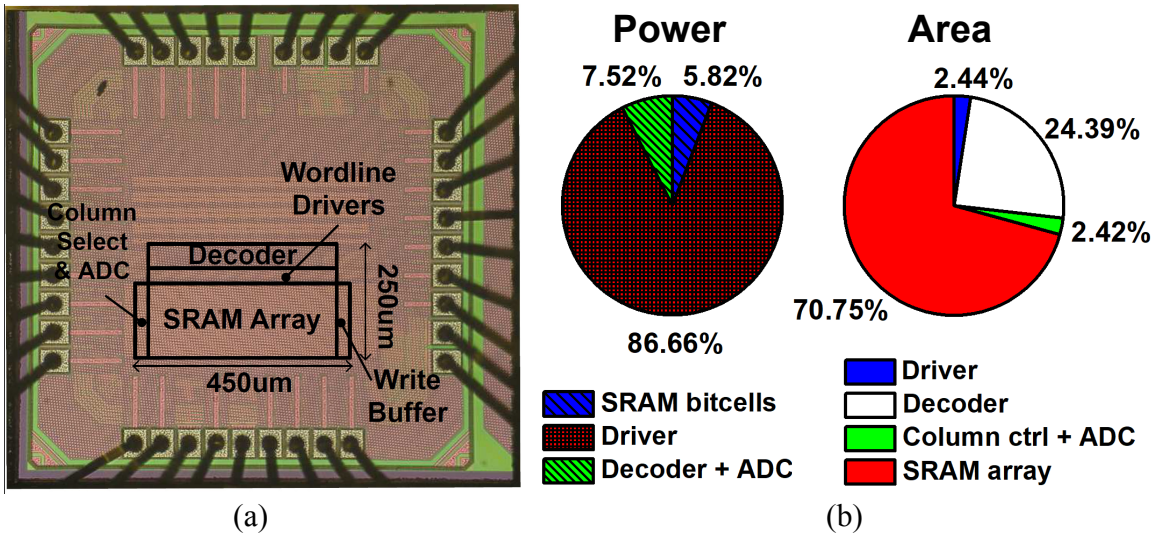
two adjacent Vrefs near the XAC value of zero with the confined linear quantization scheme increases from 21 mV [32] to 49 mV in this work (at 0.6V supply). As shown later, this helps to improve the CIFAR-10 accuracy from 85.7% in [32] to 88.8%.

3.3 Measurement Results

We prototyped the proposed XNOR-SRAM macro in a 65-nm CMOS (Fig. 3.8(a)). The area and power breakdowns are presented in Fig. 3.8(b). The area of XNOR-SRAM is majorly consumed by the bitcell array, where the array efficiency is 70.75%. On the other hand, the XNOR-mode driver dominates the total power as it needs to supplies the current of the restive voltage divider formed for XAC evaluation.

3.3.1 Energy Consumption and Performance Measurements

We have measured the power and energy dissipation of the XNOR-SRAM macro under a range of conditions. First of all, the power consumption depends on the XAC result (Fig. 3.9). This is because most of the power is consumed in the form of the crowbar current in the resistive voltage divider. The input data that corresponds



(a) (b)

Figure 3.8: XNOR-SRAM prototype chip micrograph, power breakdown and area breakdown.

to XAC value near 0 poses the worst case for energy-efficiency. The post-layout simulation shows that the worst-case crowbar current is 1 mA and lasts for 1.26 ns in the second half of a clock cycle at 0.6V. Under this worst case, we measured that XNOR-SRAM consumes 235.5 pJ and takes 54.21 ns for 64 operations of 256-input XAC at 1.0V. Fig. 3.10 shows the energy and the maximum frequency with voltage scaling from 1.0 V to 0. 5V. At 0.6 V, XNOR-SRAM achieves 2.48 fJ per operation. Considering one operation is either ternary multiplication or accumulation, this marks the energy efficiency of 403 TOPS/W.

3.3.2 Comparison to a Digital Baseline Design

As a comparison, we designed a well-crafted digital baseline in the same technology. It performs the exact the same function of the XNOR-SRAM but uses digital XNOR gates, digital adders, and conventional SRAM (Fig. 3.11). This baseline hardware reads 64 binary weights row by row, receives 256 binary/ternary inputs also one

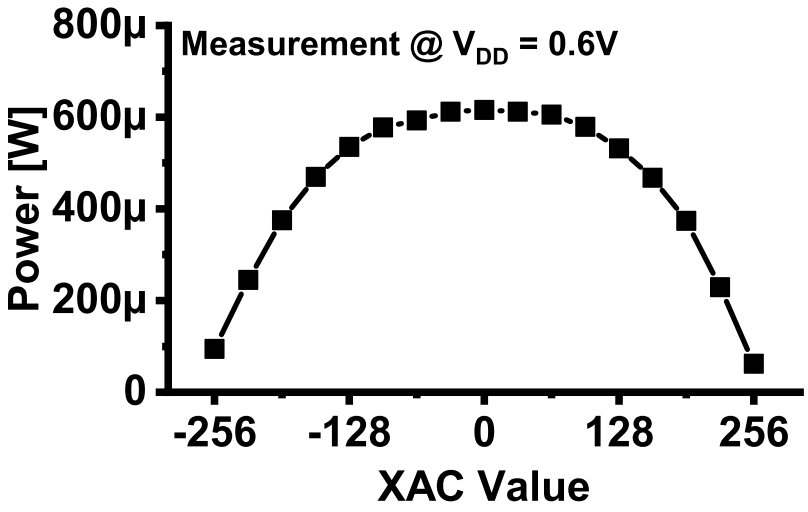


Figure 3.9: Data-dependent XNOR-SRAM power.

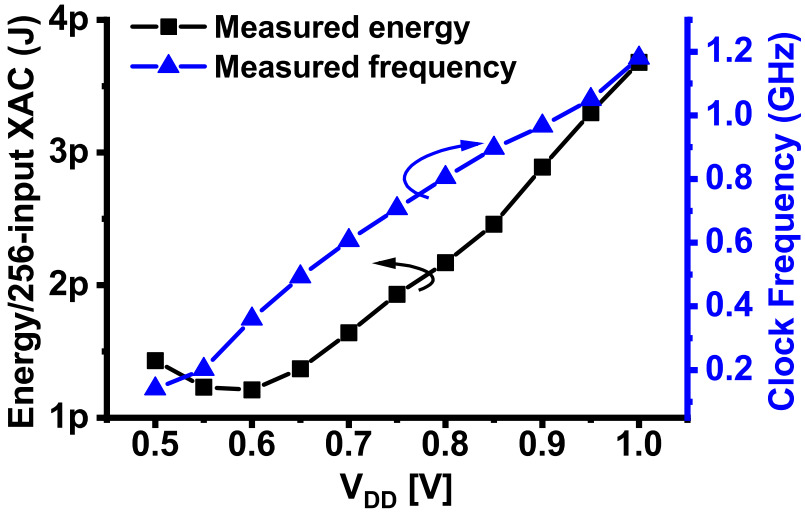


Figure 3.10: Energy and frequency scaling with supply voltage.

by one, and finally accumulates 64 XAC values over 256 cycles. The SRAM for weight memory was generated by a commercial memory compiler. The digital design was synthesized and automatically placed/routed using commercial EDA tools (Fig. 3.11(b)). The digital baseline achieves 500 MHz clock frequency. The post-layout simulation based on parasitic-annotated netlists at 1.0V supply (typical corner, 25°C) shows that the digital baseline consumes 7.81 nJ and 514 ns for the same

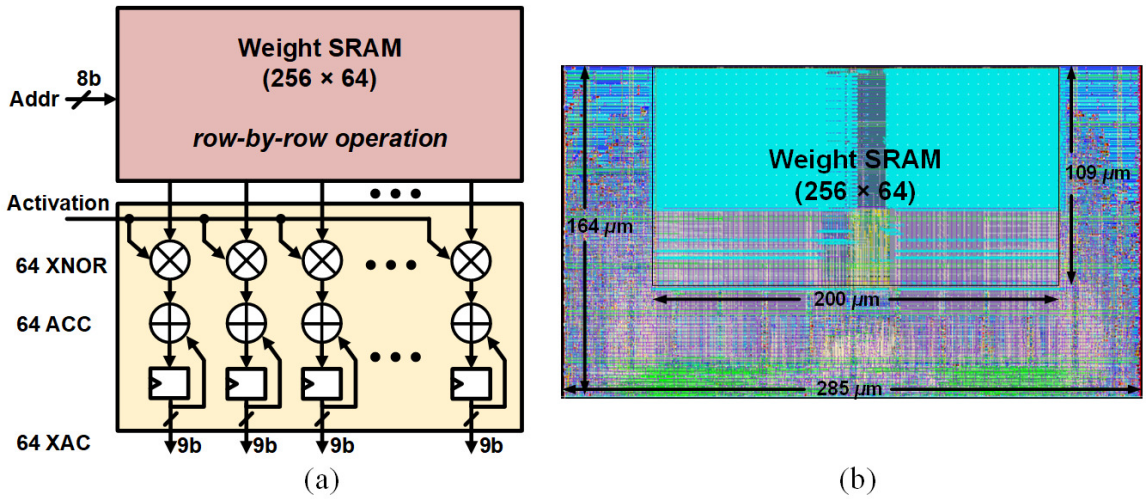


Figure 3.11: Block diagram and layout of digital baseline design for XAC accelerator.

64 256-input XAC operations, which represents 33X worse energy and ∼300X worse EDP than the XNOR-SRAM macro also operating at 1.0V (Fig. 3.12). Similar EDP gain of ∼300X is achieved for XNOR-SRAM down to 0.6V supply. Note that the XNOR-SRAM prototype chips were functional down to 0.5V supply, however additional energy/EDP savings were not achieved below 0.6V, because the circuit delay increases rapidly as the supply voltage gets closer to the near-threshold voltage. During this increased cycle time, the XNOR-SRAM consumes more energy from leakage and crowbar current.

3.3.3 Variability and Compensation

Fig. 3.13 shows the measured V_{RBL} variability resulting from process variation and parasitics across different columns and data patterns, where the top and bottom bars represent $+3\sigma$ and -3σ points, respectively. The highest variation (20mV standard deviation) occurred at the lowest XAC value of 0. The systematic strength imbalance between NMOS and PMOS can skew the transfer function. We addressed this by

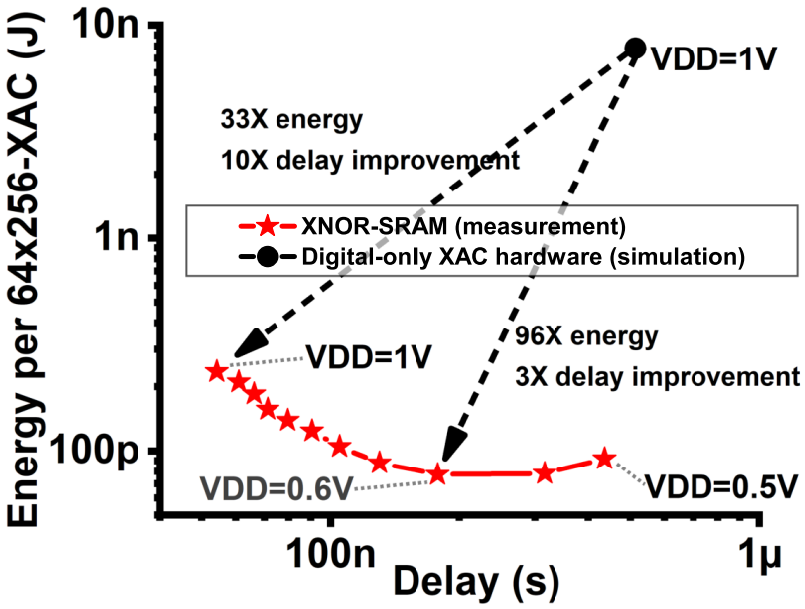


Figure 3.12: Energy and delay comparison with digital baseline.

providing a knob to tune the body bias for PMOS transistors in the XNOR-SRAM array at marginal area/power penalty (Fig. 3.14).

To compensate for the local variability or offset of the ADC, 10 external Vrefs for the 11-level (3.46-bit) flash ADC were first calibrated for the corresponding 10 reference XAC values. For each reference XAC value X_f, 2,000 combinations of random input vectors, columns and weight matrix that yield XAC values that fall in the range of $[X_f - 5, X_f + 5]$ were used to optimize each comparator's Vref. The Vref was initialized at $0.5 \times V_{DD}$. After each combination, the actual comparator output was compared to the ideal output. If it was correct, no change was made to the reference voltage; otherwise, a small and exponentially decayed correction amount will be added to or subtracted from the current reference voltage, depending on whether the ideal output is low or high. Each chip undergoes this type of reference voltage calibration process before performing in-memory computing operation.

After compensation of the ADC offset, there are two remaining major variability

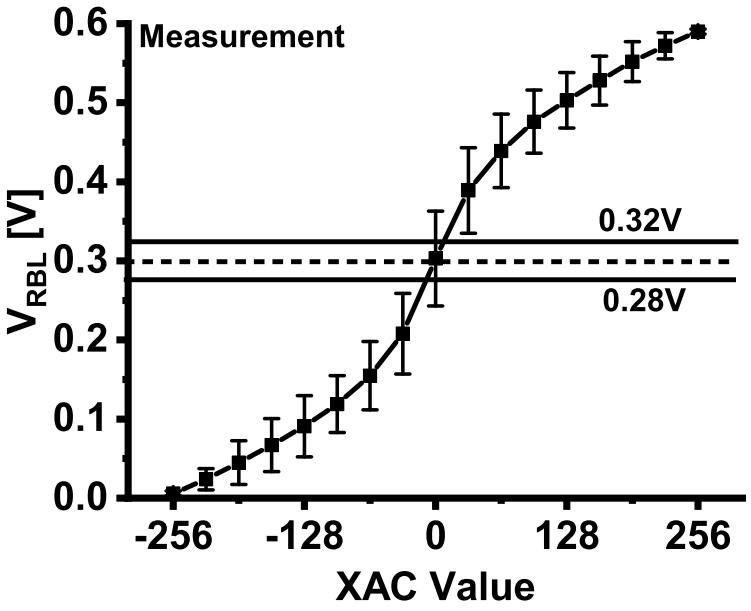


Figure 3.13: Measured transfer function and variability.

sources: i) the transistor mismatch in the XNOR-SRAM cells and ii) the IR drop on RBL wires. The transistor mismatch causes the bitcells in the different rows but in the same column have different PU/PD strength (after the array-wide body bias calibration), making RBL voltage depend on the input/weight pattern even for the same XAC value. On the other hand, the RBL IR drop is a function of input/weight patterns.

Table 3.1: RBL Voltage Variance of a Single Column at 1.0V and 0.6V Supply Extracted from Post-layout Monte-Carlo Simulations.

VDD [V]	ΔV_{RBL} [mV] from mismatch	ΔV_{RBL} [mV] from IR drop	ΔV_{RBL} [mV] from both
1.0	2.71	36.4	36.8
0.6	7.09	6.06	9.33

We performed Monte-Carlo simulations for the parasitic-annotated netlist of a

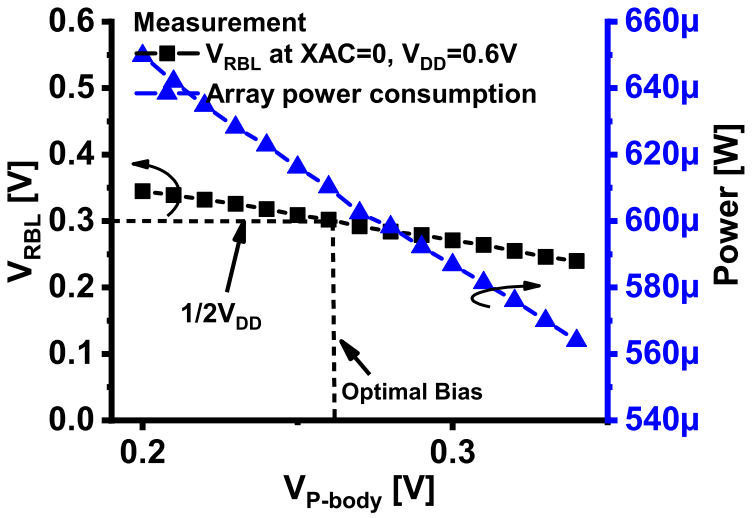


Figure 3.14: Body bias tuning for PMOS/NMOS mismatch.

single column of bitcells to characterize these two variability sources. We used 1,000 random combinations of input/weight vectors that result in the XAC value of 0. To isolate the impact of each variability source, in our extracted simulations, we i) included only the mismatch for the cell transistors; ii) included only the extracted resistance of the RBL; iii) included both two variability sources. Table 3.1 summarized the results of the post-layout simulations. We can see that at 1V, as the current is very large, the IR drop along the RBL contributes most to the overall variation of V_{RBL}. At 0.6V, variation due to IR drop significantly decreases as the current decreases, reducing the overall amount of variation to just a quarter of that at 1V.

3.3.4 The Statistical Model of XNOR-SRAM and Voltage Scaling

We developed the statistical model of XNOR-SRAM as a function of the XAC value. To do so, we measured the ADC output for 1,600 times for each XAC value, 25 times per column. Each time a random test vector that will result in the target XAC value for a given column is generated. Based on 1,600 measured ADC outputs

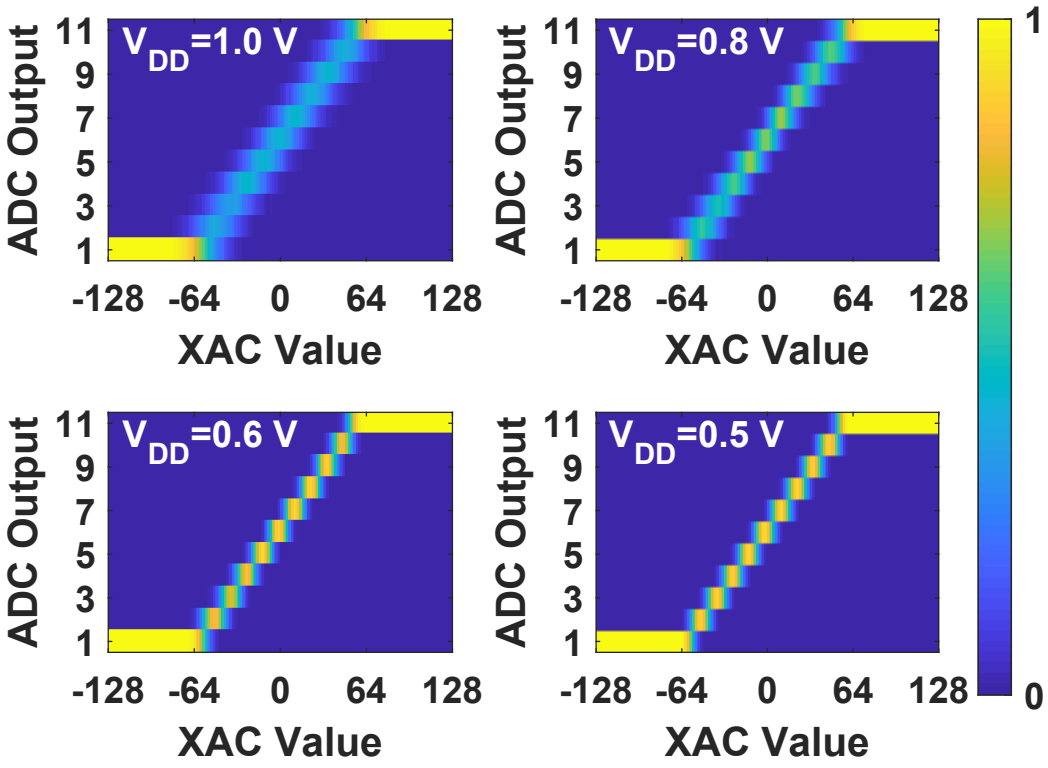


Figure 3.15: Measured ADC output probability distribution as a function of XAC value at V_{DD} of 1.0V, 0.8V, 0.6V, and 0.5V.

for each XAC value, we estimated the probability distribution of the ADC output as function of the XAC value (Fig. 3.15). We iterated this experiment at four different supply voltages of 1.0V, 0.8V, 0.6V, and 0.5V.

Counter-intuitively, Fig. 3.15 shows that, as supply voltage lowers, the ADC output distribution becomes tighter. To see it clearly, we can model the read bitline voltage V_{RBL} as a function of XAC value X and V_{DD} as

$$V_{RBL}(X, V_{DD}) = \bar{V}_{RBL}(X, V_{DD}) \pm \Delta V_{RBL}(X, V_{DD}), \tag{3.9}$$

where $\bar{V}_{RBL}(X, V_{DD})$ represents the average V_{RBL} for given XAC value X under supply voltage V_{DD} over all possible combinations of input and weight vector, $\Delta V_{RBL}(X, V_{DD})$ represents the standard deviation of the actual V_{RBL} over all possible combina-

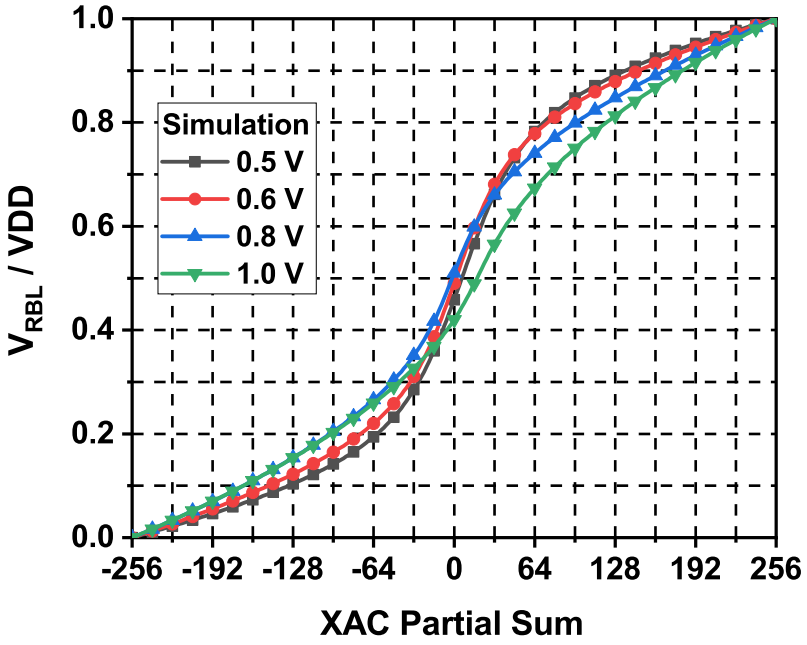
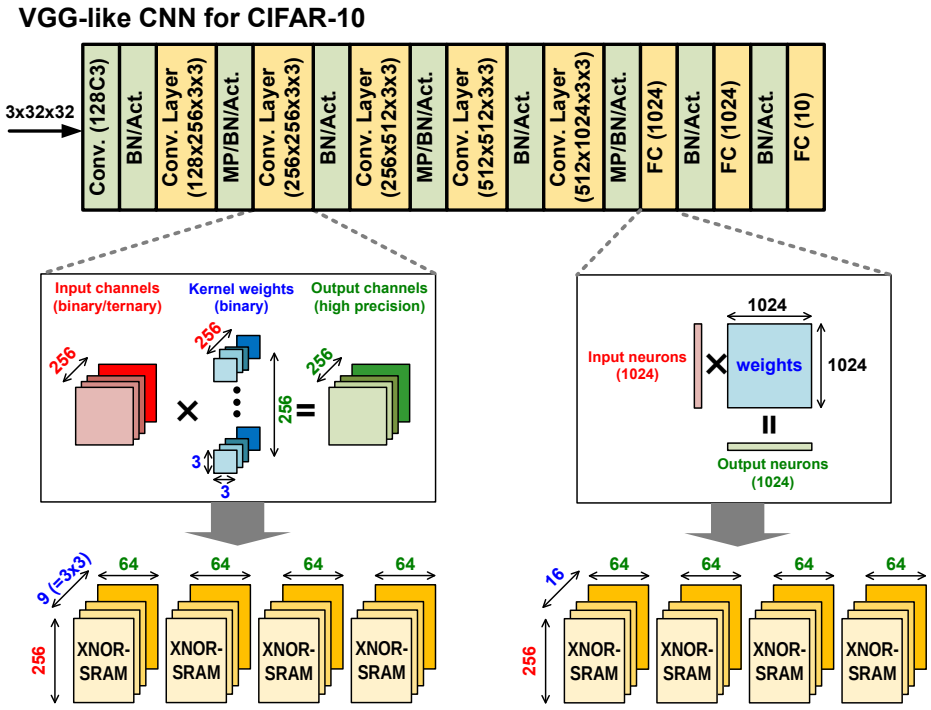


Figure 3.16: Normalized transfer function at different V_{DD}.

tions of input and weight vector for given XAC value X. ADC's Vrefs are calibrated against $\bar{V}_{RBL}(X, V_{DD})$ as aforementioned. ADC quantization error is then governed by the distribution of $\Delta V_{RBL}(X, V_{DD})$ and quantization scheme. The reduced ADC quantization error at lower V_{DD} can be explained from two aspects: reduced $\Delta V_{RBL}(X, V_{DD})/V_{DD}$ and enhanced normalized slope of transfer function.

Voltage Scaling of RBL Voltage Variance

As shown in Table 3.1, according to our post-layout simulation on a single column of XNOR-SRAM array, RBL voltage variance at $X = 0$ reduces from 36.8 mV to 9.33 mV when we scale V_{DD} from 1.0 V to 0.6 V. This reduction in RBL voltage variance majorly comes from the reduction of variance contribution from IR drop along the RBL, which is a result of reduction in current.



(a) (b)

Figure 3.17: Mapping convolution and fully-connected layers of deep CNNs onto XNOR-SRAM arrays.

Normalized Slope of Transfer Function

As V_{DD} decreases, the transfer function slope in near-zero region increases when normalized to V_{DD} as shown in Fig 3.16. As a result, two adjacent XAC values will be more separated in terms of V_{RBL}/V_{DD}, tolerating larger variance in V_{RBL}/V_{DD}.

Combining the above two aspects, as V_{DD} decreases, the enhanced normalized slope of V_{RBL} transfer function and reduced variance of V_{RBL} lead to reduced ADC quantization error as shown in Fig. 3.15.

3.3.5 Strategy for Mapping DNNs onto XNOR-SRAM Arrays

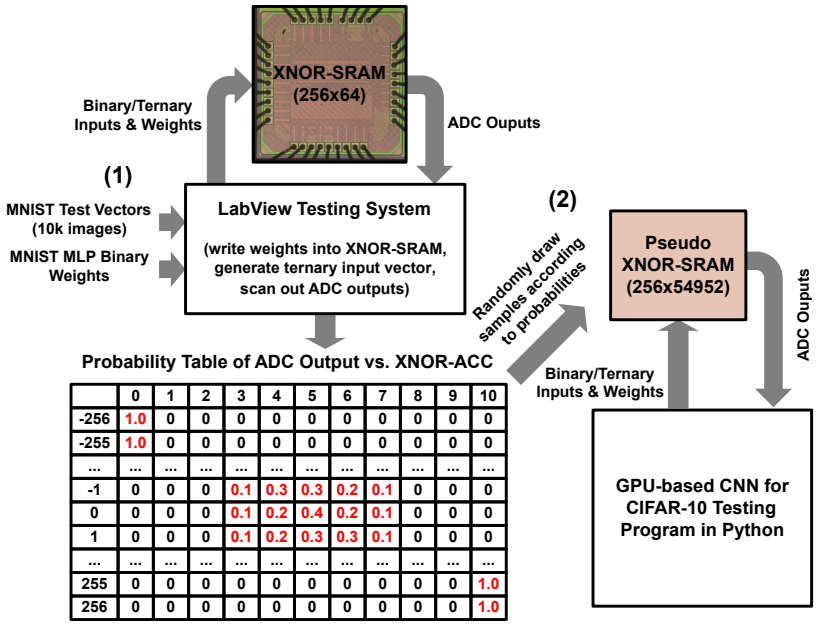
Mapping convolution layers of deep CNNs onto XNOR-SRAM arrays is illustrated in Fig. 3.17(a). We propose to use a mapping strategy where (1) kernels for separate

input channels are stored on separate rows, (2) kernels for separate output channels are stored on separate columns, and (3) weights within each kernel (e.g., 9 weights for 3×3 kernel) are stored in separate XNOR-SRAM macros. The XAC results (partial sums) obtained from separate macros are accumulated digitally, to generate the final sum results for each neuron. This scheme extensively re-uses the input activations, with stationary weights in the XNOR-SRAM arrays.

As illustrated in Fig. 3.17(b), it is straightforward to map fully-connected layers of DNNs, where activations are in vectors and weights are in matrices. This nicely maps to the row drivers for activations and weights stored in the XNOR-SRAM. For the fully-connected layers whose size is larger than 256×64, we break the large weight matrix into a number of small sub-matrices (that fit XNOR-SRAM macros) and accumulate the matrix-vector multiplication results accordingly.

3.3.6 DNN Accuracy Characterization

Using the XNOR-SRAM macro, we evaluated the accuracy of DNNs for MNIST and CIFAR-10 datasets. For MNIST, an MLP with three hidden layers, each with 512 neurons, is used (784-512-512-512-10). For CIFAR-10, we evaluated two deep CNNs: VGG-like CNN and ResNet-14. VGG-like CNN [17] has six convolutional layers and three fully-connected (FC) layers: 128C3-128C3-MP2-256C3-256C3-MP2-512C3-512C3-MP2-1024FC-1024FC-10FC, where nC3 represents a convolutional layer with n 3×3 filters, mFC is a FC layer with m neurons and MP2 is a max-pooling layer with 2×2 pooling size. ResNet-14 [17, 79] consists of 3 basic residual blocks (block widths of 80, 160 and 320), with a total of 13 3×3 convolution layers, two 1×1 convolution layers in short-cut paths (not counted for the number of layers), and 1 FC layer. Starting from the first hidden layer of the MLP/CNN, XNOR-SRAM computes 256-input XACs for MAC/convolution operations in all convolution/FC layers (e.g., all



	0	1	2	3	4	5	6	7	8	9	10
-256	1.0	0	0	0	0	0	0	0	0	0	0
-255	1.0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...
-1	0	0	0	0.1	0.3	0.3	0.2	0.1	0	0	0
0	0	0	0	0.1	0.2	0.4	0.2	0.1	0	0	0
1	0	0	0	0.1	0.2	0.3	0.3	0.1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...
255	0	0	0	0	0	0	0	0	0	0	1.0
256	0	0	0	0	0	0	0	0	0	0	1.0

Figure 3.18: Measurement based simulation framework for CIFAR-10 accuracy evaluation using XNOR-SRAM macros.

yellow-colored layers in Fig. 3.17). Accumulation of XAC outputs, pooling, and batch normalization are performed in digital simulation with bit precisions of 12, 12, and 10, respectively. Note that the digital hardware that executes these functions would degrade the overall energy-efficiency to some extent. Recently, several works have tried to shed a light on this matter [80].

The MNIST accuracy results were obtained entirely from measurements, while the CIFAR-10 accuracy results were obtained from our measurement based simulation framework (illustrated in Fig. 3.18) due to limited scan chain throughput of the prototype chip. Employing the same methodology used to generate the probability distribution in Fig. 3.15, ADC output distributions for each possible XAC value were estimated from measured samples from 10k MNIST test images MLP inference. Each column was sampled from the probability table, obtaining an ADC output for each bitcount, then this mapping was kept for all the 10k CIFAR-10 test images. The

Table 3.2: Measured MLP (for MNIST) and CNN (for CIFAR-10) Accuracy Summary at 0.6V Supply.

Dataset	MNIST		CIFAR-10			
DNN Model	MLP		VGG-like CNN		ResNet-14 CNN	
Activation Precision	Binary	Ternary	Binary	Ternary	Binary	Ternary
SW Baseline Accuracy	98.77%	99.07%	88.60%	90.70%	89.61%	90.80%
HW Measured Accuracy	98.65% (-0.12%)	98.84% (-0.23%)	87.23% (-1.37%)	88.78% (-1.92%)	85.72% (-3.89%)	87.05% (-3.75%)

measured distributions were then used to draw random samples in a GPU-accelerated Python program that simulates XNOR-SRAM XAC and quantization operations for inference of 10k CIFAR-10 test images using trained binary CNN [17]. Our Python simulation program repeated 20 runs with different random seeds, and the average accuracy values are reported.

Table 3.2 summarizes the measured accuracy results of MLP for MNIST and VGG/ResNet-14 CNNs for CIFAR-10 datasets with binary/ternary activations at 0.6V chip #2. It is hypothesized that ResNet-14 CNN has more accuracy degradation than VGG CNN compared to the software baselines, because ResNet-14 CNN is deeper and requires higher partial sum precision to maintain high accuracy. It can be seen that DNNs with ternary activations demonstrate relatively higher accuracy than those with binary activations for both MNIST and CIFAR-10 datasets, and our XNOR-SRAM can execute both ternary and binary activations in a single cycle with the same design change (Sec. 3.2.2).

Fig. 3.19 shows the accuracy characterization of CNN for CIFAR-10 with binary activations/weights across different supply voltages and different chips, where top and bottom bars represent $+\sigma$ and $-\sigma$ points, respectively. With tighter ADC output

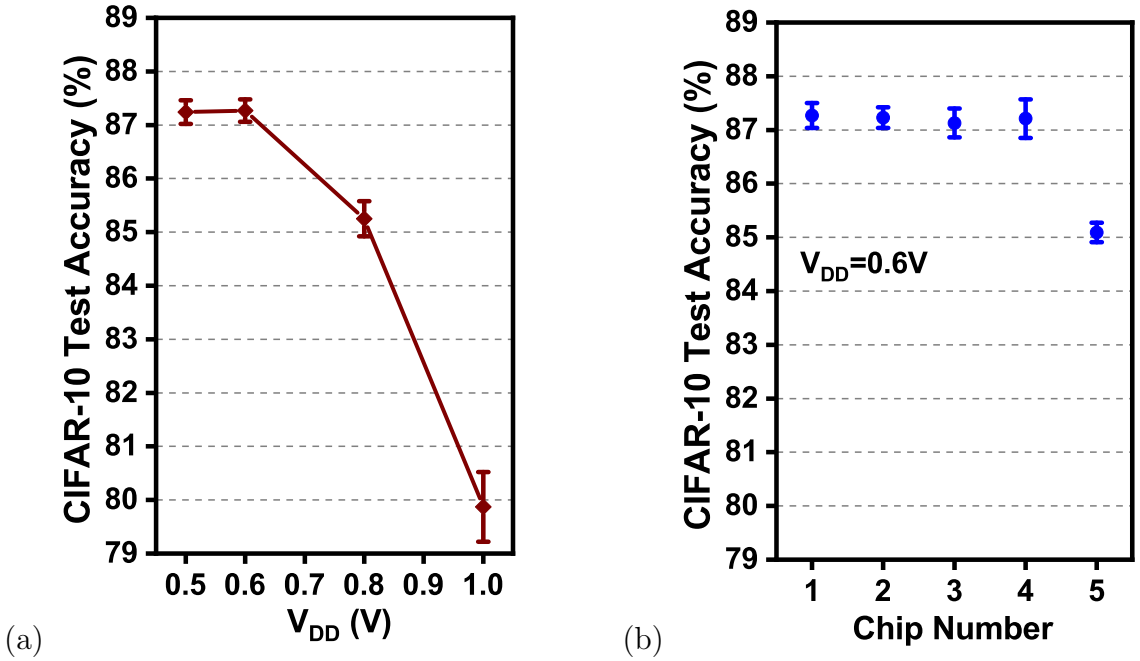


Figure 3.19: Accuracy characterization of binary CNN for CIFAR-10 for (a) Chip #1 at different supply voltages and (b) five different chips at 0.6 V.

distribution for XAC values at lower V_{DD} (Fig. 3.15), the CNN accuracy improves as we lower V_{DD}, down to 0.6V (Fig. 3.19(a)). Fig. 3.19(b) shows the accuracy distribution of five different chips at 0.6V supply, where most chips exhibit a relatively constant mean accuracy of >87%. Note that chip 5 achieves lower accuracy compared to the other four chips we measured. While we are in lack of sufficient access to the internal signals of the prototype chip, we still found that chip 5 behaves similarly to other chips except that the ADC outputs exhibit larger errors as compared to those of other chips.

3.3.7 Ensemble Networks for Accuracy Improvement

In the deep learning literature, ensemble neural networks have been commonly used to improve classification accuracy [1, 2, 81–83], where a collection of networks

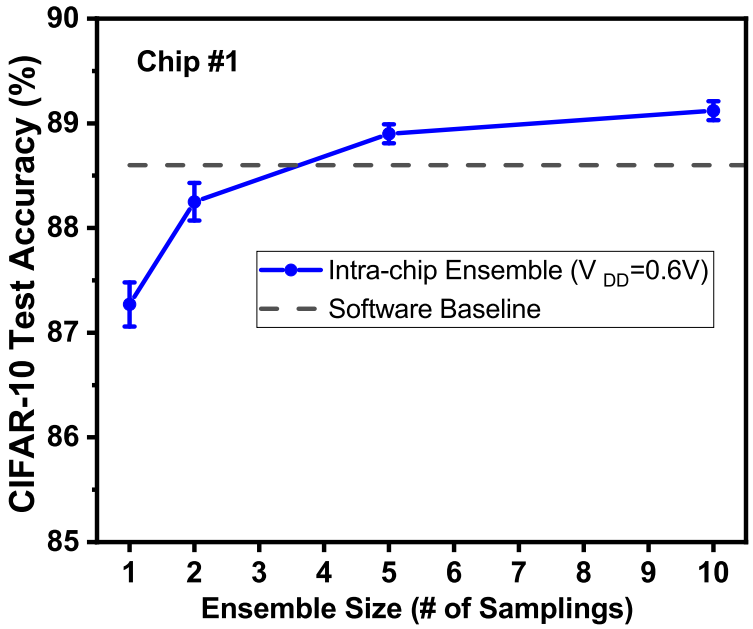


Figure 3.20: Binary DNN accuracy improves with ensemble hardware exploiting intra-die variation.

(typically less than 10) are trained separately with different initial random weights or training data, and the final prediction is calculated as the average of the predictions of all networks models. The reason that ensemble methods improve accuracy is that different models will usually not make the same errors on the test set [84].

Inspired by such ensemble algorithms, we investigated combining multiple XNOR-SRAM arrays from the same chip or different chips, in order to possibly improve the XNOR-SRAM hardware accuracy. Compared to the software baseline, the mixed-signal XNOR-SRAM based DNN accuracy was degraded due to intra-/inter-column transistor mismatch, RBL resistance difference, and ADC offsets. Compared to the ensemble algorithms, the main difference in our proposed *ensemble hardware* for XNOR-SRAM is that, we do not train different DNNs, instead we use identical DNNs but exploit the hardware variability as the source of the difference in prediction errors.

First, we used the measurement data from a single chip (chip #1) and ran-

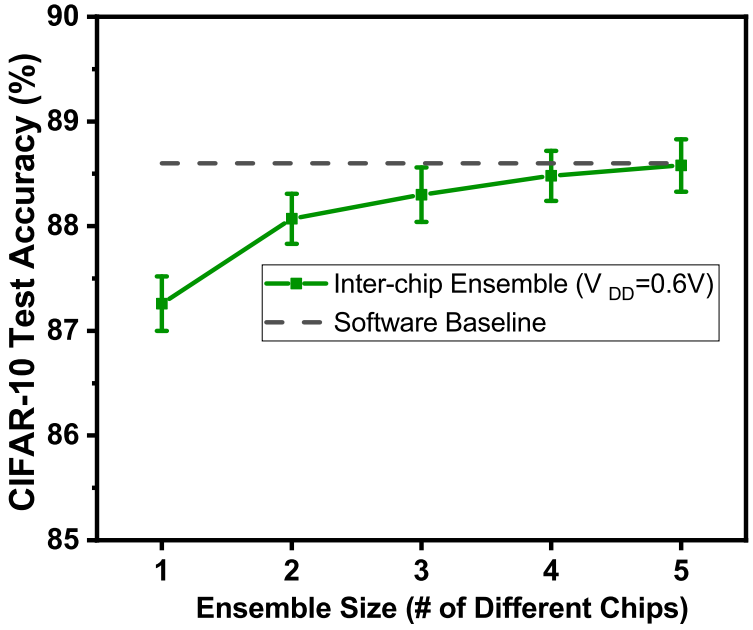


Figure 3.21: Binary DNN accuracy improves with ensemble hardware exploiting intra-die variation from 5 different chips.

domly sampled the probability table in Fig. 3.18 and evaluated the CNN accuracy for CIFAR-10. To test the ensemble hardware exploiting intra-die variation, we iterated this procedure for 2, 5, and 10 times (ensemble size) with different random sampling, and averaged the values of 10 neurons of the last output layer from ensemble hardware for classification. This intra-chip ensemble hardware accuracy results are shown in Fig. 3.20, where considerable accuracy improvement is achieved, while trading off area and energy. For ensemble sizes of 5 or larger, the CNN hardware accuracy values are even higher than the software baseline accuracy.

Next, we used the measurement data from five different chips, and experimented to ensemble different number of chips to exploit inter-die variation. Even though chip #5 exhibits somewhat lower CNN accuracy than other chips (Fig. 3.19(b)), Fig. 3.21 shows that the CNN accuracy for CIFAR-10 continuously improved when we increasingly used more chips. In both Fig. 3.20 and Fig. 3.21, top/bottom bars

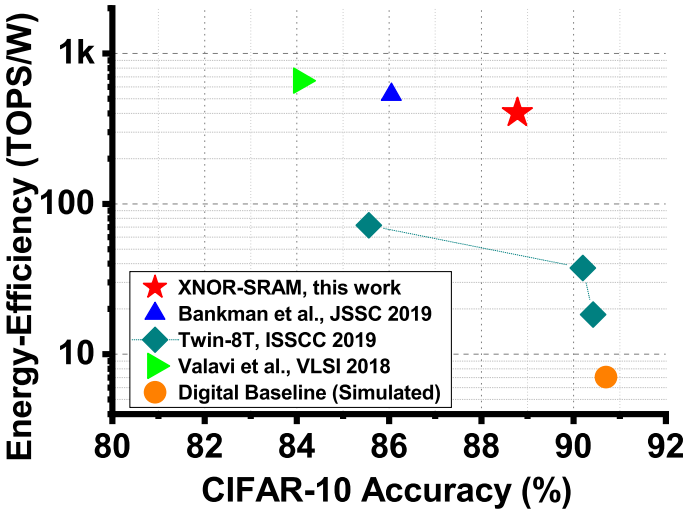


Figure 3.22: Energy-efficiency (TOPS/W) and Cifar-10 accuracy comparison against in-/near-memory computing literature.

represent $+\sigma$/-σ points.

3.3.8 Comparison

Fig. 3.22 shows the comparison of energy-efficiency (TOPS/W) and CIFAR-10 accuracy against prior in-memory computing works. The neural network architecture of ours (XNOR-SRAM), digital baseline, and [31] is the same. [23] uses a simplified version (2×2 convolution filters, etc.) of the same network. [36] uses a different residual CNN (ResNet) for the same dataset of CIFAR-10. Compared to [31], energy-efficiency is ∼38% lower, but CIFAR-10 accuracy is significantly higher by 4.7%. Compared to [36], the CIFAR-10 accuracy is 1.4% lower, but the energy-efficiency is >10X higher. Table 3.3 shows a more detailed comparison to recent in-SRAM computing works [31, 36, 41] and the digital baseline. Our XNOR-SRAM macro improves energy/EDP by ∼100X/300X over digital baseline performing the same XAC operations. Compared to other in-SRAM computing works in the literature, XNOR-SRAM also demonstrates the best trade-off in high energy-efficiency

Table 3.3: Comparison with Recent In-SRAM Computing Works.

	[28]	[31]	[36]	Digital baseline	**This work**
Technology	65nm	65nm	55nm	65nm	**65nm**
SRAM bitcell	10T	10T1C	8T_MSB 8T_LSB	6T	**12T**
SRAM array size	256×64	64×64 64 tiles	64×60	256×64	**256×64**
# of rows turned on	16	64	9/18	1	**256**
Power supply	0.8-1.0V	1.0V	1.0V	1.0V	**0.6-1.0V**
Column sensing	7b integrating ADC	SA + DAC	5b SAR ADC	digital adder	**3.46b flash ADC**
Energy efficiency (TOPS/W)	51.3 (A: 6b, W: 1b, mult./avg.)	658 (binary XAC)	18.4-71.9 (A: 1/2/4b, W: 2/5/5b)	4.2 (ternary XAC)	**403 (ternary XAC)**
GOPS/mm^2	57	1498	Not reported	1369	**5461**
MNIST accuracy (%)	98.0	98.58	99.02/99.52 (A: 1/4b, W: 2/5b)	99.07	**98.84**
CIFAR-10 accuracy (%)	N/A	84.09	85.56/90.42 (A: 1/4b, W: 2/5b)	90.70	**88.78**

and high CIFAR-10 classification accuracy.

3.4 Conclusion

In this chapter, we present an in-memory computing SRAM macro titled "XNOR-SRAM" that computes ternary-XNOR-and-accumulate operations in binary/ternary MLP and CNNs with high energy-efficiency and high accuracy. Our 256×64 XNOR-SRAM asserts all 256 rows simultaneously, performing a 256-input ternary XAC in a

single cycle, via analog accumulation of bitwise XNOR results on the read bitline voltage, which is digitized using an optimized 11-level flash ADC embedded in the periphery. Our 65nm XNOR-SRAM prototype achieves energy-efficiency of 403 TOPS/W for XAC operations and 88.8% test accuracy for CIFAR-10 dataset, achieving 33X better energy and 300X better energy-delay product than digital ASIC baseline with off-the-shelf SRAM. Compared to non-linear quantization, the proposed confined linear quantization scheme improves CIFAR-10 accuracy from 85.7% to 88.8% owing to wider reference voltage intervals. Further accuracy improvement has been shown by employing an ensemble hardware of XNOR-SRAM arrays/chips implementing the same DNNs, exploiting intra-/inter-die variation.

Chapter 4

C3SRAM: A CAPACITIVE IMC SRAM MACRO

4.1 Introduction

As deep convolutional neural networks (DCNNs) continue to demonstrate improvements in inference accuracies [1, 2, 85–87], deep learning is shifting towards edge computing. This development has motivated works on low-resource machine learning algorithms [16, 17, 88–93], and their accelerating hardware [7, 8, 19, 20, 94]. The most common operations in DCNNs are multiply-and-accumulate (MAC), which dominates power and delay. MAC operations have high regularity and parallelism, therefore is very suitable for hardware acceleration. However, the amount of memory access severely limits the energy-efficiency in conventional digital accelerators [7,8,19,20,27]. As a result, in-memory computing (IMC) has become increasingly appealing to DCNN acceleration.

IMC is the design approach that performs highly-parallel computation inside the memory block without explicit row-by-row memory access. Recent IMC works [27, 29, 31, 32, 36, 38, 40, 41, 95–100] show significant energy-efficiency and throughput advantages over conventional architectures. These benefits come at the cost of accuracy degradation from analog-mixed-signal (AMS) computing non-idealities. Hence, robust computing mechanisms and error-tolerant algorithms are the IMC design's main considerations and challenges [77].

This chapter presents an IMC SRAM macro based on capacitive-coupling computing (C3), hence named C3SRAM. The prototyped macro is 256×64 in size and computes 64 256-input binary-MAC operation (bMAC) in parallel. The macros can be

used in a modular fashion to support networks of arbitrary size. The 65-nm prototype chip demonstrates 671.5 TOPS/W energy efficiency, and 1,638 GOPS throughput, which is 3,975X improvement in energy-delay product (EDP) than digital baseline. It achieves 98.3% accuracy for MNIST and 85.5% for CIFAR-10 dataset.

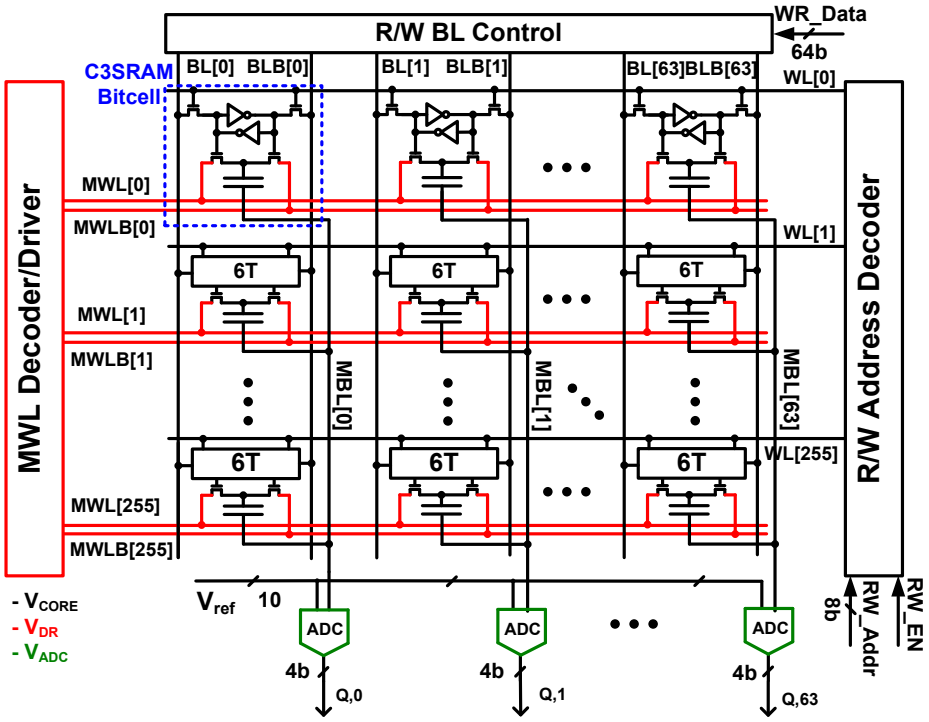


Figure 4.1: Architecture of C3SRAM in-memory computing macro.

4.2 Architecture and Operation

4.2.1 Memory Array Operation

We present the C3SRAM architecture and the operations in detail. Fig. 4.1 presents the architecture of the proposed macro. C3SRAM performs a fully-parallel vector-matrix multiplication of 256 binary inputs and 256×64 binary weights. The macro consists of a 256×64 memory cell array, SRAM peripherals for read/write operations, input activation decoder/driver, and per-column flash analog-to-digital

converters (ADCs).

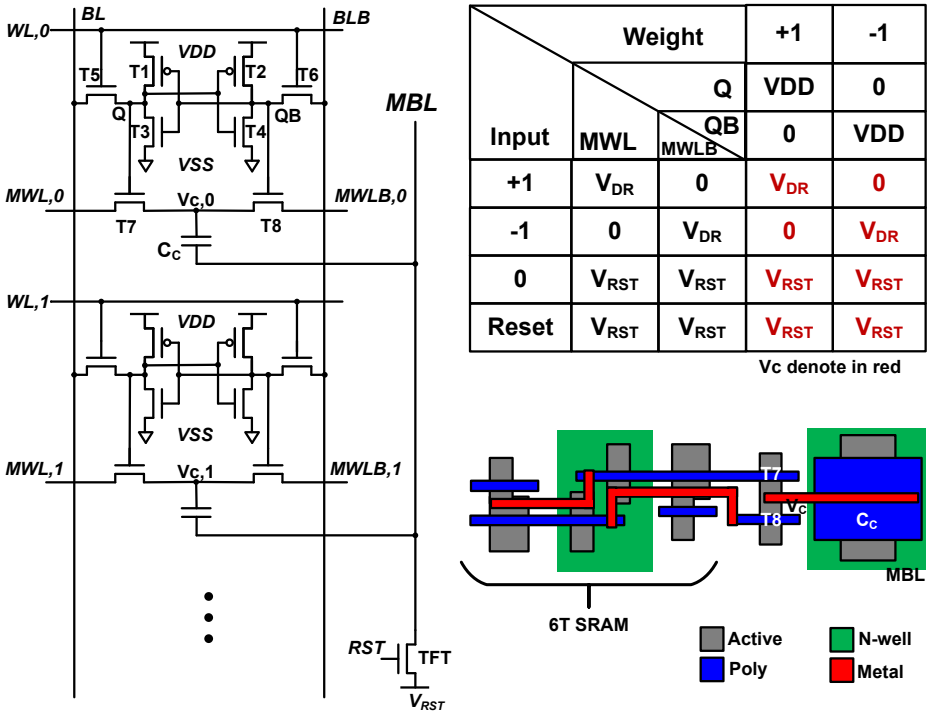


Figure 4.2: C3SRAM bitcell design and in-cell MAC operand table.

Fig. 4.2 shows the 8T1C bitcell layout of the proposed design, a circuit diagram showing two bitcells in a column, and the table of XNOR operands. The bitcell is 80% larger than the conventional 6T bitcell in the same logic design rule, due to the two additional pass transistors and one capacitor constituting 27% of the bitcell area. The capacitor is implemented as MOSCAP for high capacitive density. A MOMCAP covering the area of a C3SRAM bitcell (MOMCAP can be placed on top of transistors with no area overhead) has 80% lower capacitance than CC (4 fF).

To perform binary dot product, the cap is charged/discharged by MAC wordlines (MWL/MWLB) via the pass transistors, which are gated by the stored weight and its complement. Since the pass transistors are NFETs, there would be highly variable threshold voltage (Vt) drop across T7/T8 if the MWLs and the memory core have

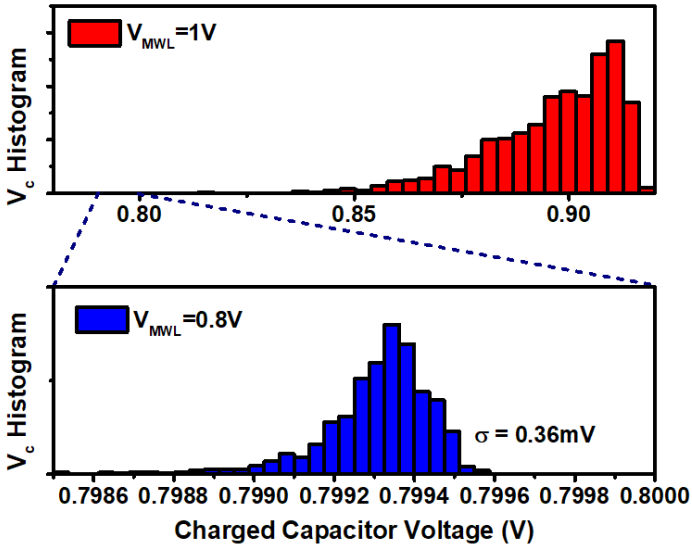


Figure 4.3: Threshold voltage variability effects on charged capacitor voltage.

the same voltage source. To avoid the variability problem, we implement T7/T8 with LVT devices, and set a separate source VDR to drive the MWLs, at 200mV lower than VCORE (e.g., 0.8V VDR for 1V VCORE). We ran Monte Carlo SPICE simulations including only CC, T7, and T8 to isolate the variation of the pass transistors' effect on capacitor charge. As shown in Fig. 4.3 histograms, the x-axis (note the different scales) shows the voltage level at the VC node. Given the 200mV margin, the variation of VC has the small sigma of only 0.36mV. T7/T8 decouple memory function and compute function to avoid potential read and write disturb [40, 97].

The bMAC operation of C3SRAM is shown in Fig. 4.4. There are two steps in this operation; each is completed in a half cycle duration. In step 1, each column's MAC bitline (MBL) is pre-charged via the footer TFT to VRST =0.5·VDR. VRST is set near the voltage corresponding to bMAC output of 0 (nominally 0.4V). This is done to minimize the voltage swing on the MBL nodes since typical bMAC outputs in BNNs have a narrow distribution near 0 value. In the same step, each row's MWL

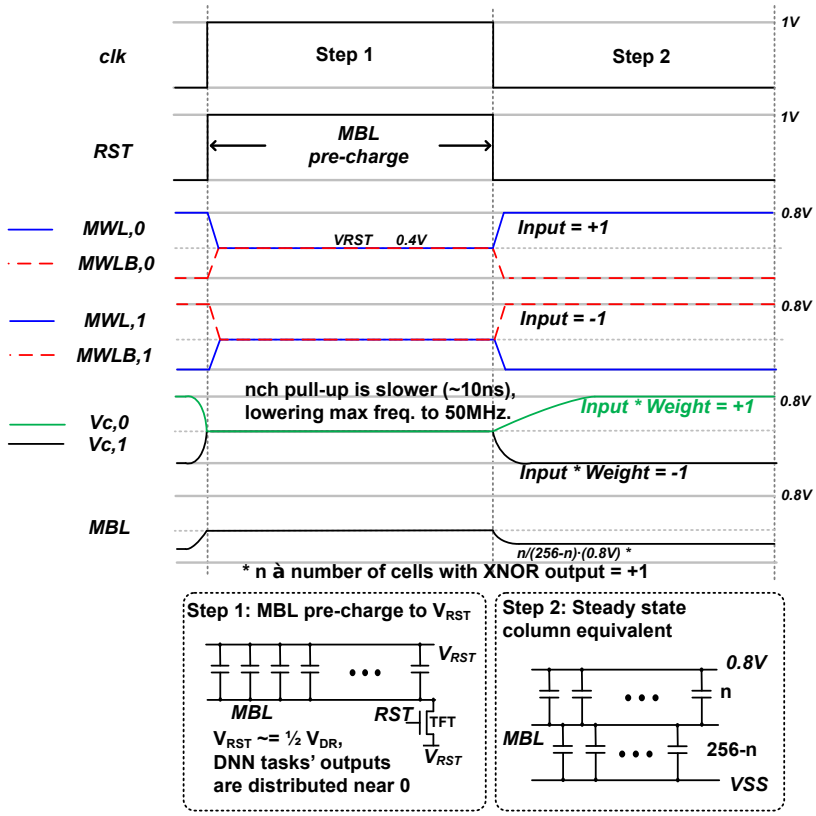


Figure 4.4: Capacitive coupling based in-memory computation of bMAC.

and MWLB are likewise reset to VRST such that there is no voltage potential on bitcell capacitors. At this step, the capacitors are effectively arranged in parallel where both nodes are reset to the same voltage, as illustrated in Fig. 4.4 bottom left.

In step 2, the footer is turned off. The 256 input activations (denoted as In_i) are applied to 256 MWLs/MWLBs in parallel. For In_i = +1 (-1), MWL is driven from VRST to VDR (VSS), while MWLB is driven to VSS (VDR). For Ini = 0, both MWL and MWLB remain at VRST without consuming dynamic power. When the weight is +1 (-1), the voltage ramping via T7 (T8) induces a displacement current through capacitor CC (4 fF) in the bitcell, whose magnitude is:

$$I_c = C_c \cdot dV_{MWL(B)}/dt. \tag{4.1}$$

The charge transferred from the bitcell to MBL is:

$$Q_{ci} = \int_0^{t1} I_c dt = 1/2 C_c V_{DR}, \tag{4.2}$$

where t_1 is the time it takes VMWL to reach VDR. The shared MBL voltage is set to:

$$V_{MBL} = C_c V_{DR} \sum_{i=1}^{256} XNOR_i/(256C_c + C_p). \tag{4.3}$$

where $XNOR_i$ is the XNOR output of the i-th bitcell and Cp is the parasitic capacitance of MBL plus the input capacitance of the ADC. At this step, each column can be seen as two sets of parallelly-connected capacitors, which are in turn connected in series between VDR and VSS, forming a capacitive voltage divider as illustrated in Fig. 4.4 bottom right.

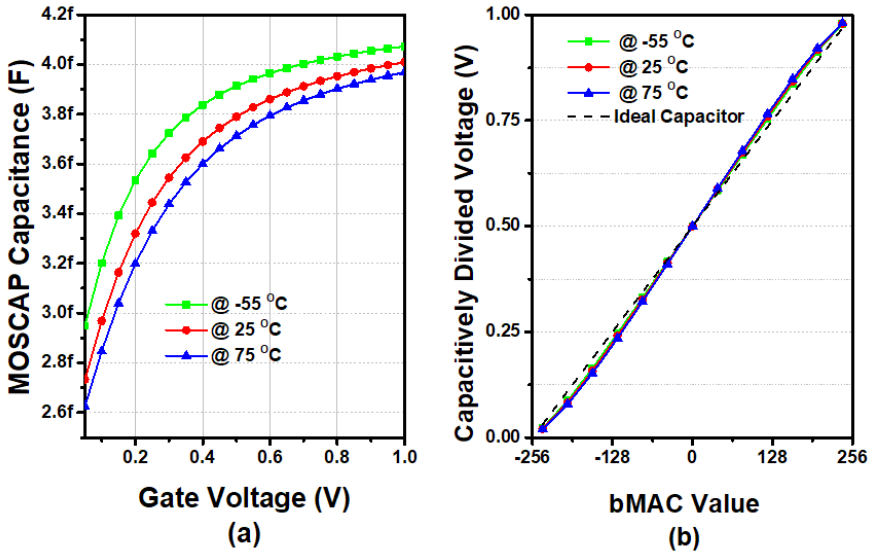


Figure 4.5: (a) MOSCAP capacitance at TT corner simulation shows variation across temperature as well as the gate voltage. (b) MOSCAP capacitive voltage divider transfer function at various temperatures.

MOSCAP's high capacitive density provides bMAC transfer curve a wider full-

scale range (FSR) than using the less capacitively-dense MOMCAP [31]. Given the same level of MBL parasitics, the FSR loss from using MOMCAP would be 80% higher than using MOSCAP. MOSCAP capacitance is dependent on temperature and gate voltage. Fig. 4.5(a) shows CC change over temperature and gate voltage. Fig. 4.5(b) shows the simulated transfer function of a capacitive voltage divider composed of CC. The good news is that the temperature-related non-ideality of MOSCAP has small impact on the transfer function stability. The voltage-related non-linearity gives the transfer function a slight sigmoidal shape which in fact could give some benefit to ADC (negligibly small in our design) because the slightly steeper slope in the region of interest provides slightly higher margins for reference voltages.

4.2.2 ADC Operation

The bMAC output of each column is a pre-activation partial sum. To digitize these values, C3SRAM includes an 11-level flash ADC per column. Each ADC consists of 10 double-sampling-based self-calibrating single-ended comparators (Fig. 6 top). Each comparator consists of an offset-cancelling capacitor followed by an inverter chain, where the first inverter acts as an amplifier.

The ADC operation has two steps (Fig. 4.6): during bMAC computation (step 2), MBL connects to the comparator input capacitor. The input and output of the first inverter are closed, placing the inverter in the high voltage gain region. In step 3, the input node of the capacitor switches to the reference voltage, and the negative feedback path is turned off. The voltage differential between VMBL and Vref then causes (dis)charging on the capacitor. The inverter previously balanced at the trip point is driven high or low according to the direction of the induced current. The gain-stage inverter chain completes the amplification to digital domain.

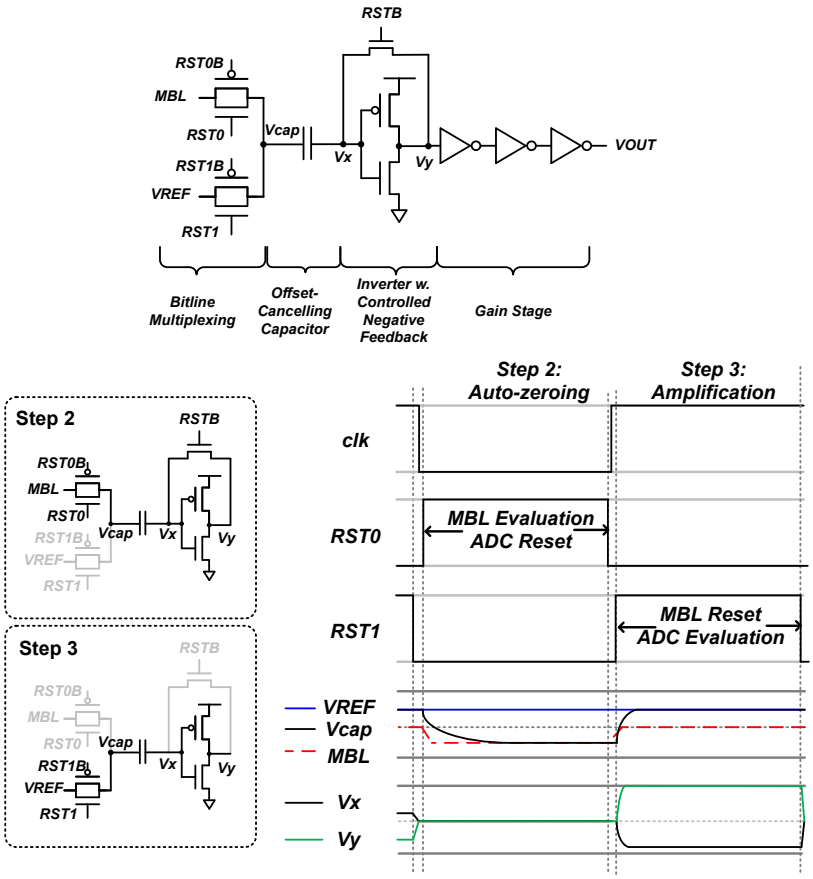


Figure 4.6: Operation of the double-sampling self-calibrating single-ended comparator.

4.2.3 Signal Switching Order

In the aforementioned three-step procedure, relevant signal transitions in step 1 and step 3 are decoupled in separate modules, meaning that while the digital output is being evaluated by the ADC, the memory array can begin computing the next batch of bMACs. This allows a half-cycle pipeline where step 1 (Fig. 4.4) and step 3 (Fig. 4.6) operate concurrently.

The bMAC operation is timing sensitive. To minimize analog non-idealities, concurrent signal switches described in the previous subsections must follow a strict order,

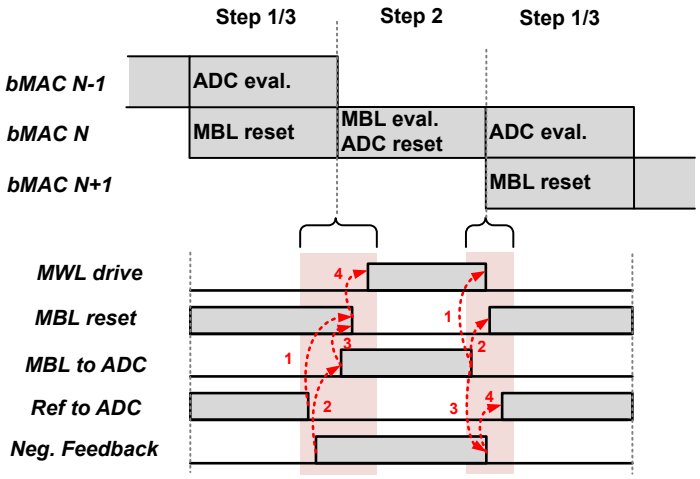


Figure 4.7: Signal transition order for reducing analog non-idealities.

shown in Fig. 4.7. We implemented timing control circuitry with minimal delay elements to guarantee the correctness of the signals' order. The relevant transitions from steps 1 and 3 to step 2 follow this order: 1) the reference voltage must be disconnected from the comparator input capacitor before MBL leaves reset, otherwise, reference voltage source would inject charge onto MBL floating node; 2) the negative feedback on the inverter stage must turn on before MBL is connected to the input capacitor; otherwise, VMBL will be affected by the induced current of inverter gates driven to trip point; 3) MBL must be connected to the comparator input capacitor before MBL leaves reset, otherwise, the charge differential of reference voltage stored on the capacitor will be injected into the floating MBL; 4) MWL cannot be driven until MBL is floating, otherwise, some coupling current would be discharged.

The relevant transitions from step 2 to step 1 and 3 follow this order: 1) MBL must disconnect from comparator input before MWL drivers switch to reset voltage, otherwise, the input changes will induce current on the MBL; 2) also, MBL needs to disconnect before MBL reset footer turns on, otherwise, V_{MBL} stored on the input capacitor would begin to reset as well; 3) also, MBL needs to disconnect before the

negative feedback is turned off, since the act of disconnection can disturb the inverter input which is at the sensitive trip point; 4) the negative feedback needs to switch off before reference voltage is connected to the comparator input, otherwise, the charge differential would be (dis)charged via the feedback path.

4.3 Algorithm and Hardware Specification

In this section, we determine the design specification pertaining to algorithmic support: activation precision and pre-activation quantization levels.

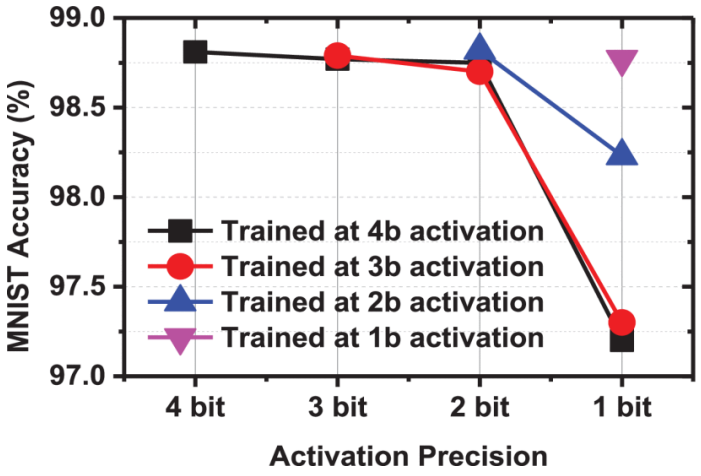


Figure 4.8: MLP on MNIST dataset inference accuracy losses at various levels of activation precisions.

4.3.1 Activation Bit Precision

Operating using the same IMC hardware, inference accuracy loss of a BNN is found less than that of a BWN with multi-bit activation [100]. One reason is that the analog representation of multi-bit activation requires additional domain conversion through DAC [36]. Another reason is that network models trained at higher precision are more sensitive to AMS error. As the study in [101] suggests, robustness is a

benefit of weight redundancy. In similarly accurate models, error in multi-bit MAC is more severe than bMAC due to BNNs' weight duplication scheme.

We examine this issue by applying various levels of stochastic error during the inference of the MNIST dataset. The network topology used in this examination is a binary-weight multi-layer perceptron (MLP) consisting of three fully-connected (FC) hidden layers, each with 512 neurons. The network is trained at various activation precisions from 1 to 4 bits. The trained MLPs are then mapped on the C3SRAM for testing inference accuracy, each time with decreasing the pre-activation resolution to simulate increasing level of stochastic noise. Here, we use the digital representation for activations as in [100], i.e., activations are fed in a bit-serial fashion and outputs are accumulated using digital adders. As shown in Fig. 4.8, at the same level of stochastic error, networks trained at higher precision degrade more. We conclude that, for IMC hardware running multi-bit activation BWN to achieve comparable accuracy as BNN, the hardware may need more resources to compensate for AMS errors and this could result in inefficiencies. For this reason, the proposed C3SRAM primarily supports BNN acceleration.

4.3.2 Partial Convolution Quantization Levels

In practical neural networks, a typical convolution filter is too large to fit in a column of memory cells, therefore would be split into several C3SRAM arrays. In such cases, each array produces partial convolution results which are then accumulated in digital peripheral to produce final output.

For the 256-row C3SRAM, the full resolution of partial convolution results is 8 bit. For an ADC to achieve this high resolution, it would incur considerable area, power, and latency overhead. The objective here is to use a lower resolution ADC design that is still able to maintain the final accuracy.

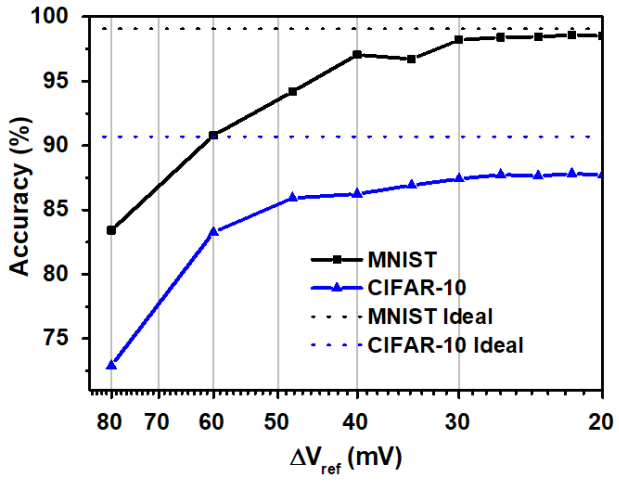


Figure 4.9: MNIST and CIFAR-10 inference accuracy increase as quantization resolution of pre-activation partial sum (256-input) increases.

To find the appropriate ADC resolution that can maintain inference accuracy, we examine the effect of quantization on the MNIST and CIFAR-10 datasets. We use the same MLP described in the previous subsection for the MNIST dataset. For CIFAR-10, we use a VGG-like convolutional BNN [9], with six convolutional layers and three FC layers. For partial convolution results in these network models, we apply several quantization levels, successively reducing ΔVref. Fig. 4.9 shows the effect of quantization on the task accuracy. We find that in both cases the accuracy reach saturation at 30 mV ΔVref which corresponds to 5-bit resolution given the 640 mV FSR. This is consistent with results in [41] where 5-bit ADC is used to achieve high accuracy on MNIST dataset using LeNet-5.

To further reduce the cost of the ADC, the pre-activation distribution can be exploited to reduce unnecessary hardware. Partial convolutions are not uniformly distributed for the entire range of the activation function input. Rather, they are usually narrowly distributed around 0 which is the point of nonlinearity in common

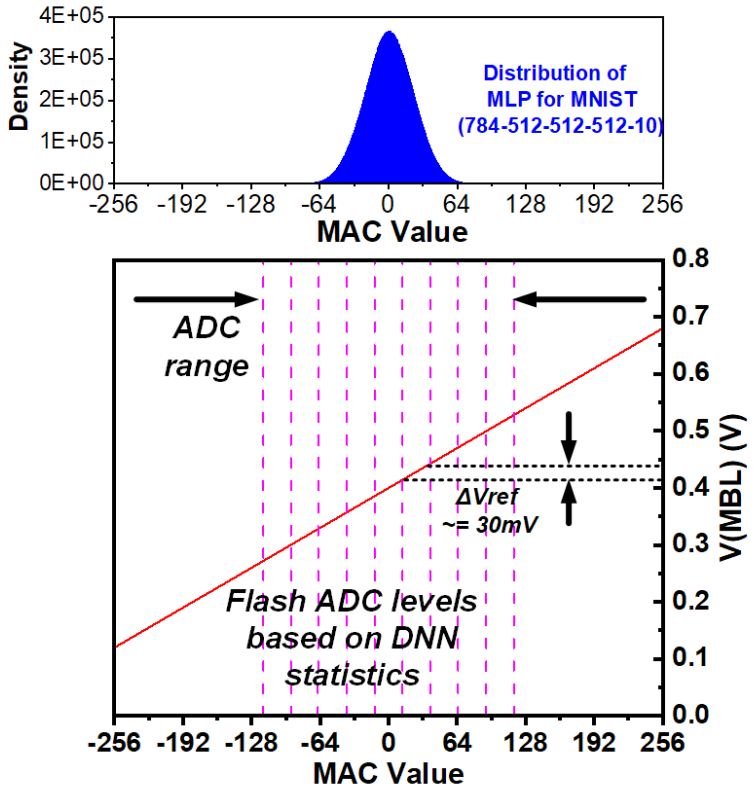


Figure 4.10: The bMAC distribution of MLP for MNIST and quantization in limited ADC range.

BWN/BNN activations functions ReLU/binary step. Fig. 4.10 top shows the MLP bMAC distribution. Since the data is distributed in a small region of the FSR, the ADC range can be confined to this region without accuracy loss. Fig. 4.10 bottom shows an illustration of an ideal transfer function and linear quantization levels in a confined range. The x-axis is the bMAC output value; the y-axis is VMBL corresponding to the bMAC output. In the voltage region corresponding to bMAC value from -120 to +120, only 11 reference levels (ΔVref = 30 mV) are needed to match 5-bit ADC resolution.

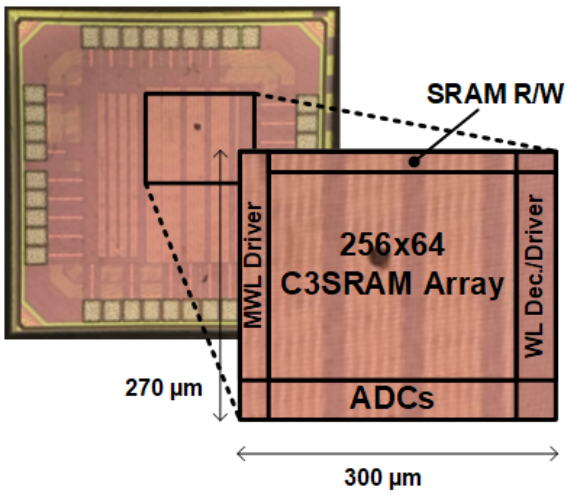


Figure 4.11: The bMAC distribution of MLP for MNIST and quantization in limited ADC range.

4.4 Measurements and Analysis

The C3SRAM macro is implemented in a 65-nm CMOS technology. Fig. 4.11 shows the micrograph of the test chip. The macro has a capacity of 16 kb at the footprint of 0.081 mm^2.

4.4.1 Energy and Throughput

The memory macro has 2 kB capacity. For bMAC computations, the macro operates at a maximum frequency of 50 MHz, limited by minimal sized footer discharging ADC input capacitors. The macro computes 64 independent 256-input bMACs per cycle. The throughput is 2×256×64/20ns = 1638 GOPS. The compute density is thus 20.2 TOPS/mm^2. At the operating voltages of 1V core supply (VCORE), 0.8V driver (VDR), and 0.6V ADC (VADC), it consumes 49 pJ excluding input/output data movement, reaching an energy efficiency of 671.5 TOPS/W, a 3,975X improvement in energy-delay-product (EDP) over the digital baseline formulated in [100],

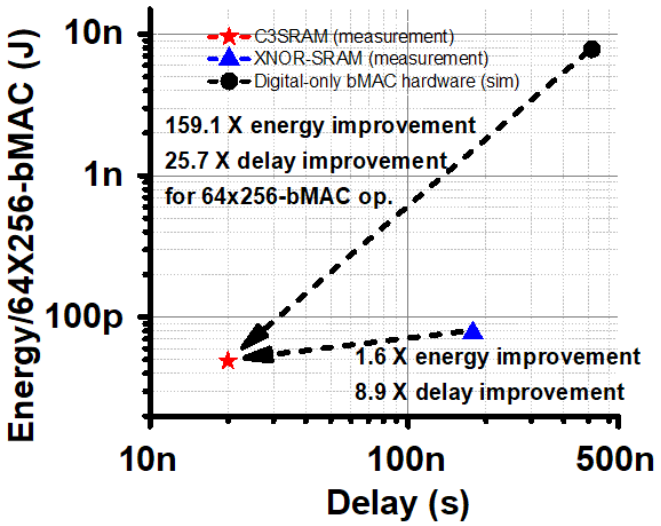


Figure 4.12: C3SRAM energy and delay comparison with XNOR-SRAM and digital ASIC with traditional SRAM and ALU.

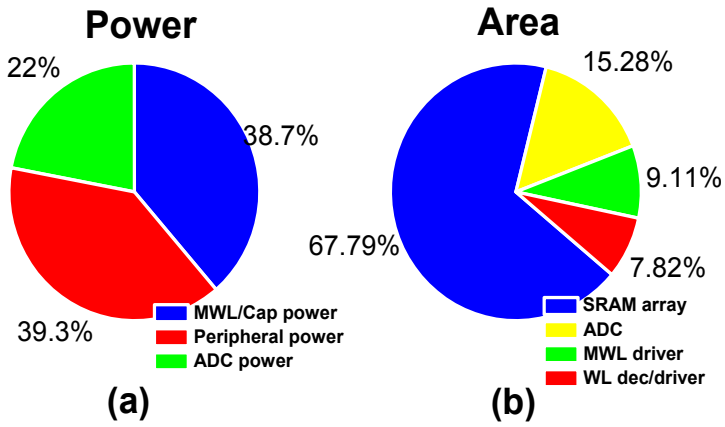


Figure 4.13: Measured power and area breakdown of the C3SRAM module.

and 14X over XNOR-SRAM (Fig. 4.12). Fig. 4.13 (a) shows the power breakdown measurements: 38.7% of the total power is consumed by driving the MWL and bit-cell capacitors, 22.0% by ADCs, and 39.3% by all other digital peripherals including MWL decoder and partial sum accumulation.

To fully benefit from the speed and power improvement of C3SRAM, striding and other dedicated input movement circuits such as those in the implementation of

Table 4.1: Comparison to Prior IMC Works.

	[28]	[97]	[31]	[32]	[36]	[29]	This work
Technology	65nm	65nm	65nm	65nm	55nm	65nm	65nm
Cell Type	10T	split-6T	10T1C	12T	Twin-8T	6T	8T1C
Power Supply	1.2V (DAC)/ 0.8V (Array)/ 1V (rest)	1V	1V	0.6-1V	1V	1V	1V (Array)/ 0.8V (Driver)/ 0.6V (ADC)
Memory Capcity	2 kB	512 B	32 kB	2 kB	480 B	16 kB	2 kB
Input Precision	6	1	1	1	1-4	8	1
Weight Precision	1	1	1	1	2-5	8	1
Output Precision	6	1	1	5	3-7	4	5[1]
Efficiency (TOPS/W)[2]	40.3	30.49-55.8	658	403	18.37-72.03	6.25	671.5
Throughput (GOPS)[3]	8	1,112.8	589.9	665	84.8-269.6	8.26	1,638

[1] Margin equivalent to 11-level mid-range of 5-bit resolution.

[2] One MAC is counted as two operations (multiplication and addition).

[3] Consider an array size of 256×64 as unit capacity.

Vesti [100] are also needed to prevent the memory macros from stalling to wait the next input arrival. Otherwise, it would impose significant overhead in activation data movement. Hence, C3SRAM is better utilized in a stand-alone module than a direct replacement of SRAM in conventional von Neumann architecture.

Table 4.1 shows the comparison of recent IMC works for neural network acceleration. C3SRAM achieves high energy efficiency and throughput. Note that some designs support higher activation precision.

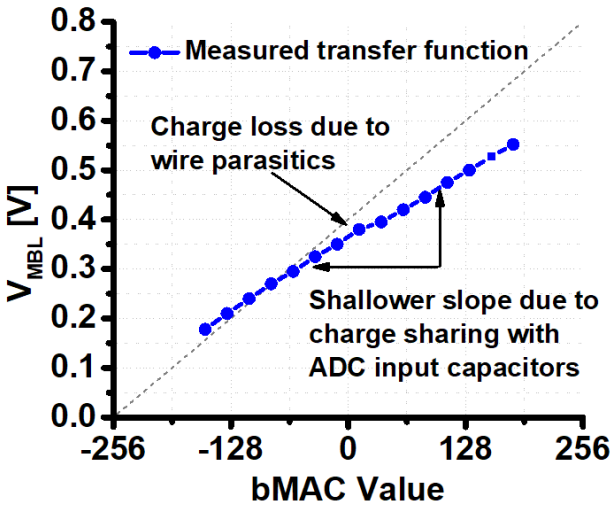


Figure 4.14: Measured VMBL transfer curve shows lower FSR from ideal curve due to charge sharing with ADC input capacitors.

4.4.2 Transfer Function

In this subsection, we measure and analyze the transfer function characteristics of C3SRAM under the same operating condition as Section V-A. The transfer function is measured according to the following steps. First we set all weights to zero, then apply known input pattern corresponding to a target bMAC output, resulting in a determinate voltage output on the MBL which we then measure. We set the off-chip flash ADC reference voltages such that we observe the result of a single comparator, i.e., set all but the first reference voltages beyond the VDR range. Then sweep the reference voltage of the comparator to find the value at which the result flips. This trip point corresponds to the bMAC output. We repeat these steps at each bMAC value to construct the transfer function. As shown in Fig. 4.14, the transfer function measurement shows good linearity and stability, while the FSR is reduced due to ADC input capacitors and MBL wire parasitics.

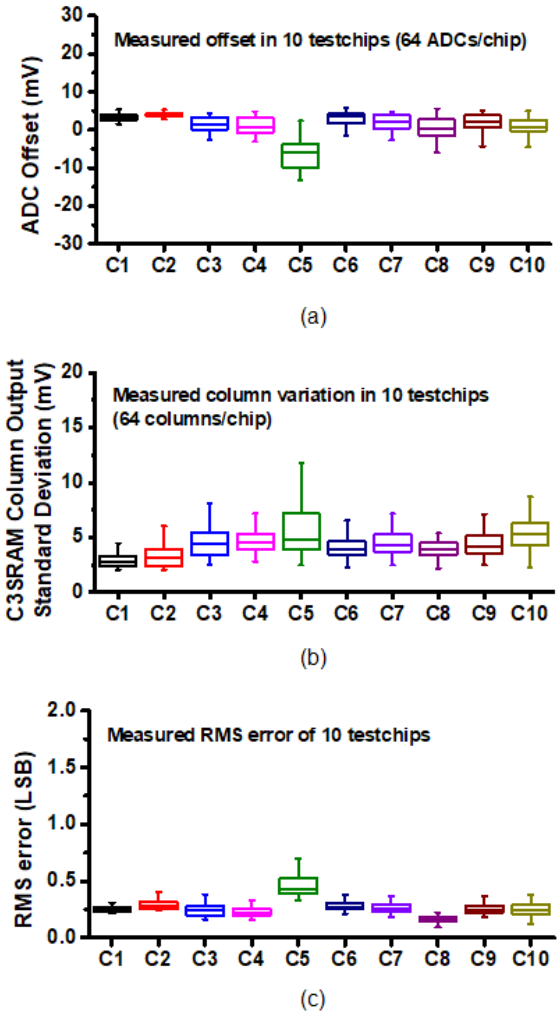


Figure 4.15: (a) ADC output offset due to comparator gain stage mismatch, (b) VMBL error variation measurement, (c) RMS error of the C3SRAM macro.

4.4.3 Variability Measurement

In this subsection, we characterize C3SRAM variabilities. The box chart in Fig. 4.15(a) shows the variation measurements of the ADC comparator offset. The data include offset statistics of 10 chips, each with 64 columns and 10 comparators per column. The variation is resultant from charge leakage, input/reference signal noise, and device mismatch. Monte Carlo simulations at TT corner shows 5mV

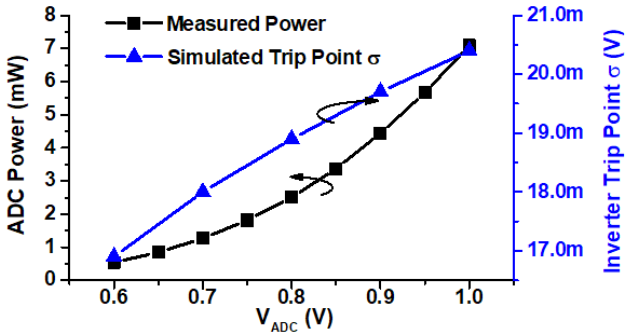


Figure 4.16: ADC power increases exponentially as ADC power supply increases; the trip point variation increases linearly.

sigma in comparator variations, consistent with measurements. The charge leakage that most significantly affects comparator output is during the capacitor input switch. As described in Section III-C, if the level of leakage or noise is greater than the delta between VMBL and reference during input capacitor switch, the comparator output can be corrupted. After the input stage, device mismatch dominates the variation. It causes trip point differential in the inverter chains. Fig. 4.16 shows that the lower the operating voltage is, the trip point variation becomes less prominent. Since the post-input offset voltage is dominated by the trip point delta between the first and second inverters divided by the gain of the first, we operate the ADC at a lower voltage of 0.6V and use long channel device for the first inverter to achieve high gain. This also helps with power reduction as shown in Fig. 4.16.

Fig. 4.15(b) shows the measured variation of bMAC operations. The main sources of variation are CC mismatch. The mismatch variation of a single CC has σ_C/C_C of 4.2% according to Monte Carlo simulation. We determine the deviation on the transfer function using propagation of uncertainty rule:

$$\sigma_{MBL} = \frac{n}{256}\sqrt{\frac{n \cdot \sigma_C^2}{n^2 \cdot {C_C}^2} + \frac{256 \cdot \sigma_C^2}{256^2 \cdot {C_C}^2}} = \frac{n \cdot \sigma_C}{256 \cdot C_C}\sqrt{\frac{1}{n} + \frac{1}{256}}. \tag{4.4}$$

The variation of VMBL from the confined region of -120 to +120 has a sigma ranging from deviation is 0.91mV to 1.77mV, based on the 600mV FSR from Fig. 4.14, consistent with Fig. 4.15(b) (which also includes intra-chip ADC offset variation).

Fig. 4.15(c) shows the RMS error of the macro performing bMAC operations. As pre-activation vary greatly in distribution variance, there is no universal input set that can characterize C3SRAM for neural networks in general. A uniform input set is used for the measurement. This result includes all non-idealities previously described, and the additional errors from unary-to-binary conversion which has no error-correction feature for low area overhead. Thus, simple bubble error can cause deviations at the final output larger than the signal variation level. This error can be mitigated with additional error correction circuitry in future works.

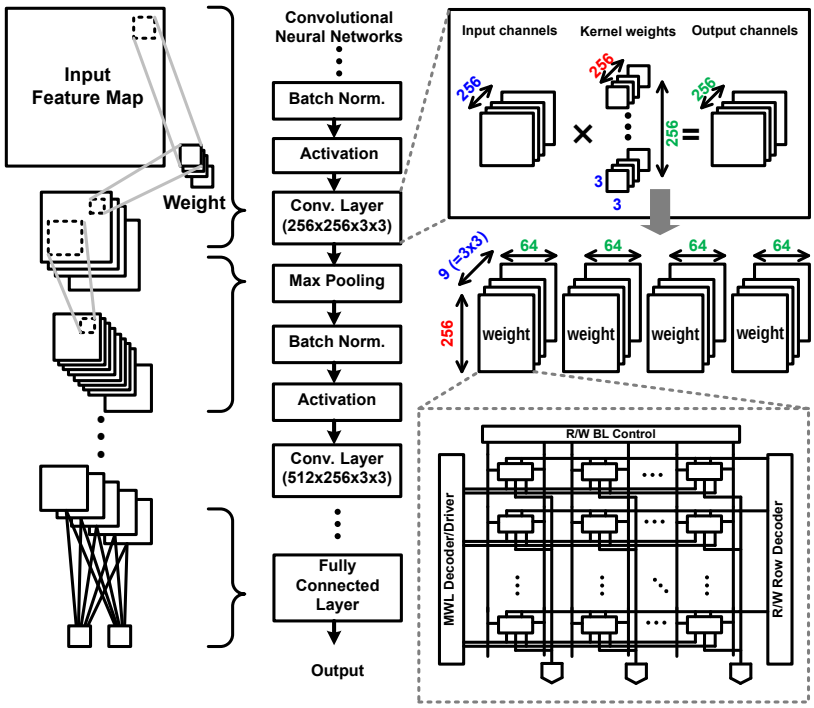


Figure 4.17: Mapping convolutional neural networks to C3SRAM-based in-memory computing.

Table 4.2: Accuracy Comparison.

	MNIST	CIFAR-10
Neural Network	MLP	VGG-like CNN
Network Topology[1]	784FC-512FC-512FC-512FC-10FC	128C3-128C3-MP2-256C3-256C3-MP2-512C3-512C3-MP2-1024FC-1024FC-10FC
Baseline Accuracy	98.7%	88.6%
Test Chip Accuracy	98.3%	85.5%

[1] nCk: $k \times k$ kernel convolution layer with n filters, mFC: m-neuron FC layer, MPp: max-pooling layer with $p \times p$ pooling size.

4.4.4 Evaluation on Neural Networks Tasks

In our evaluation for BNN accuracy, C3SRAM is responsible for the computations of convolution layers and FC layers, and all other operations of the BNN are performed in digital simulation. To evaluate the accuracy performance of C3SRAM for deep neural networks, C3SRAM computes all bMAC operations from the first hidden layer. A weight-stationary mapping scheme optimized for data reuse is adapted in our experiments. The mapping of FC layer weights in C3SRAM is as such: weights of a layer is organized column-wise, inputs/activations are applied at each row. On the other hand, convolutional layer mapping is an extension of the FC layer mapping. Mapping a 3×3×256 filter from a convolution layer is the same as mapping nine 256-neuron FC layer weights. The channels are organized in column orientation, each channel's kernel is distributed across multiple macros. The partial sums produced by ADCs are accumulated to generate the pre-activation for each neuron. The mapping

of a representative 256-channel 3×3 kernel filters is shown in Fig. 4.17.

As detailed in Table 4.2, we evaluated the inference accuracy of C3SRAM for MNIST and CIFAR-10 datasets. Max-pooling and batch-normalization are performed in the digital domain with bit precisions of 12 and 10, respectively. The accuracy for MNIST is 98.3%, against the digital baseline result of 98.7%. This accuracy result was obtained from direct measurements of the entire network. For CIFAR-10, due to test chips' limited throughput, the accuracy is evaluated from simulations based on measured error probability. We injected AMS and quantization errors in the inference of CIFAR-10 test images. At 20 runs with random seeds, the average accuracy is at 85.5%, while the digital baseline accuracy is 88.6%. This accuracy can be improved with better trained model, as researches [92, 101] have found BNN conversion with wider topology improve both robustness and accuracy. As [92] demonstrates the algorithmic advances of BNNs, the scalable mapping of C3SRAM could be used as computational primitive for larger BNNs for more complex machine learning tasks.

4.5 Conclusion

The IMC concept is developed to meet the challenge of memory bottleneck in neural network inference in conventional hardware. It can achieve high parallelism and throughput via memory cell density and achieves low power from analog computing and substantial reduction in data access. IMC faces design challenges of its own in the form of analog computing robustness issues, from process variation to system noise. Design decisions such as error-tolerant algorithms and low-variation hardware are important considerations. In this chapter, we present C3SRAM, an IMC macro for neural network acceleration. It supports noise-resistant binary neural networks, utilizes low variability components, and is scalable to map large neural networks in a modular fashion. Using robust capacitive coupling mechanism, the architecture can

reach comparable accuracy with algorithmic baseline. The 16 kb prototype in 65nm achieves the energy-efficiency of 671.5 TOPS/W and throughput of 1,638 GOPS for bMAC operations.

Chapter 5

VESTI: ENERGY-EFFICIENT XNOR-SRAM BASED IMC ACCELERATOR FOR DEEP NEURAL NETWORKS

5.1 Introduction

In Chapter 3, we demonstrated a new SRAM macro that can perform MAC along the bitline of the SRAM macro with all rows turned on simultaneously [32]. Titled as *XNOR-SRAM*, it performs XNOR-and-ACcumluate (XAC, replacing MAC in the binary-weight DNN) fast and energy-efficiently, and supports core computing primitives of state-of-the-art DNNs and CNNs, achieving highly competitive classification accuracy. In 65nm CMOS, the XNOR-SRAM chip measurements reported 235.5 pJ energy and 54.21 ns latency for completing 64 operations of 256-input XAC at 1V. A well-crafted digital hardware, which computes the same XAC but has to read out each row of weights from off-the-shelf SRAM, takes 7.81 nJ and 514 ns for 64 256-input XAC operations, which are respectively 33X higher energy and 9.5X longer latency than those of XNOR-SRAM [32]. When the convolution and MAC operations of DNNs are measured with the single XNOR-SRAM array hardware and other computations (e.g., max-pooling, batch normalization) are simulated digitally, 85.7% accuracy for CIFAR-10 dataset and 98.3% for MNIST dataset have been reported.

Although a large amount of energy reduction is demonstrated, it should be noted that XNOR-SRAM and most in-SRAM computing works that turn on all rows or columns simultaneously only demonstrate a relatively small custom SRAM array at the single-array-level [28, 30, 32]. To implement an overall DNN accelerator using in-memory computing SRAMs, there are several important challenges and missing

pieces, including integration of many in-memory computing SRAM arrays, activation storage/communication, additional digital logic for non-MAC operations, and row-by-row write energy consideration.

In this chapter, we substantially expanded the single-array-level prior XNOR-SRAM design towards a configurable DNN accelerator architecture that integrates 72 XNOR-SRAM arrays with inter-array communication, and supports a wide range of DNN/CNN algorithms with configurable activation precision. The proposed accelerator supports 3×3 and 1×1 convolutional kernels and up to 256 feature maps in a convolutional layer.

The main contributions of this work are as follows:

- We construct the chip-level DNN/CNN accelerator architecture that employs many instances of in-memory computing SRAM (e.g., XNOR-SRAM) macros with a methodology to efficiently load/map weights onto such XNOR-SRAM arrays for convolution layers and fully-connected layers of DNNs.

- By re-using the XNOR-SRAM macro originally designed for binary activations and weights, we add peripheral digital logic that can support multi-bit activations, which becomes an effective knob to favorably trade-off energy versus accuracy.

- We employ double-buffering technique with two groups of in-memory computing SRAMs, which effectively hides the latencies of re-programming in-memory computing SRAM arrays with new DNN weights.

- We evaluate the chip-level energy benefits and remaining bottlenecks of in-memory computing based DNN accelerators.

In this chapter, we are focusing on SRAM as the memory substrate of in-memory

computing. On the other hand, researchers have also presented in-memory computing designs based on emerging non-volatile memory technologies, such as Phase Change Memory (PCM) [102, 103], Resistive RAM (RRAM) [96, 104], and Magnetic RAM (MRAM) [105,106]. As compared to SRAM, these technology promises a bitcell that can store multi-bit weight, often in the form of analog variables (e.g., resistance), in a smaller footprint. This is a significant benefit to supporting DNN and CNN algorithms using multi-bit weights. However, these emerging memory devices exhibit significant challenges including high variability and non-linearity [107,108], and most importantly the manufacturability has not reached that of CMOS technologies, which severely limits system-level integration with many large arrays.

The remainder of the chapter is organized as follows. Sec. 5.2 discusses practical design challenges when we design a chip-level DNN accelerator using many in-memory computing macros such as XNOR-SRAM. In Sec. 5.3, we describe the microarchitecture of the proposed accelerator, optimal precision study and the methodology to efficiently map optimized DNNs/CNNs onto XNOR-SRAM arrays. Sec. 5.4 reports experimental results on speed, energy dissipation, and classification accuracy across several workloads. Finally, this chapter is concluded in Sec. 5.5.

5.2 Practical Challenges of In-memory Computing based Accelerators

Although XNOR-SRAM as well as other in-memory computing SRAMs show promising energy-efficiency at the single-array-level, there are several important challenges and missing pieces towards building a chip-level DNN accelerator using these arrays.

Integration of many in-memory computing SRAM arrays: First, it should be noted that XNOR-SRAM as well as most of the in-memory computing hardware only demonstrated a relatively small custom SRAM array (around a few hundred

kb or less). Therefore, to implement an overall DNN accelerator, many of these in-memory computing SRAM arrays need to be employed and integrated together with on-chip communication networks.

ADC overhead and offset cancellation: Second, the ADC design incurs area and power overhead, which adversely affects the array efficiency and energy-efficiency. In addition, ADC performance is sensitive to offset or variability, and for DNN applications, evaluation should be made on how much DNN accuracy degardation occurs due to the variability. Ideally the ADC offsets should be calibrated out using offset compensation circuits [109], but such calibration circuits will increase the area/power further.

Post-processing modules: Third, towards evaluating accuracy for CNNs or DNNs, these designs employ a considerable amount of post-processing modules such as partial sum accumulation, batch normalization, pooling, non-linear activation, etc. These post-processing modules add system-level design complexity and energy consumption beyond those of the single in-memory computing SRAM array.

Activation storage and communication: Fourth, typically the in-memory computing SRAM arrays store the DNN weights, while the activations are applied as inputs to the custom SRAM array either at the wordlines or bitlines. Different activation values need to be applied to the SRAM array every cycle, which means that the activations need to be stored in a separate memory (e.g., another SRAM array) and communicated to the in-memory computing SRAM array at the right time. However, the energy of activation storage and communication are typically not included in the in-memory computing SRAM macro energy values.

Write energy: Finally, since the state-of-the-art DNNs can be very large, we will not be able to store the entire weights of DNNs in in-memory computing SRAMs without consuming a huge amount of area and static power. To that end, it will

be necessary to reload different weights in-memory computing SRAM arrays at different times. While XAC operation for in-memory computing SRAMs can be fully parallelized by turning on all the rows, write operation of loading new weights to the SRAM still requires row-by-row operation, which incurs long latency and consumes write energy. Typically, the reported in-memory computing SRAM macros do not include any write energy, but for a system-level accelerator, the corresponding write energy of in-memory computing SRAM macros should be characterized and included.

In the subsequent sections, we will describe how a number of these key challenges have been addressed and inclusively implemented in the proposed accelerator design, and will report the accelerator evaluation results.

5.3 Vesti Accelerator Design

5.3.1 Microarchitecture Overview

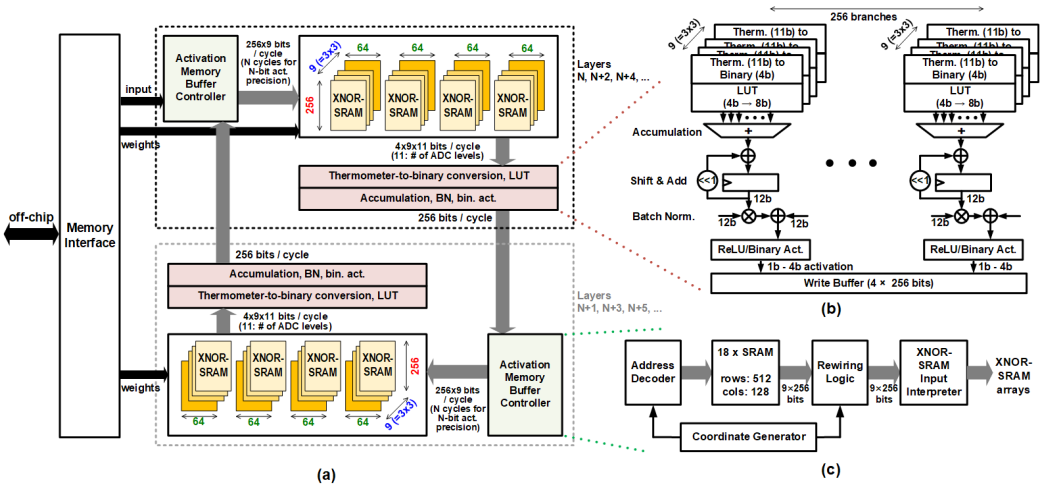


Figure 5.1: (a) Overall microarchitecture of proposed in-memory computing Vesti accelerator. (b) Computations for the thermometer-to-binary conversion, LUT, batch normalization, etc. (c) Block diagram of the activation memory buffer.

We designed the microarchitecture of the Vesti accelerator, which integrates the

aformentioned key missing pieces, and can execute inference for a wide range of DNNs/CNNs. Fig. 5.1(a) shows the overall microarchitecture that computes a deep CNN inference on a layer-by-layer basis. Employing the double-buffering scheme [110]), it consists of two symmetric cores, one of which performs the in-memory computation, while the other can load new weights from an on-chip global buffer or an off-chip DRAM. Each block consists of (1) the ensemble of the XNOR-SRAM macros that perform XAC from convolution and fully-connected layers, and (2) a digital ALU that performs other operations such as batch normalization, activation, and max-pooling. In this work, 36×2 XNOR-SRAM macros are employed (Fig. 5.1) to support representative CNNs for CIFAR-10 dataset [35]. To support larger CNNs, the XNOR-SRAM macros can be time-multiplexed and re-used over time. At 1V, the XNOR-SRAM macro can operate at 1.2 GHz and the digital ALU was synthesized and placed/routed at 0.55 GHz in the same 65nm CMOS technology.

The inputs and weights are fetched from off-chip DRAM. The inputs are saved in the activation memory buffer and weights are written into the XNOR-SRAM macros. The 36 XNOR-SRAM macros in each core are divided in 4 groups. The 4 groups share the input activations, performing XAC operations for up to 256 (=64×4) output feature maps in parallel. The 9 XNOR-SRAM macros in each group can accept inputs from up to 256 input feature maps when performing 3×3 convolution and up to 2304 (=256×9) inputs when performing fully-connected matrix vector multiplication. In each cycle, up to 36 256-input XAC operations can be executed in parallel. The outputs of 36 XNOR-SRAM macros are processed by a 256-way digital ALU where ADC output decoding, partial sum accumulation, max-pooling, batch normalization and binary/ReLu activation are performed. The results will be saved back to the activation memory buffer of the other core, which will perform computation for the ensuing layer in a similar fashion.

Double buffering technique [110] is employed to hide the weight loading latency. While one core is performing computation for one layer, the other core loads the weights for next layer. For example, for two adjacent convolution layers, of which the channel size and map size is 256 and 16×16, respectively, the 36 XNOR-SRAM macros in one core complete the convolution in 256 cycles; at the same time, the 36 XNOR-SRAM macros in the other core load the weights for next layer in 256 cycles (row by row). Fig. 5.3 shows the timing diagram that illustrates these operations for two adjacent layers of a CNN. This layer-by-layer operation will continue to perform all the layers of a given DNN/CNN.

5.3.2 Multi-bit Activation Support

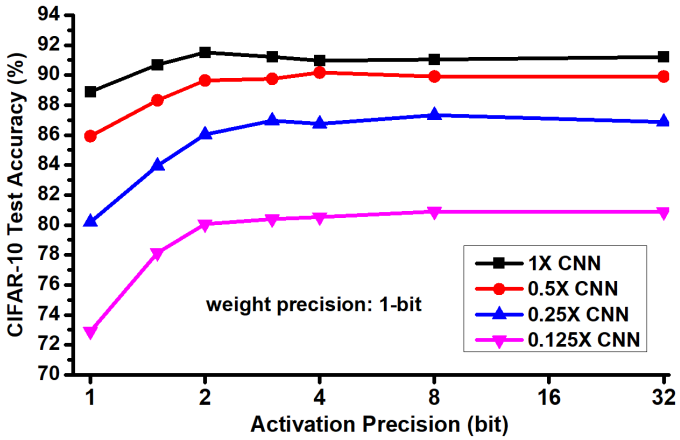


Figure 5.2: Classification accuracy for CIFAR-10 dataset is shown across different activation precision values for four different DNN sizes. For all data points, the weight precision is binary (only two values of +1 or -1).

While the XNOR-SRAM macro has native support on binary activations and binary weights, it should be noted that DNNs with binary activations and binary weights do not yet reach the same accuracy level of their higher precision counterparts [16, 35]. Therefore, in our proposed Vesti accelerator, we support weights with

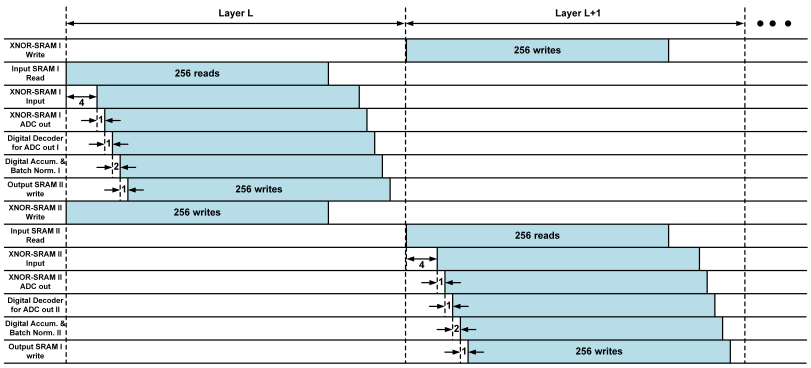


Figure 5.3: Timing diagram of the accelerator operation for two adjacent layers of a CNN, including in-memory computing, double-buffering, and peripheral computations.

binary precision (+1 or -1) but activations with configurable precision from 1-bit to 4-bit. This scheme will not only minimize the weight memory footprint, but can also reach the level of DNN accuracies with floating-point precision [91].

Our choice on binary weight and multi-bit activation is based on the algorithm level experiments that we conducted. In particular, we swept (1) several CNN sizes (various numbers of feature maps per layer) for the CIFAR-10 dataset, and (2) activation precision values including binary, ternary, 2-bit, 4-bit, 8-bit, and 32-bit. The results are shown in Fig. 5.2. 1X CNN represents the network of input-128C3-128C3-MP2-256C3-256C3-MP2-512C3-512C3-MP2-1024FC-1024FC-10FC, which was presented in [35]. Here, 128C3-128C3 refers to the convolution layer with 128 input feature maps, 3×3 kernels and 128 output feature maps, MP2 refers to 2×2 max-pooling, and 1024FC refers to the fully-connected layer with 1024 hidden neurons. 0.5X CNN represents the network input-64C3-64C3-MP2-128C3-128C3-MP2-256C3-256C3-MP2-512FC-512FC-10FC, where the number of feature maps in all convolution layers is reduced by half compared to those in the 1X CNN, and the number of hidden neurons in the fully connected layers is reduced by half as well. Similarly, 0.25X CNN

and 0.125X CNN represents the networks where all dimensions are reduced by 4X and 8X, respectively, compared to the 1X CNN.

It can be seen in Fig. 5.2 that the accuracy of models using floating-point activation precision could be reached by employing 3-/4-bit activation precision with binary weights. However, employing binary activations do show considerable degradation in CNN accuracy. To that end, we use in-memory computing hardware with fixed binary weights, but employ configurable precision for the inputs and neuron activations at the periphery of the XNOR-SRAM array.

In the microarchitecture level, the support of the multi-bit inputs is done by performing XAC operation for each bit of input/activation using XNOR-SRAM macros and then shift-and-accumulate the bitwise XAC results in the digital peripheries over multiple cycles (e.g., N cycles for N-bit precision of activations). This is illustrated in Fig. 5.1(b), together with other digital computations at the periphery. Configurable precision (from 1-bit to 4-bit) for activations can be flexibly supported at the cost of additional clock cycles.

5.3.3 Activation Memory

Activation memory in each core is employed to store the input feature maps for that core and to save the output activations from the other core. Take a 256C3-256C3 convolution layer as an example. This convolution layer consists of 256 input feature maps, each of 32×32 pixels. The input feature maps stored in an activation memory need to be read out to perform convolution. 256×3×3 pixels should be fetched out and presented to the 36 XNOR-SRAM macros at every cycle. A relatively large FIFO-based buffer between the activation memory and XNOR-SRAM array can reduce activation memory access by exploiting convolutional data reuse. However, this will result in considerable power consumption in the buffer and significant area for the

control logic.

To get rid of the buffer, we propose a new way to store the feature maps in the activation memory. In particular, we divide the overall activation memory in 9 activation SRAM blocks of 128 rows and 256 columns as shown in Fig. 5.4. The input feature maps are divided in 3×3 tiles. The input feature maps pixels are grouped and stored in the 9 SRAM blocks according to their position in the 3×3 tiles they belong to. By storing pixels in this fashion, we can read all the 256×3×3 pixels in a single cycle given the fact that any 3×3 patch of input feature maps is now stored in 9 different SRAM blocks. A controller block is designed to generate corresponding addresses for the 9 SRAM blocks.

As shown in Fig. 5.1(c), the activation memory buffer has four parts: (1) coordinate generator, (2) address decoder, (3) rewiring logic, and (4) XNOR-SRAM input interpreter module, along with the 18 SRAM blocks. The functionalities of these blocks are described in more detail below.

Coordinate Generator: The coordinate generator block generates the x and y coordinates of the output map pixels sequentially in a row-major order. In case that the convolutional layer is followed by a max-pooling layer, the coordinates correspond to each pooling window will be generated in a row-major order inside each pooling window to ease the buffer size requirement for max-pooling operation. These coordinates will serve as inputs to our address decoder and rewiring logic blocks to different combinations of row addresses and read enable signals for SRAM blocks.

Address Decoder: The address decoder block generates activation memory addresses for the 9 SRAM blocks according to the output feature map coordinates. Zero padding can be supported smoothly by the address decoder as well. When the generated addresses are found invalid for the input feature maps, corresponding read enable signals will be inactive and substitute the SRAM output with zero values.

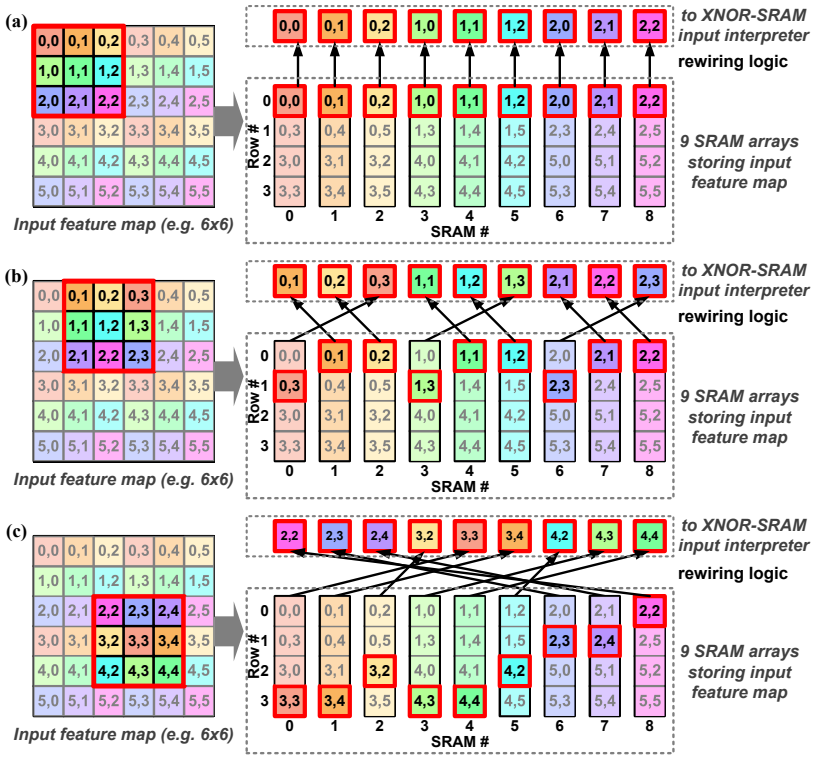


Figure 5.4: Illustration of convolution layer feature map storage and access scheme in 9 separate SRAM arrays. (a) 3×3 window starting from (0, 0); (b) 3×3 window starting from (0, 1); (c) 3×3 window starting from (2, 2).

Since the feature map size and channel size vary from layer to layer, we further divide each 256-bit-word-length SRAM block in two 128-bit-word-length SRAM blocks. For some layers (e.g., map size = 32×32, channel size = 128), we concatenate these two in depth direction to form a deeper SRAM block with 128-bit word length; for some layers (e.g., map size = 16×16, channel size = 256), we concatenate these two in word direction to form a wider SRAM block with 256-bit word length.

Rewiring logic: As shown in Fig. 5.4, although we can always fetch any 3×3 patch in input feature maps from 9 different SRAM blocks at the same time, the order of the 9 SRAM outputs do not always align with the row-major order in each 3×3 patch. We need to rewire the SRAM outputs to make sure the 3×3 patches be

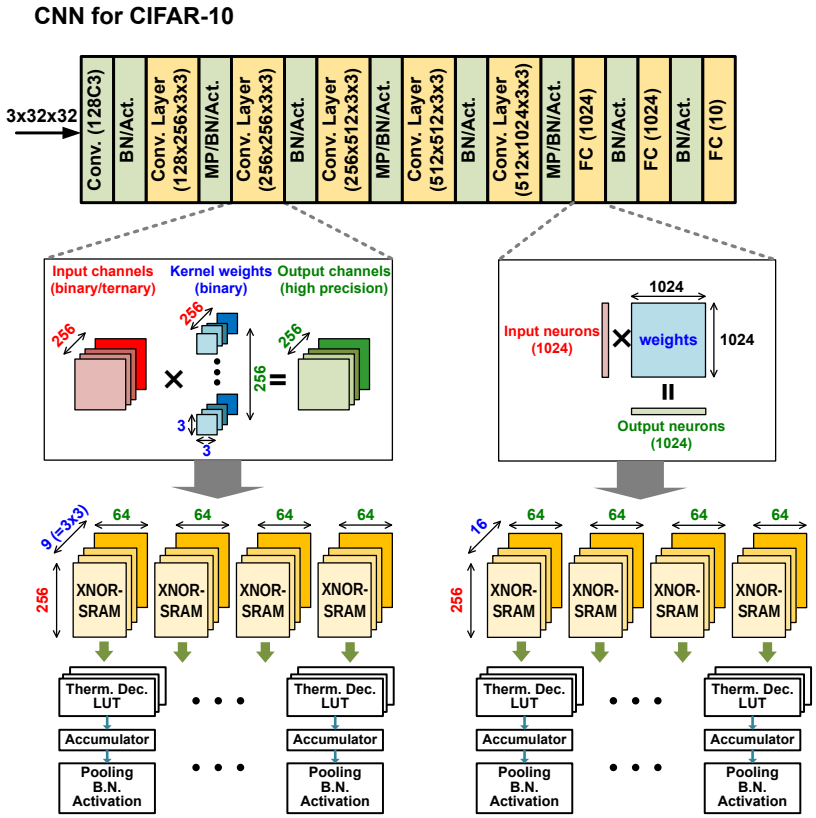


Figure 5.5: Mapping convolution layers (left) and fully-connected layers (right) of deep CNNs onto the Vesti accelerator.

presented to the XNOR-SRAM macros in correct order. There are 9 different rewiring patterns in total, depending on the 3×3 patch row and column offset remainder modulo by 3. Fig. 5.4 illustrates 3 different patterns with an example of 6×6 feature maps, where the patch row and column offset is (0, 0), (0, 1) and (2, 2), respectively.

XNOR-SRAM input interpreter: Depending on whether we operate the XNOR-SRAM in 1-bit binary activation (+1/-1) or multi-bit binary activation (+1/0) mode, the XNOR-SRAM input interpreter will generate proper wordline inputs for XNOR-SRAM array from what it receives from the rewiring logic.

5.3.4 Mapping of Convolution, Fully Connected, and Other Layers

For convolution layers, we propose a mapping scheme where the same location pixel (x, y) of the kernel from all the kernels for different input/output feature maps will be stored in the same XNOR-SRAM array. Other location pixels of the kernels will be stored in different XNOR-SRAM arrays. This is illustrated in Fig. 5.5 (left). Then, the XAC or bitcount accumulation results will be gathered from these multiple XNOR-SRAM arrays and accumulated together, to obtain the final output activation result. This scheme enables extensive re-use of the activations, with weights being stationary at the XNOR-SRAM arrays.

Using the weight-stationary scheme in the XNOR-SRAM macros, it is straightforward to map fully connected layers of DNNs, where neurons/activations are in vectors and weights are in matrices. This nicely maps to the row drivers for activations and weights stored in the SRAM. For the fully-connected layers whose size is larger than 256×64, we break the large weight matrix into a number of small sub-matrices and accumulate the matrix-vector multiplication results accordingly. This is illustrated in Fig. 5.5 (right).

We implement other computation modules such as max-pooling, batch normalization, and non-linear activation with all-digital circuits, which will reside at the periphery of XNOR-SRAM macros. The computing sequences of these modules are as follows. Once the XAC operations are done inside the XNOR-SRAM array, they are digitized with the flash ADC and outputs a thermometer code. The thermometer code from each column gets converted to a binary number and a simple look-up table (LUT) is used to bring it back to the exact bitcount value, according to the non-linear quantization scheme that was employed with regards to the ADC reference voltages. Then, the same columns from different XNOR-SRAM arrays are accumulated (to

compute the summation of the partial sums of all convolution pixels), which then sends out the final output sum value. Using the trained batch normalization parameters, this final sum value goes through batch normalization and non-linear quantization (thresholding for binary/ternary activations, ReLU for multi-bit activations). Finally when the current layer's computation is completed, the activation outputs are ready to serve as the input activations for the next layer of the DNN/CNN.

5.4 Experimental Results

5.4.1 Experiment Setup

The Vesti accelerator consists of two main parts: (1) multiple instances of XNOR-SRAM arrays (including ADC peripheries), and (2) activation SRAMs and additional digital logic/control. As reported in [32], the XNOR-SRAM array itself consists of a custom SRAM array and peripheral mixed-signal circuits, which are designed and laid out manually. The remaining modules including activation SRAMs and additional digital logic/control are implemented through the standard cell based design flow. Activation SRAMs are off-the-shelf 6T SRAMs, and are obtained from industrial memory compiler. Digital logic/control modules are implemented in RTL, synthesized using Synopsys Design Compiler, and placed and routed using Cadence Innovus tool in the same 65nm CMOS technology.

We characterized the power/energy consumption, throughput and accuracy (for MNIST and CIFAR-10 datasets) of the Vesti accelerator. For all the MAC or XAC operations in the convolution and fully-connected layers, the XNOR-SRAM testchip measurement results from [32] are used. For all other digital logic and off-the-shelf SRAMs, the power consumption results are obtained from Synopsys PrimeTime simulation of the post-layout netlist with RC parasitics and actual data switching activity

information.

We considered MLPs and CNNs for image classification tasks for MNIST and CIFAR-10 datasets, respectively. For MNIST, a MLP with three hidden layers, each with 256 neurons, is used. The CIFAR-10 CNN architecture used in this section is adopted from the CNN reported in [35], consisting of six convolution layers and three fully-connected layers, which is identical to the 1X CNN that was used in Section 5.3.2 except that the last two convolution layers have 256 feature maps. This represents the network of input-128C3-128C3-MP2-256C3-256C3-MP2-256C3-256C3-MP2-1024FC-1024FC-10FC.

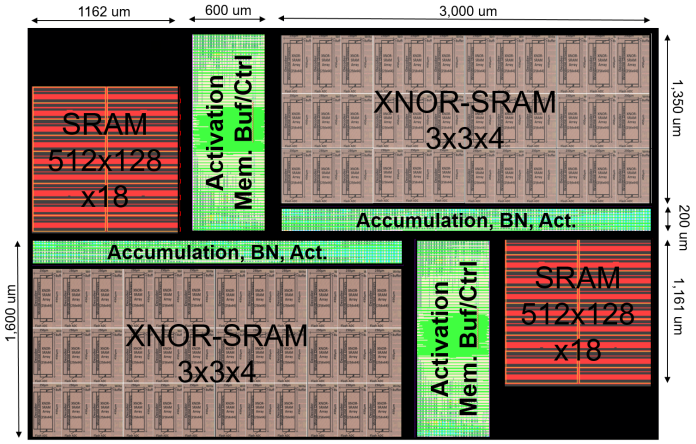


Figure 5.6: Including the XNOR-SRAM prototype chip layout, the layout of activation memory buffer/controller, accumulation and batch normalization modules are shown.

The ADC employed in [32] was a 11-level flash ADC, which non-linearly quantized the analog bitline voltage to produce approximate partial XAC values. Since the distribution of partial XAC values was found to be concentrated around zero, finer-grain quantization has been employed around zero XAC values in [32]. The effect of noise/offset for the ADC, as well as process variation of XNOR-SRAM bitcells, contribute to the 3-σ spread of the analog read bitline voltage for same XAC values

in Fig. 3.13.

To evaluate the accuracy of binary-weight multi-bit-activation MLPs and CNNs on Vesti, we first obtained a probabilistic model for the XNOR-SRAM XAC and quantization operations from XNOR-SRAM chip measurements. Specifically, we used a total of 656k (513 XAC values×64 columns×20 samples/XAC/column) random test vectors and measured the outputs of the XNOR-SRAM to build XNOR-SRAM's probabilistic model as a function of XAC value. We simulated BNN model accuracy on Vesti by running software simulations where we stochastically quantized all the 256-input XAC partial sums according to the measured probabilistic model. Offset calibration or other techniques that lower the ADC noise/offset can result in tighter distribution of the probabilistic model, which will result in better DNN accuracy. Note that the probabilistic model was characterized based on single-array measurement. Array-to-array variation could potentially result in further accuracy loss, while more accurate array-by-array ADC calibration could alleviate the loss. In addition, hardware-variation-aware network retraining algorithm [111] could potentially help minimize the accuracy loss due to the array-to-array variation.

5.4.2 Area, Energy, Throughput, and Accuracy

In Fig. 5.6, the placed-and-routed layout of all digital peripheral blocks as well as the 72 XNOR-SRAM arrays are shown. The total area of the Vesti accelerator is 15 mm^2 in the 65nm CMOS process. The width and height of different modules that comprise the accelerator are shown as well. Multiple XNOR-SRAM arrays consume 54% of the total area, the activation SRAMs consume 18%, and remainder of the area (28%) is occupied by digital logic and control modules.

For MNIST MLP, we simulated and evaluated the total MLP energy for various activation precisions of 1-3 bits in Fig. 5.7. The XNOR-SRAM macro energy is

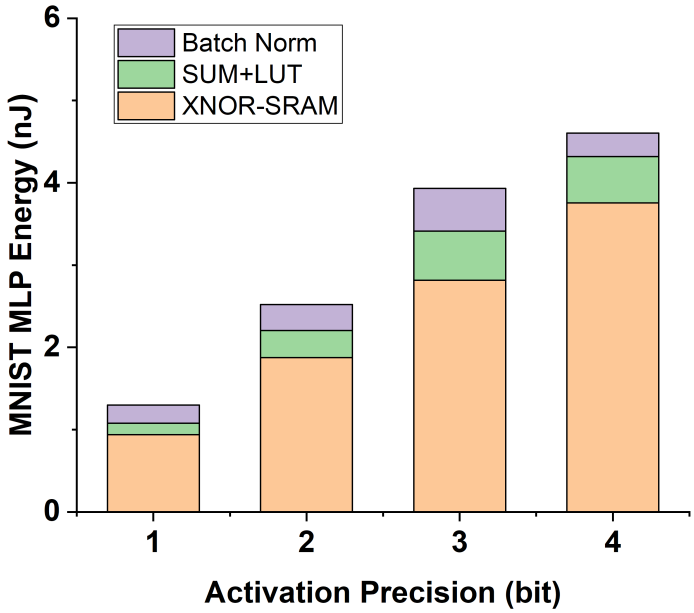


Figure 5.7: Energy breakdown of the entire MLP designed for MNIST dataset.

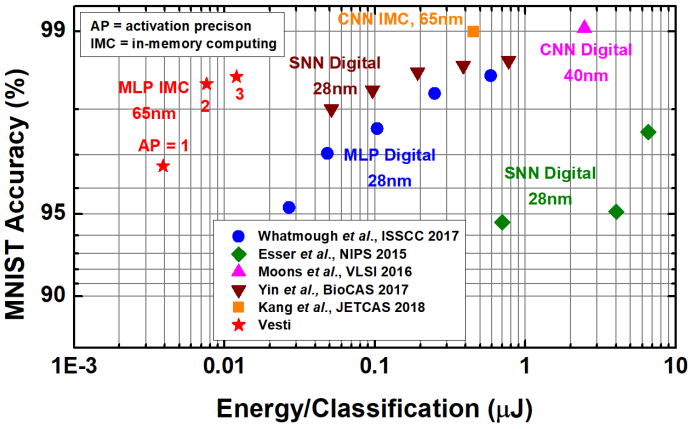


Figure 5.8: For MNIST dataset, accuracy vs. energy (per classification) comparison with prior works for MNIST dataset classification.

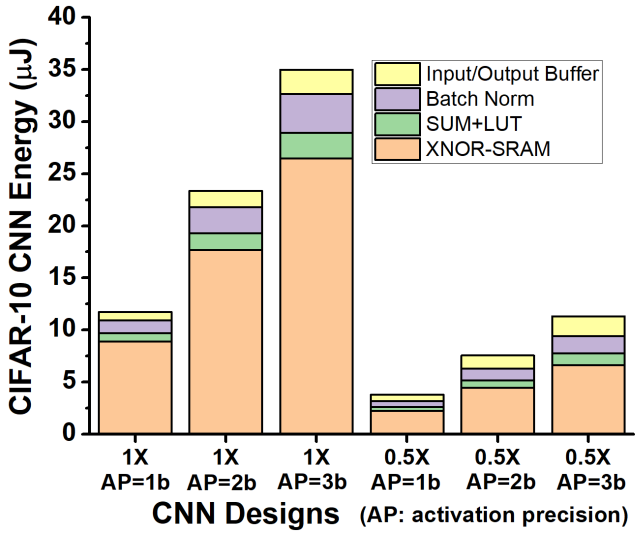


Figure 5.9: Energy breakdown of the entire CNN designed for CIFAR-10 dataset. Two different sizes of CNNs (1X, 0.5X) and three different activation precision schemes (1-bit to 3-bit) are shown.

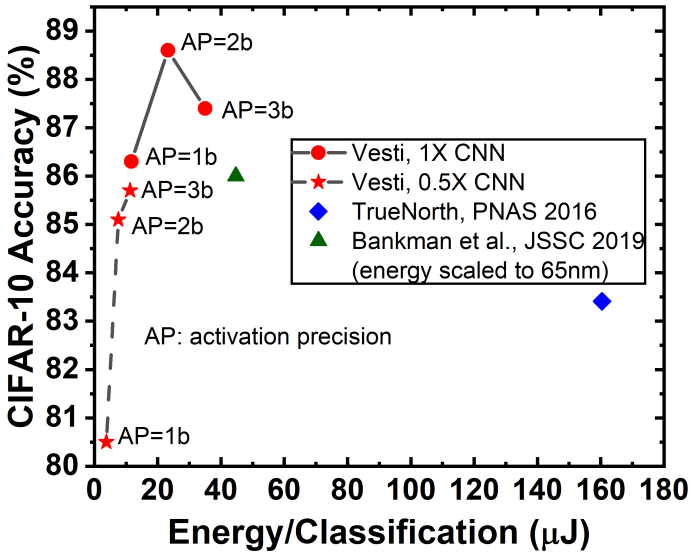


Figure 5.10: Accuracy vs. energy comparison for CNNs for CIFAR-10 with variable activation precision with prior works.

acquired from chip measurements [32], and energy for other digital components is obtained from post-layout simulation with data switching activity. Since activations with N-bit precision consume N cycles to compute using the XNOR-SRAM array, the overall energy roughly increases linearly with the activation precision. To perform a single inference of the MLP using the 1-bit activation precision, the Vesti accelerator consumes 21 cycles. At the 0.55-GHz clock frequency, which the Vesti accelerator can operate at, the throughput of 26M inferences per second is achieved. Fig. 5.7 shows the energy breakdown across various activation precision.

For the MNIST dataset, in Fig. 5.8, we compared Vesti (with 1-/2-/3-bit activation precision) to several prior works [20, 98, 112–114] that demonstrated the energy and accuracy for the entire DNN for MNIST dataset. For all data points of prior works shown in Fig. 5.8, we directly grabbed accuracy/energy numbers for MNIST classification from [20, 98, 112–114] without adapting them. 'SNN' stands for spiking neural network implementations [112, 114], and 'IMC' represents SRAM-based in-memory computing works [98]. It can be seen that the proposed Vesti accelerator results in superior accuracy versus energy trade-offs compared to the state-of-the-art DNN implementations (MLP, CNN, SNN, and IMC) for MNIST dataset.

PROMISE [33], another mixed-signal in-memory computing accelerator, achieved 0.49 μJ/classification for the MNIST dataset using the MLP of 784-512-256-128-10 with 8-bit precision, but did not report the classification accuracy (due to the lack of accuracy information, this work was not included in Fig. 5.8). Vesti achieves 0.012 μJ/classification for the MLP of 784-256-256-256-10 with 1-bit weight and 3-bit activation precision, which resulted in 98.5% classification accuracy. Considering that similar MLP networks were employed in both works, Vesti achieves $\sim$40X energy improvement.

For CIFAR-10 CNN, we also simulated and evaluated the total CNN energy for

Table 5.1: Comparison with Prior Works.

	[115]	[23]	Vesti	[20]	[112]	[113]	[114]	[98]
Technology (nm)	45	28	65	28	45	40	28	65
Model	SNN	CNN	CNN/ MLP	MLP	SNN	CNN	SNN	CNN
Power Supply (V)	0.6-0.8	0.6-0.8	1	0.6-1.1	0.6-0.8	0.55-1.1	0.9	1
Clock Frequency (MHz)	–	10	550	667	–	204	163	–
CIFAR10 Accuracy (%)	83.41	86.05	88.6	–	–	–	–	–
CIFAR10 Throughput (FPS)	1249	237	328k	–	–	–	–	–
CIFAR10 Energy/ Prediction (μJ)	163	3.8	23.3	–	–	–	–	–
MNIST Accuracy (%)	–	–	98.5	98.5	99.42	99	98.7	99
MNIST Throughput (FPS)	–	–	8.6M	–	1k	13.4k	91.6k	70
MNIST Energy/ Prediction (μJ)	–	–	0.012	0.588	108	0.45	0.773	0.45

various activation precisions of 1-3 bits for two CNN sizes (1X and 0.5X, as described in Section 5.3.2) in Fig. 5.9. Since activations with N-bit precision consume N cycles to compute using the XNOR-SRAM array, the overall energy roughly increases linearly with the activation precision. To perform the single inference of the CIFAR-10 CNN using the 1-bit activation precision, the Vesti accelerator consumes 1,676 cycles. At 0.55-GHz clock frequency, which the Vesti accelerator can operate at, this marks the throughput of 328K inferences per second. Fig. 5.10 shows the energy across three activation precision values (from 1-bit to 3-bit) and two CNN sizes (1X and 0.5X CNN). For the CIFAR-10 dataset, in Fig. 5.10, we show the energy and accuracy trade-offs for variable activation precision using Vesti accelerator, and also

compared to the energy and accuracy of the all-digital TrueNorth processor reported in [115] and the mixed-signal binary CNN processor reported in [23] (energy scaled to 65nm). Regarding the comparison with [23], we used an energy scaling factor of 11.77X from 28nm to 65nm, based on the dynamic energy consumption of two commercial register file designs (array size of 256×64) at 28nm (0.81 V supply) and 65nm (1.0 V supply). As shown in Fig. 5.10, Vesti accelerator results in superior accuracy and energy trade-offs for two different CNN sizes (1X and 0.5X) for multiple activation precision schemes for the CIFAR-10 dataset. Comparison with prior works is summarized in Table 5.1.

5.4.3 Discussions

For binary-weight CNNs, as shown in Fig. 5.2, 3-bit and higher activation precision schemes have achieved similar accuracy in software simulations. With finer-grain activation representation, 3-bit (or higher) activation hardware designs seemingly become more sensitive to similar amount of quantization error to signal ratio, introduced by 11-level ADCs, compared to coarser-grain 1-/2-bit activations. Given similar amount of perturbation in XAC values, higher-precision-activation hardware are experiencing more degradation in accuracy, especially for larger networks such as the 1X CNN in Fig. 5.10. To achieve further accuracy improvement for 3-bit and higher activation precision schemes, we postulate that ADCs with more levels (higher precision) will be necessary.

In addition, the degraded accuracies for 3-bit activation scheme of 1X CNN in Fig. 5.10 seem to be partially due to the fact that ReLU activation generates a relatively large portion of zero activations for 3-bit or higher activation precisions, which leads to overall small accumulation values and concentrates the bitcount values near zero, in turn becoming more susceptible to variability. Reducing the variability effect

is a crucial concern for in-memory computing designs involving analog computing.

With substantial reduction in on-chip DNN energy, off-chip DRAM access could be a larger portion of the chip-level energy. This requires an in-depth analysis and potentially need to exploit some of the recent DRAM energy improvement techniques such as [116].

5.5 Conclusion

In-memory computing for DNNs and CNNs has been recently gaining significant attention. This is because memory access is the main bottleneck to scaling delay and energy dissipation of the MAC operation in digital DNN/CNN accelerators. Most in-SRAM computing works that turn on all rows or columns simultaneously [28, 30, 32] have only implemented a single custom SRAM array, which only computes a small portion of total MAC operations in DNNs. In such prior works, the rest of the operations and overall architecture to implement the DNN accelerator have not been shown or implemented in hardware.

In this chapter, we substantially expanded the single-array-level prior XNOR-SRAM work [32] towards a configurable DNN accelerator architecture that integrates 72 XNOR-SRAM arrays. The proposed Vesti architecture features (1) methodologies to efficiently load/map weights onto such XNOR-SRAM arrays for convolutional layers and fully-connected layers of DNNs, (2) multi-bit activation memory storage and control, (3) double-buffering technique to hide the latencies of re-programming in-memory computing SRAM arrays, and (4) inter-array communication.

Due to these comprehensive designs, Vesti simultaneously achieves both high accuracy and low energy for representative DNNs that are benchmarked for MNIST and CIFAR-10 datasets. The Vesti accelerator presented in this work feature essential techniques for in-SRAM computing based deep learning processors, which can fit

under the stringent power/energy envelopes of mobile, wearable, and IoT devices.

Chapter 6

PIMCA: A PROGRAMMABLE IMC ACCELERATOR FOR ON-DEVICE DEEP NEURAL NETWORK INFERENCE

6.1 Introduction

In the era of artificial intelligence (AI), various deep neural networks (DNN) have achieved human-level performance in many recognition tasks. These DNNs usually require billions of multiply-and-accumulate (MAC) operations, soliciting energy-efficient architecture for on-device DNN inference. In-memory computing (IMC) has emerged as a promising technique owing to high parallelism, reduced data communication, and energy-efficient analog MAC computation. Single-macro-level [117, 118] or layer-/system-level [99, 119–121] IMC designs have been demonstrated with high energy-efficiency. However, a limited number of IMC macros were integrated on-chip [119,120], or the dataflow of IMC/non-IMC operations were hard-wired [99,121], exhibiting limited flexibility to support layer types other than batch normalization (BN) and activation layers. Furthermore, hardware loop support is often omitted [99, 119–121], incurring large overhead in instruction counts.

In this chapter, we present a programmable IMC accelerator (PIMCA) in 28nm CMOS integrating 3.4-Mb capacitive IMC SRAM macros, demonstrating one of the largest integrations for IMC design to date. Also, as part of the instruction set architecture (ISA), a flexible single-instruction-multiple-data (SIMD) processor is implemented to support a range of non-MAC vector operations, such as average-/max-pooling, residual layers, etc. The ISA also features hardware loop support, reducing instruction count and latency. Leveraging recent low-precision DNN algorithms [122],

PIMCA supports DNNs with 1-b and 2-b precision.

6.2 Design and Optimizations of PIMCA

6.2.1 Overall Architecture of PIMCA

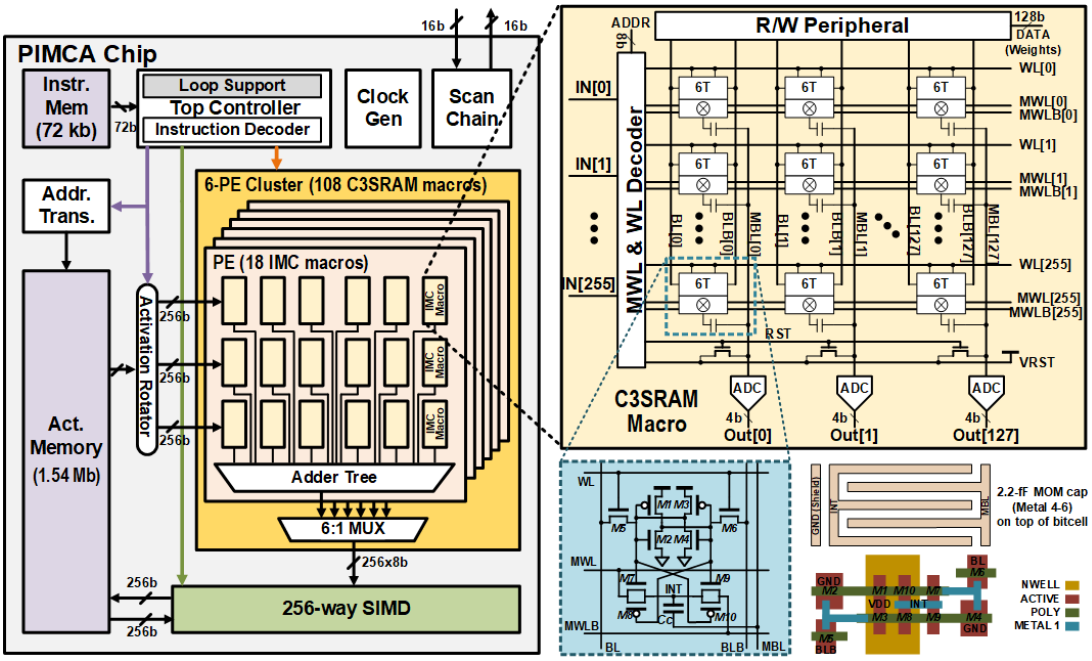


Figure 6.1: Overall architecture of PIMCA.

Fig. 6.1 depicts the overall architecture of PIMCA, which integrates 108 IMC SRAM macros organized in 6 processing elements (PEs). In each PE, 18 macros are organized in a 3×6 array. At each cycle, one PE is activated, and the active PE can perform matrix-vector multiplication (MVM) using 1 to 18 macros. Each of the selected macros yields 128 4-b partial sums, which can be accumulated to 256-d 8-b results by a configurable adder tree in the PE, in either 256-d 9-input mode for 3×3 convolution or 128-d 18-input mode for 5×5. The accumulation results can be further processed by the 256-way SIMD processor for non-MAC operations and the SIMD outputs are written back to the activation memory (AM).

We designed the IMC SRAM macro based on the C3SRAM module [117] described in Chapter 4, which contains 256 by 128 bitcells. Two transmission gates and a coupling capacitor are added to the traditional 6T SRAM bitcell. Transmission gates instead of NMOS pass gates [117] are used for better MAC word line driving capability. Unlike using MOS capacitors in the original C3SRAM in [117], the coupling capacitor Cc (2.2fF) is implemented as a MOM capacitor (M4-M6) on top of the bitcell for better area-efficiency and capacitance linearity. It performs 256×128 MVM in one cycle by simultaneously turning on all 256 rows and 128 columns. The binary multiplication result of each bitcell are accumulated over MBLs through capacitive coupling. The MBL voltage of each column is converted to 4-b values by an 11-level flash. The peripheral circuitry such as decoders, flash ADC, and R/W control is clock-gated to reduce the clock power when the macro is idle.

Fig. 6.2 illustrates the proposed ISA and the six-stage PE+SIMD pipeline. Every cycle, an instruction is fetched from instruction memory (IF), then decoded (ID), followed by reading input vectors from AM (LD), performing MAC (IMC) and vector operations (SIMD) in one of the PEs and the SIMD processor subsequently, and optionally writing the SIMD results back to AM (WB). The ISA has two types of instructions: regular PE+SIMD and loop instructions. A regular instruction performs MAC operation in PE and non-MAC operation in SIMD. It contains three major fields: i) read and write (R/W) addresses and AM enable, ii) PE and macro selection and accumulation mode control, iii) SIMD operands and operation code. The regular instructions contain a 6-b field defining repetition times (up to 64) of this instruction. This loop support saves the instruction counts for implementing the innermost loop such as horizontal sliding in each row of output feature map. Read and write addresses are automatically increased by one when one regular instruction is repeatedly executed. In addition, loop instructions are devised to support generic for-loops.

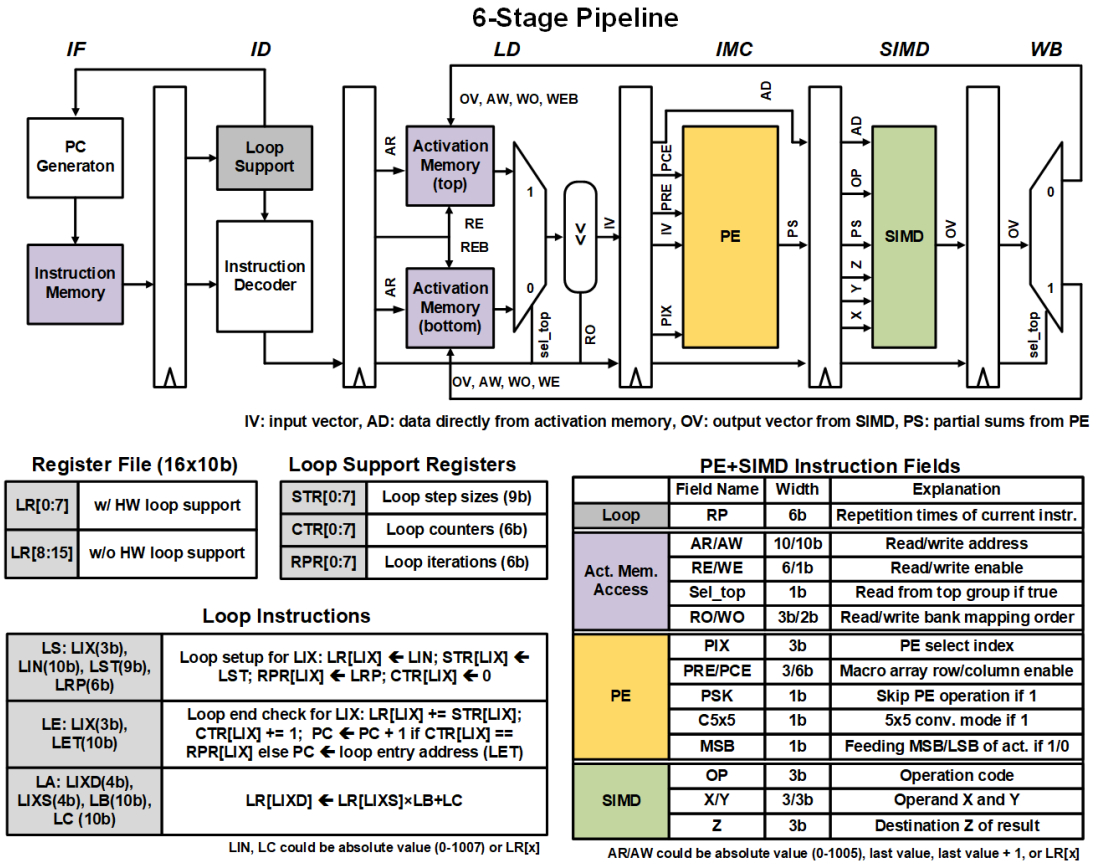


Figure 6.2: The 6-stage pipeline, instruction set and loop support of PIMCA.

Pairs of loop-setup (LS) and loop-end-check (LE) instructions can define up to eight levels of nested for-loops. The parameters of a loop are set in the dedicated sets of loop registers/counters (LR/LC). The proposed hardware loop support reduces the instruction count by 4X with minimal loop latency overhead.

6.2.2 Storage and Access Pattern of Activation Memory

PIMCA also integrates 1.54-Mb activation memory (AM) using off-the-shelf single-port SRAM for storing input image, intermediate data, BN parameters, and final outputs. To avoid R/W conflict, we split the AM in two groups: top and bottom. To compute a DNN layer, input data are read from the top (bottom) group, whereas

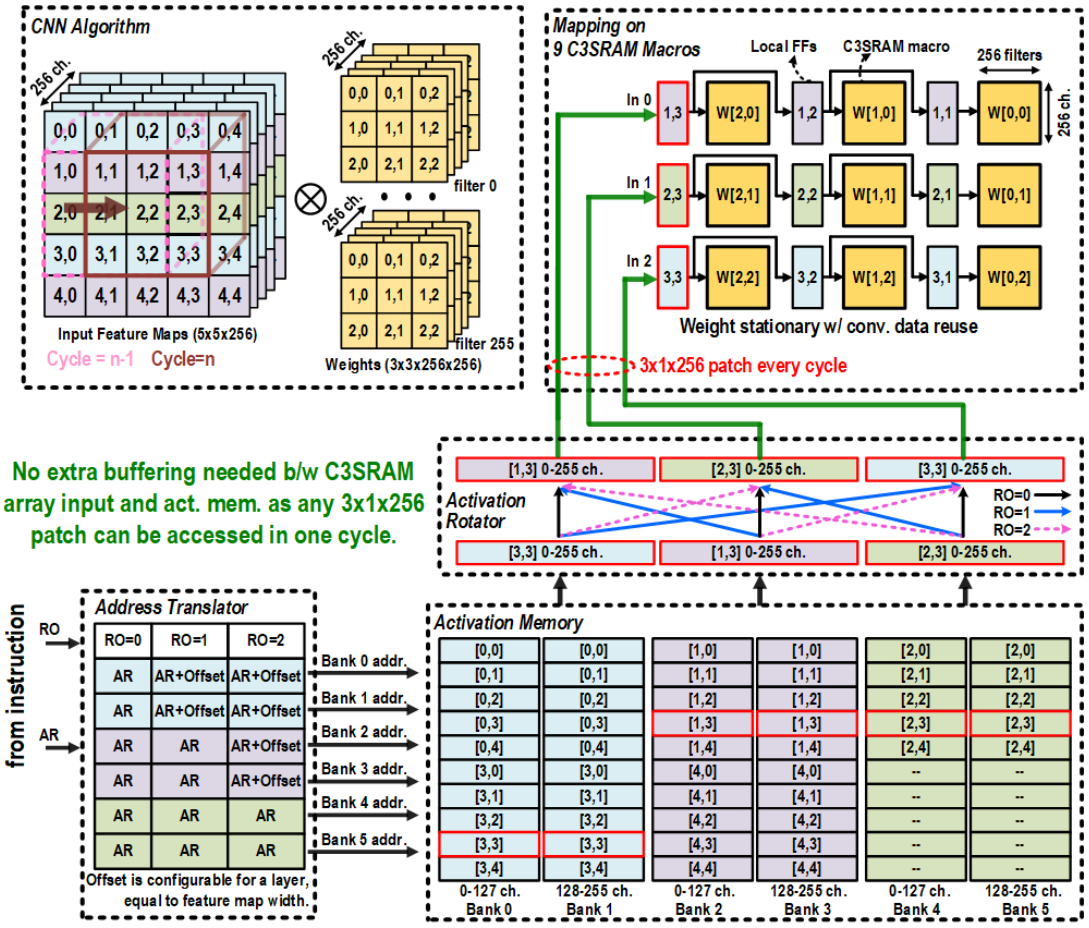


Figure 6.3: Storage and access pattern of activation memory of PIMCA.

the output data are written back to the bottom (top) group. Each group of the AM is further divided into six banks (1024×128 b) to support flexible yet efficient AM access. As shown in Fig. 3, each row of a bank can store the activations of up to 128 channels, and three consecutive rows of feature maps are stored across six different banks. Aided by proper address generation for different banks and activation rotation, the active PE can access any 3×1×256 (height×width×channel) input patch in a cycle, simplifying the streaming process by eliminating the need for extra buffering between AM and PE (Fig. 6.3).

6.2.3 PE Organization and Mapping

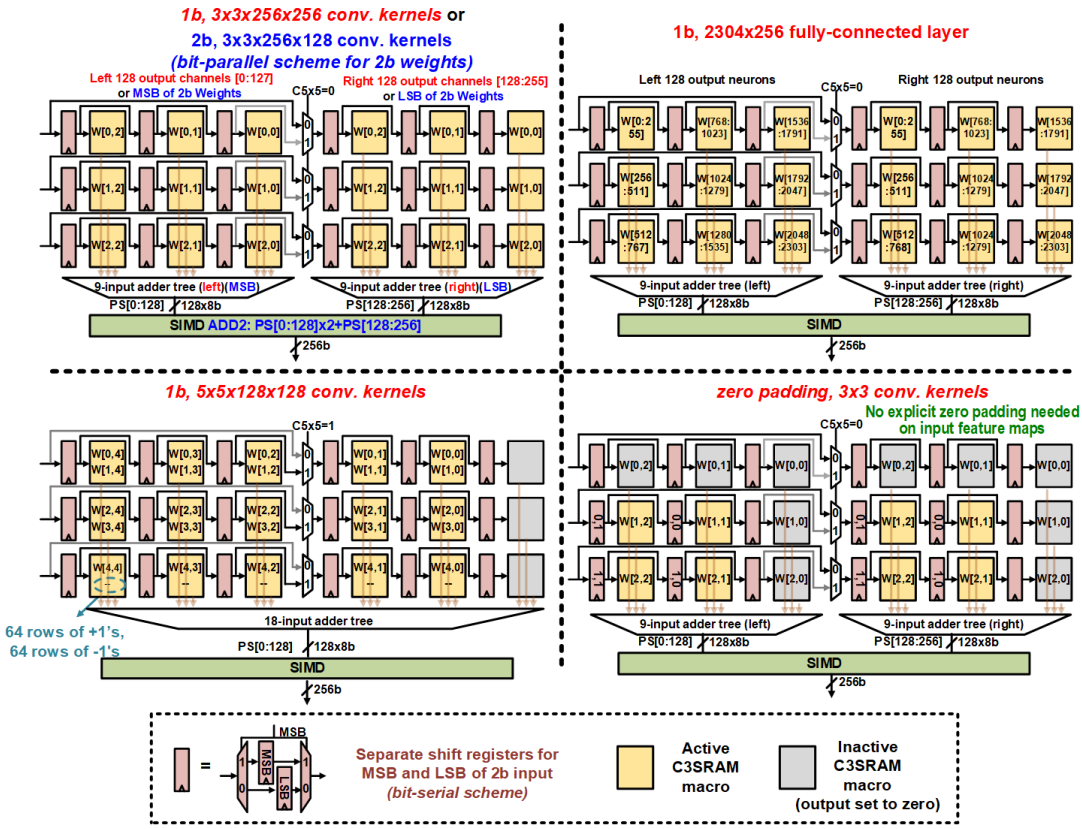


Figure 6.4: Flexible C3SRAM macro mapping of PE in PIMCA.

We organize the macros in a PE to effectively support typical convolution kernel sizes, namely 3×3, 5×5, and 1×1. For 3×3, the 3×6 IMC macros in a PE are split into two 3×3 groups. The 1-b or 2-b kernels of a 256×256 or 256×128 (input channels x output channels) convolution layer can be mapped in that PE (Fig. 6.4 top left), where the left and right groups share the input vectors. The input registers for the 3×3 macro group are pipelined horizontally, exploiting the convolutional data reuse. Also, 5×5 1-b convolution kernels of a 128×128 convolution layer can be mapped by connecting the output of input registers of the left group to the input of the right group (Fig. 6.4 bottom left). Inactive macros' outputs are held zeros; therefore, by

disabling corresponding macros when input padded zeros are being processed, explicit zero padding can be avoided (Fig. 6.4 bottom right), saving MVM computation energy for zero inputs.

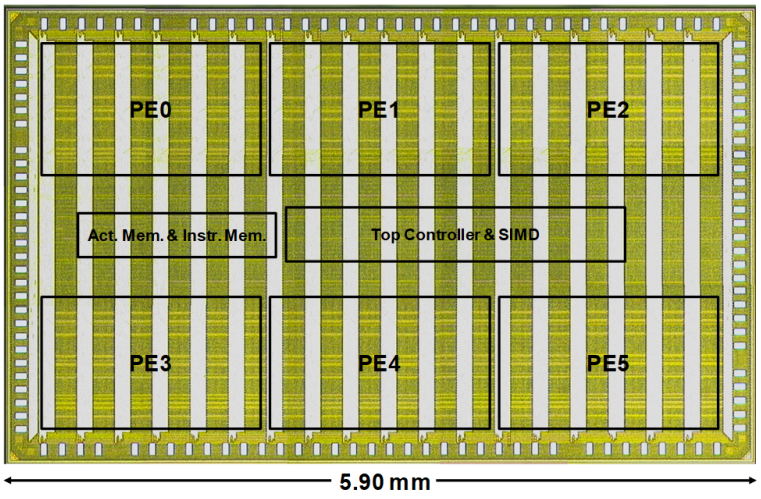
6.2.4 SIMD Processor

The 256-way SIMD processor performs non-MAC computing acceleration. It can directly use the output of the selected PE or fetch data from AM. Each way of the SIMD processor contains four 8-b registers (R0-R3) and a 10-b register (R4). The MSB of R4 of the 256 ways are taken as the output of the SIMD processor. Eight types of operations are supported:

1. ADD: perform addition between X and Y and store the result in Z;
2. ADD2: perform addition between 2X and Y and store the result in Z;
3. COMP: compare X and Y and save (X>Y) to Z;
4. CMP2: compare X against R0, R1, R2 and save (X>R0)+(X>R1)+(X>R2) to Z for 2-bit activation;
5. LOAD: load X to Z;
6. MAX: load maximum of X and Y to Z;
7. RSHIFT: right-shift X by amount of Y and save the result to Z;
8. LSHIFT: left-shift the output of data memory to X by 1 bit.

In particular, 'ADD2' operation (2X+Y) is included to support both bit-serial scheme for 2b input (X and Y from the same SIMD lane) and bit-parallel scheme (Fig. 6.4 top left) for 2b weight (X and Y from left and right lanes). 'LSHIFT'

operation is mainly designed for loading data from activation memory and writing data back to activation memory in bit-by-bit fashion.



Technology	28nm
Chip Area	20.9 mm^2
Digital SRAM	201 kB
IMC SRAM	432 kB
Supply Voltage	0.82/1 V
Frequency	40 MHz
Activation Precision	1/2 bit
Weight Precision	1/2 bit
Peak Performance	4.9 TOPS (1b)
Power	108 mW
Peak/Avg. Energy Efficiency	437/289 TOPS/W (1b)

Figure 6.5: PIMCA prototype chip micrograph and performance summary.

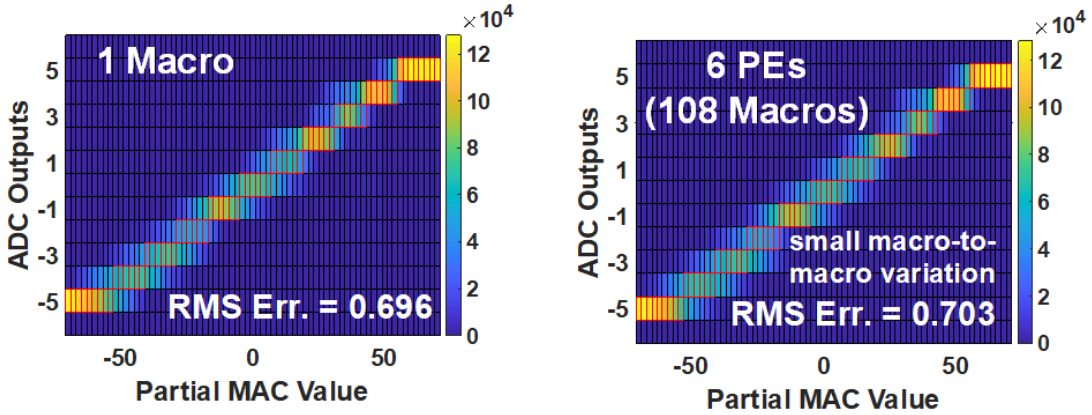


Figure 6.6: Measured histogram of ADC outputs vs. partial MAC values.

6.3 Measurement Results

The PIMCA testchip is implemented in 28nm CMOS (Fig. 6.5). It operates at 40 MHz under 1V supply. We evaluated the testchip on three different DNN workloads for CIFAR-10 datasets: VGG-9 (A:1b, W:1b), ResNet-18 (A:1b, W:1b) and

ResNet-18 (A:2b, W:2b), where A and W stand for activation and weight precision, respectively.

6.3.1 ADC Output Variation

Due to bit-cell capacitor mismatch, ADC offset variation and other non-ideal effects, the 11-level flash ADCs exhibit output errors which might hurt the DNN inference accuracy. To evaluate the ADC output errors, we first program all the C3SRAM macros to all +1's, and generate 400 random input binary vectors (+/-1) that contain the same number of +1's for each possible partial MAC value between -70 and +70. Fig. 6.6 (left) shows the histogram of ADC outputs vs. partial MAC values from a single C3SRAM macro while Fig. 6.6 (right) shows the histogram of ADC outputs vs. partial MAC values from all 108 macros. The root-mean-square (RMS) error of ADC outputs is 0.696 LSB and 0.703 LSB for single-macro and 108-macro cases, respectively. It can be seen that the macro-to-macro variation is relatively small as the ADC output RMS error does not increase much when we increase the scale from single-macro to 108-macro, which makes large-scale IMC design an viable option.

To alleviate the accuracy degradation due to the ADC output variation, we developed noise-aware DNN training framework that incorporates the measured conditional probability distribution inferred from Fig. 6.6 in the training process.

6.3.2 DNN Workload Accuracy and Energy Efficiency

Fig. 6.7 shows the detailed network structure and exact PE macro mapping of VGG-9 and ResNet-18 used in our experiments. Binary VGG-9 DNN requires 2.89 Mb weight storage, which can fully fit in the PIMCA chip. Binary ResNet18 need 4 times of mapping its convolution layers sequentially onto PIMCA chip. Measured

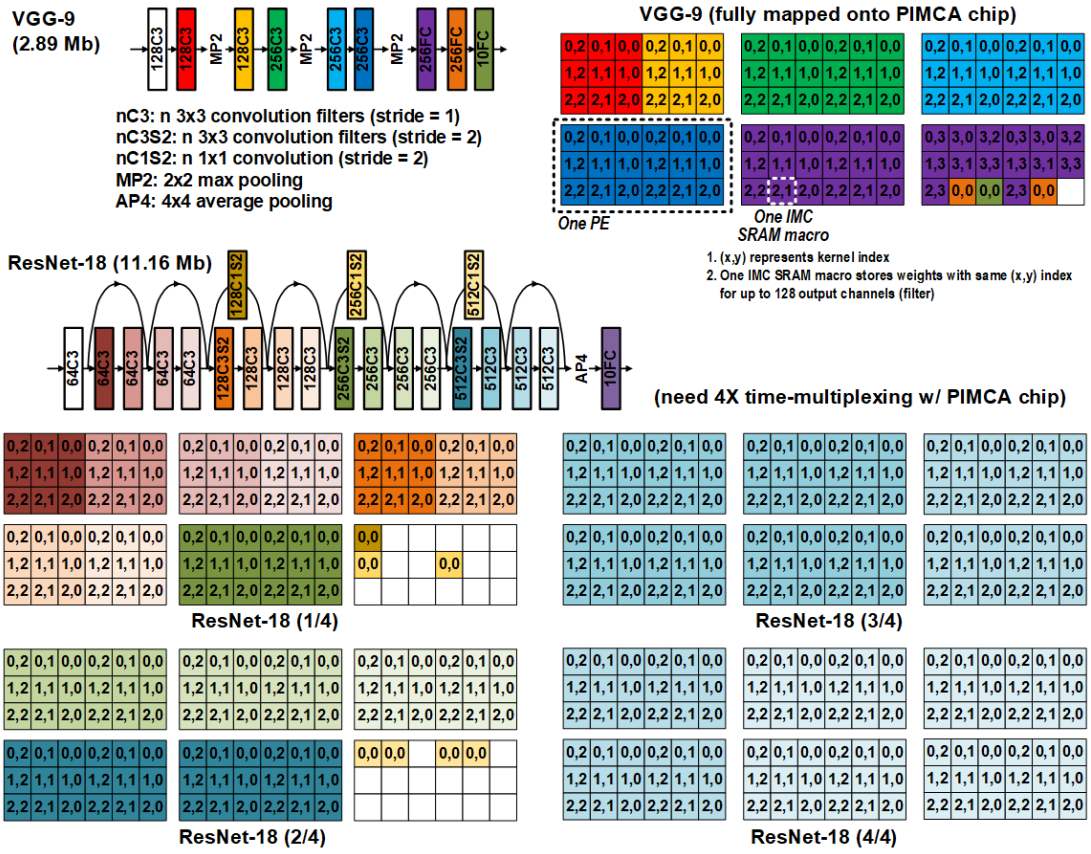


Figure 6.7: Mapping of VGG-9 and ResNet-18 CNNs on 6 PEs of PIMCA chip.

accuracy and energy efficiency of three different DNN workloads are summarized in Fig. 6.8. On-device inference of binary VGG-9 consumes 2.36 μJ energy and 47.3 μs latency, leading to 289 TOPS/W average energy-efficiency. When a DNN continuously uses 18 macros per PE, 437 TOPS/W peak energy-efficiency is achieved as shown in Fig. 6.9 (left). According to the measured power breakdown shown in Fig. 6.9(right), C3SRAM macro MWL decoder, driver and cell array dominate the total power of PIMCA chip.

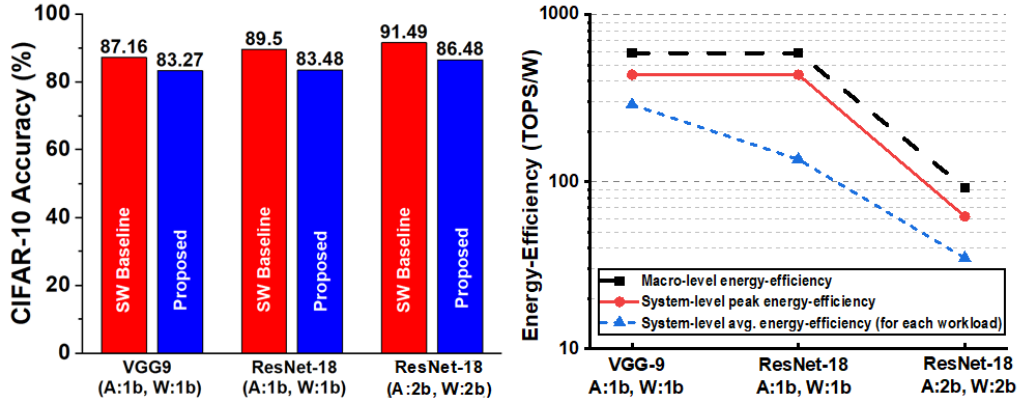


Figure 6.8: DNN accuracy and energy efficiency evaluation on PIMCA chip.

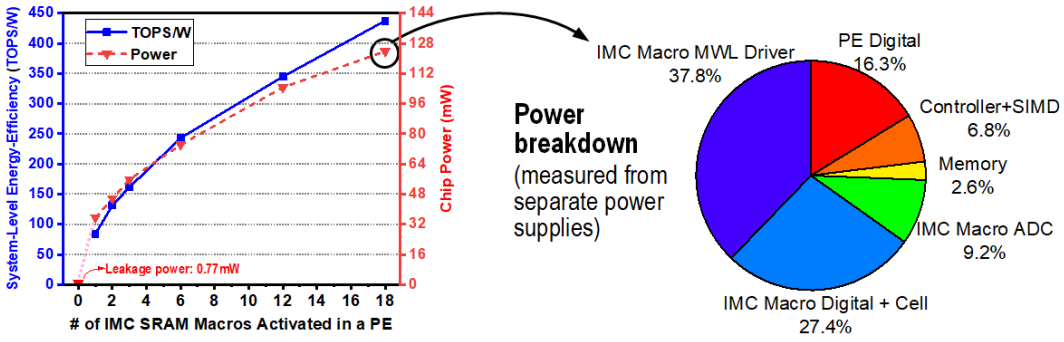


Figure 6.9: Power measurement of PIMCA prototype chip.

6.3.3 Comparison with Related Works

Table 6.1 shows the comparison to prior works on system-level IMC accelerator design. Compared to the existing system-level IMC SRAM designs [99, 119–121], the PIMCA chip demonstrates the largest integration of IMC SRAMs (3.4 Mb). With flexible non-MAC operation support in SIMD processor, the PIMCA chip can support a wide variety of layers such as max/average pooling layers, residual path addition, etc., for which other works rely on host processors [121] or external digital

processors for processing. With the aid of customized ISA and reconfigurability of PE macro mapping, different convolution kernels such as 5×5 and 1×1 can be efficiently supported in addition to the commonly used 35×53 convolution kernels. Compared to the other programmable IMC processor [121], PIMCA achieves higher macro-level energy efficiency and lower energy for binary VGG model inference.

6.4 Conclusion

In this chapter, we present a programmable in-memory computing (IMC) accelerator integrating 108 C3SRAM macros of a total size of 3.4 Mb, demonstrating one of the largest integrations for IMC design to date. A custom ISA featuring IMC+SIMD functional units and hardware loop support is proposed to support a range of deep neural network (DNN) layer types. The 28nm prototype chip achieves system-level energy efficiency of 437/62 TOPS/W at 40 MHz, 1V supply for DNNs with 1b/2b precision.

Table 6.1: Comparison to Prior Works on System-level IMC SRAM Design.

	VLSI'19 [119]	ISSCC'20 [120]	JSSC'19 [99]	JSSC'20 [121]	**PIMCA**
Technology	65nm	65nm	65nm	65nm	**28nm**
DNN Model	RNN	CNN/FC	CNN	CNN/FC	**CNN/FC**
Supported CNN Kernel	N/A	3×3	3×3	3×3	**3×3, 5×5, 1×1**
Flexible Non-MAC Operation Support	No	No	No	Yes	**Yes**
Supply Voltage (V)	0.9-1.1	0.9-1.05	0.94/0.68 /1.2	0.85-1.2	**0.82/1**
Area (mm^2)	9.6	5.66	12.6	13.5	**20.9**
Clock Frequency (MHz)	5-75	50-100	100	40-100	**40**
IMC Bitcell	6T	8T	10T1C	10T1C	**10T1C**
Digital SRAM (kb)	80	1,312	64	256	**1,608**
IMC SRAM (kb)	64	4	2,304	576	**3,456**
Bit Precision	1b	2/4/6/8b (Act.) 4/8b (Weight)	1b	1b-8b	**1b-2b**
ADC Precision	3b	5b	1b	8b	**4b**
Performance (TOPS)	0.61	0.17-2.0	18.9	2.2 (1b)	**4.9 (1b)**
Power (mW)	N/A	31.8-65.2	N/A	N/A	**124 (1b)**
IMC-Macro-Level Peak Energy Efficiency (TOPS/W)	51.6	158.7	866	400 (1b)	**588 (1b)**
System-Level Peak Energy Efficiency (TOPS/W)	11.7	35.8	658	N/A	**437 (1b)**
System-Level Avg. Energy Efficiency (TOPS/W) for ResNet-18	N/A	6.91 (4b/4b)	N/A	N/A	**136 (1b) 35 (2b)**
Energy per Inference for VGG (μJ)	N/A	N/A	3.55 (1b)	5.31 (1b)	**2.36 (1b)**

Chapter 7

A FIXED-POINT CONVOLUTIONAL NEURAL NETWORKS LEARNING PROCESSOR

7.1 Introduction

With the advent of artificial intelligence (AI), various deep neural networks (DNNs) such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have emerged and achieved human-level performance in image/speech recognition tasks. DNNs are typically trained by GPUs in 32-bit floating-point (FP32) precision after sending all training data to a central server. To train DNNs with a user's sensitive data, local training capability on mobile or edge devices is desired due to privacy concerns. DNN inference accelerators [123, 124] have progressively optimized performance and energy efficiency, but inference-only DNN accelerators cannot adapt to the evolving user-centric application scenarios. Since DNN training is considerably more compute-/memory-intensive than inference, it is challenging to perform DNN training on resource-limited mobile devices. In particular, the back-propagation (BP) based training algorithm involves weight transpose for convolution and fully-connected (FC) layers. Maintaining two copies of weights (original and transposed) or employing transposable SRAM [42] incurs large area overhead or custom SRAM bitcell design.

Recently, deep learning processors with on-chip learning functionalities have been proposed [125–127]. An AI core capable of both learning and inference for DNNs with programmable architecture and custom ISA was presented in [125], featuring a 2-D processing element (PE) array that supports floating-point precision and flexible

dataflow such as weight-/output-/row-stationary schemes. However, the flexibility comes at the cost of the complexity of PE-unit-level instructions and overhead in inter-PE communications. A reinforcement learning (RL) accelerator was presented in [126] using 16-bit floating-point precision (FP16), and proposed a transposable PE array with an additional buffer, which was optimized for FC layers. However, if the same technique were to be applied to convolution layers in CNNs, the additional buffer requirement will be even larger. A deep learning neural processing unit (LNPU) [127] was proposed for accelerating on-chip training of DNNs with input sparsity, using fine-grained mixed precision of FP16 and FP8. However, the on-chip weight transpose capabilities for both feed-forward (FF) and feed-backward (FB) phases have not been reported, without which additional off-chip memory processing will be required. The learning processors in [125–127] are all based on floating-point arithmetic, while training hardware using fixed-point precision could achieve higher density and energy efficiency [128].

In this chapter, we present a programmable CNN learning processor in 16-bit fixed-point precision (FX16) that performs end-to-end training using a simple top-level dataflow controller with layer-level instructions. Our CNN learning processor integrates large on-chip SRAM (>1 MB) that eliminates off-chip weight memory requirement for on-device training of small/medium-size CNNs such as LeNet-5 [129] with up to 131k parameters, and features a new cyclic weight storage scheme that allows accessing the same SRAMs for non-transpose/transpose operations during FF/FB phases for both convolution and FC layers of CNNs. The 65nm prototype chip achieves 2.6 TOPS/W at 0.55V, demonstrating energy efficiency improvements of 20-50% over ASICs with FP16 precision [126, 127] and >5X over Nvidia Titan XP GPU.

7.2 SGD Based Learning Algorithm

Stochastic gradient descent (SGD) based learning algorithm is a well known DNN optimizer. A common variant of SGD algorithm introduces a momentum term W_m which remembers previous update of parameters:

$$W_m^t = mW_m^{t-1} + \eta W_g^t, \tag{7.1}$$

$$W^t = W^{t-1} - W_m^t, \tag{7.2}$$

where η is the learning rate, m is the momentum factor and $0 < m < 1$, W_g is the gradient of loss function L with respect to weight W.

To calculate the weight gradient W_g, each iteration consists of three phases: feed-forward (FF), feed-backward (FB), and weight gradient calculation (WG), as shown in Fig. 7.1. In the FF phase, activations a_l in each layer are computed and stored layer by layer in the forward direction:

$$a_l = \varphi_l(W_l \cdot a_{l-1} + b_l), \tag{7.3}$$

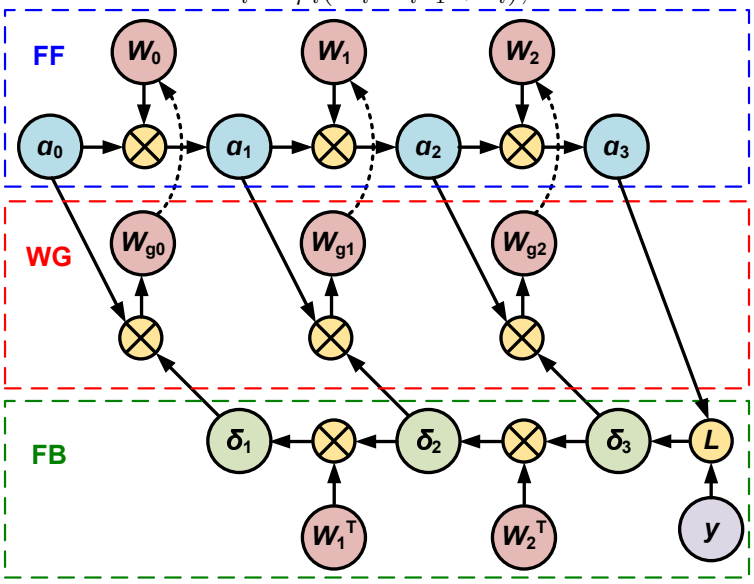


Figure 7.1: Back-propagation based DNN training algorithm in three phases: feed-forward (FF), feed-backward (FB), weight gradient calculation (WG).

where φ_l is the activation function such as ReLU, sigmoid, etc., "$\cdot$" represents linear matrix multiplication or convolution, and b_l is the bias value for layer l. The term in the parenthesis of Eq. (4) is called pre-activation value o_l.

In the FB phase, the local gradients δ_l ($=\partial L/\partial o_l$) are evaluated first at the output layer using the output activations and target labels, and then δ_l are back-propagated layer by layer:

$$\delta_l = (W_l^T \cdot \delta_{l+1}) \odot \varphi_l\prime(o_l), \tag{7.4}$$

where "$\odot$" represents element-wise multiplication. Note that the weights W are transposed during the FB phase. For 4-D convolution kernels, the two dimensions for input and output channels are transposed, while the other two dimensions are flipped. In the WG phase, the weight gradients $W_{g,l}$ are computed using previous-layer activations a_l and current-layer error signals δ_{l+1}:

$$W_{g,l} = \frac{1}{N} a_l^T \cdot \delta_{l+1}, \tag{7.5}$$

where N is the mini-batch size in each iteration.

In addition, max-pooling (MP) layers are often included in CNNs. After MP layers in the FF phase, activations are down-sampled by selecting the maximum value of each $p\times p$ patches of the feature maps, and MP indices are stored. During the FB phase, local gradients are back-propagated through MP layers such that the maximum-value pixels obtain the local gradients whereas other pixels get zeros.

7.3 Learning Processor Design and Optimization

In this section, we describe our proposed energy-efficient CNN learning processor that implements end-to-end SGD with momentum based learning algorithm.

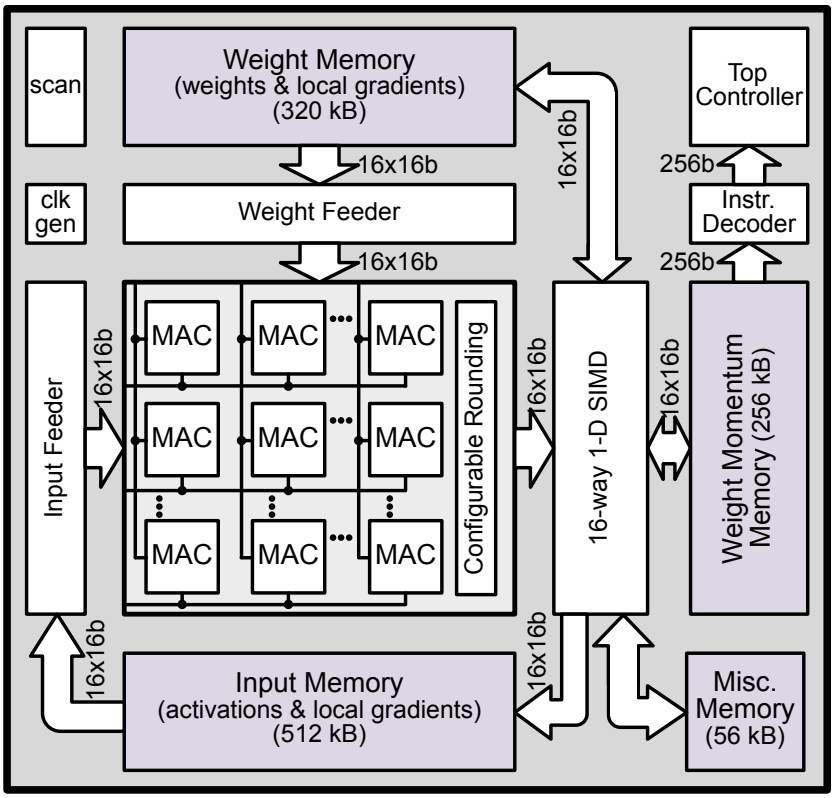


Figure 7.2: Overall architecture of the proposed CNN learning processor.

7.3.1 Overall Architecture and Custom ISA

Our CNN learning processor (Fig. 7.2) consists of a 2-D MAC array, input and weight feeder, a 16-way 1-D SIMD processor, input memory, weight memory, weight momentum memory and miscellaneous memory for bias, bias momentum, pooling indices, and activation derivatives. Output-stationary dataflow similar to [123] is employed for multi-dimensional MAC operations in the FF, FB, and WG phases of all the layers. 16 inputs from the input feeder and 16 weights from the weight feeder are broadcast across the 16×16 MAC array, where 4×4 patches of up to 16 output feature maps are computed in parallel as shown in Fig. 7.3. Both inputs and weights are in FX16 precision. The partial sums are accumulated in FX22 precision, which eventually are serially shifted out to a pipelined 16-way SIMD processor. A

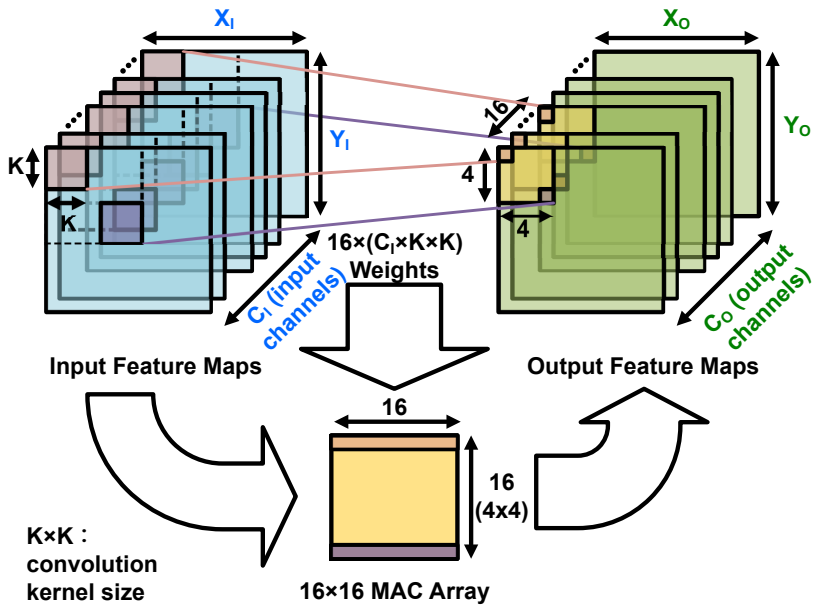


Figure 7.3: Output-stationary dataflow for a convolution layer using a 16×16 MAC array, where weights and inputs are broadcast across the MAC array vertically and horizontally, respectively, and the outputs are accumulated inside each MAC.

configurable rounding module (22-bit to 16-bit) is added before feeding MAC outputs to the SIMD processor, which performs element-wise operations such as ReLU, max-pooling, up-sampling, mask, local gradient transpose, weight/bias updates, and saving results back to on-chip memory. The entire columns or rows of the MAC array can be disabled by clock gating when they are not utilized or when the operands are zeros. The CNN learning processor supports kernel sizes from 3×3 to 5×5 with arbitrary zero padding.

To make the learning processor programmable for different CNN topologies and training hyper-parameters, we developed a custom instruction set architecture (ISA). The CNN model structure and the training hyper-parameters are embedded in a series of layer-level instructions. Fig. 7.4 shows the instruction fields defining a layer-level computation step in a training iteration. Each instruction might take many cycles to

Instruction Fields	Comments
Training Stage (2b)	FF / FB / WG
Layer Type (1b)	conv. / FC
Input Dimension (30b)	depth (12b) / height (9b) / width (9b)
Weight Dimension (16b)	filters (12b) / kernel Size (4b)
Zero Padding (4b)	0~15
Activation (1b)	ReLU / linear
Bias(1b)	yes / no
Data Addresses (83b)	input R/W (14b x 2), weight R/W (14b x 2), bias (7b), activation derivative (10b), pooling index (10b)
Special Post-processing (9b)	pooling / up-sampling / reshape / transpose / loss evaluation / generate activation derivatives (AD) / mask with AD / update weights / update biases
Training Hyper-parameters (41b)	learning rate (16b), momentum factor (16b), batch size (9b)
MAC array configuration (3b)	reconfigurable rounding (2b), skip MAC array (1b)
End of Program (1b)	one training iteration ends

Figure 7.4: Custom instruction fields that define layer-level computing steps.

complete, depending on the layer size and mini-batch size.

With limited on-chip memory size, the training mini-batch size (e.g., 40-100) is divided into J groups such that each group could fit in the on-chip memory. Weight/bias gradients resulted from each group are multiplied by the learning rate and then accumulated in the weight/bias momentum memory, and the weights/biases are updated after we obtain the averaged weight/bias gradients after J groups.

7.3.2 Input Convolutional Reuse

The input memory consists of 16 SRAM modules, and each SRAM has 16,384 rows of 16-bit words. Each row stores a 4×4 patch of the input feature maps. In the input feeder, a two-level FIFO array (Fig. 7.5) is included to reduce the input memory access. Since the receptive field of a 4×4 output patch with 5×5 kernel is of

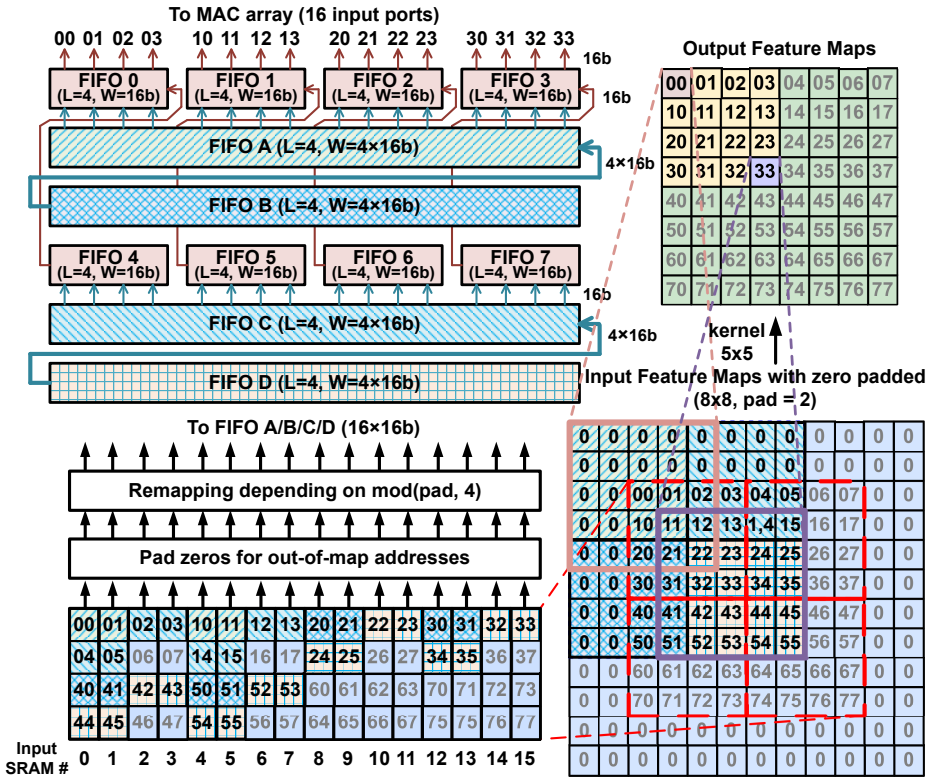


Figure 7.5: Input feeder with two-level FIFO array for convolutional data reuse and 16-input SRAM for flexible zero padding required in CNN training, illustrated with an example of 5×5 same-mode convolution (pad = 2) with 8×8 input feature maps.

size 8×8, four rows of inputs corresponding to an 8×8 receptive field are first loaded to FIFOs A/B/C/D in four cycles (Fig. 7.5). Subsequently, by properly loading data from FIFOs A/C to FIFOs 0-7 in parallel and shifting data stored in FIFOs 0-7 and FIFOs A-D serially, the outputs of FIFOs 0-3 are fed to the MAC array as the input data for convolution. Thanks to the input feature map storage pattern, no matter how many zeros are to be padded, any 4×4 patch in an input feature map is guaranteed to be stored in 16 different SRAMs and could be accessed in one cycle. Therefore, the input feeder can support any zero-padding for convolution with appropriate address generation and remapping logic (Fig. 7.5), which could be different between the FF

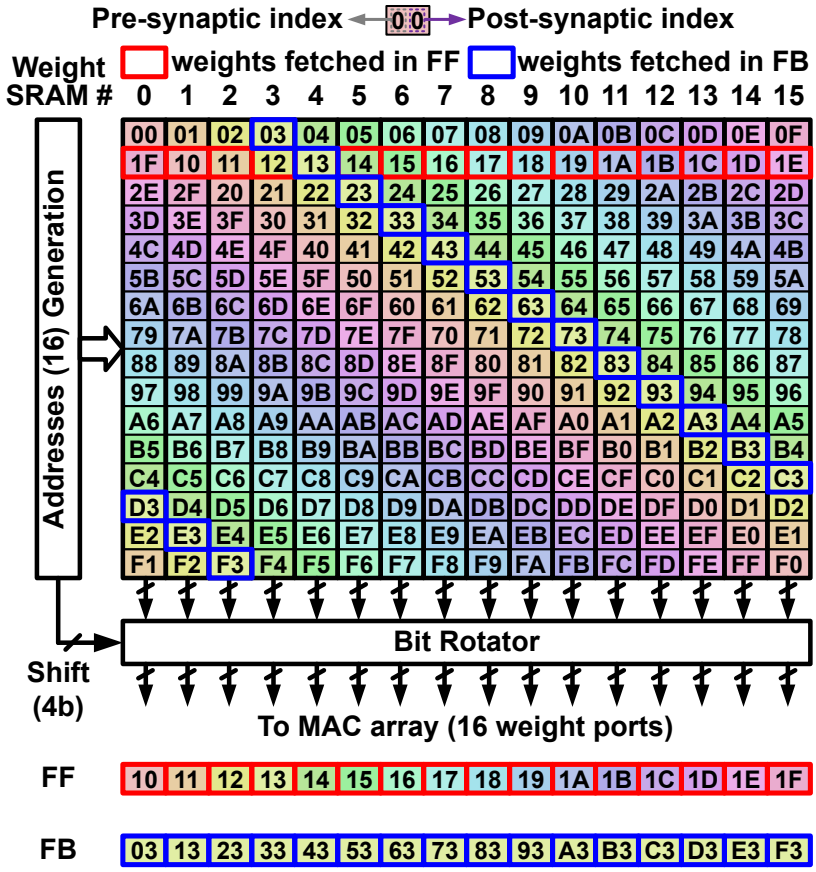


Figure 7.6: Weight feeder with cyclic weight storage/access scheme enables using the same off-the-shelf SRAMs for both non-transpose (FF phase) and transpose (FB phase) operations. Two examples are shown: (1) reading 16 weights associated with pre-synaptic neuron 1 in FF phase and (2) reading 16 weights with post-synaptic neuron 3 in FB phase.

and FB phases for the same convolutional layers.

7.3.3 Dual-read-mode Weight Storage Scheme

To avoid explicit weight transpose during FB phases, we propose a new on-chip weight storage and read scheme with off-the-shelf SRAMs (Fig. 7.6) that can be read in two modes: non-transpose mode and transpose mode. For FC layers, instead of

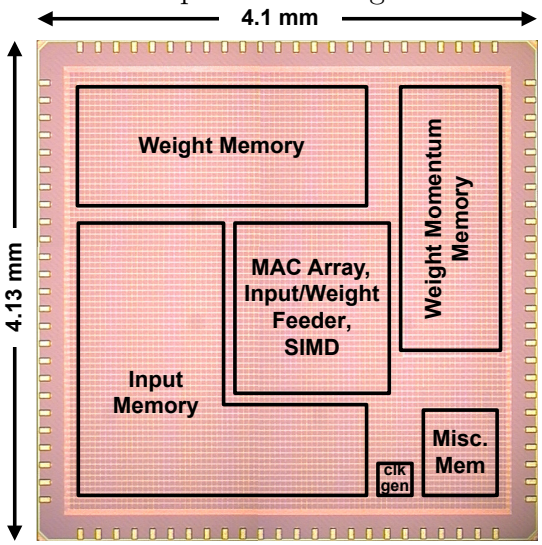
storing 16 weights (16×16-bit) associated with one presynaptic neuron in the same row of a single SRAM, we store the weights in 16 independent SRAMs, each with 16-bit word length (i.e. one weight per row), in a cyclic fashion (Fig. 7.6). The rotation amount of 16 weights in a row is $mod(ix, 16)$, where ix is the presynaptic neuron index. Combined with proper rotation by the bit rotator, both the 16 weights associated with one presynaptic neuron (non-transpose mode) and the 16 weights associated with one postsynaptic neuron (transpose mode) can be accessed in one cycle, eliminating the need for explicit weight transpose. The same idea is applied to the input memory access for activations/gradients of FC layers, such that we can read them in neuron-major order and image-major order in the FF/FB and WG phases, respectively. A similar diagnoal data mapping scheme was proposed in [130] for 2-D discrete/inverse discrete cosine transform (DCT/IDCT) in video coding standards.

Table 7.1: Comparison with Other Transpose Techniques.

	[42]	[126]	This work
SRAM Bitcell	Transposable 8T	6T	6T
Transpose Technique	Row-/column-wise read & write	Buffer broad-/uni-casting	Diagonal storage pattern & rotation
Area Overhead	>100%	N/A	6.4%
Weight Precision	1-bit/4-bit	16-bit	16-bit

The proposed dual-read-mode weight storage scheme can be applied to weights in convolution layers as well, with ix being the input channel index. Weights in a k×k kernel share the same amount of rotation, and the access order inside a k×k kernel is reversed during the FB phases. Compared to SRAM weight storage for

inference (FF only), the proposed scheme supports both FF and FB with only 6.4% area overhead due to the breakdown of the memory and the bit rotator. On the other hand, transposable bitcell designs [42] incur >100% overhead due to additional transistors and non-push-rule design.

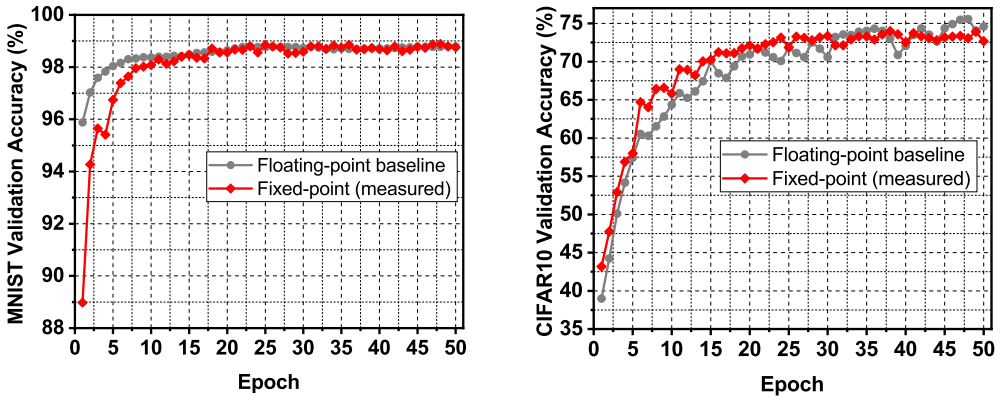


Technology	CMOS 65nm GP
Die Area	4.1 × 4.13 mm^2
Gate Count	9.2M (NAND2)
SRAM Size	1.12 MB
SRAM Bandwidth	27.6 GB/s
Clock Frequency	294.4 MHz @ 1V 53.1 MHz @ 0.55V
Power	299 mW @ 1V 10.45 mW @ 0.55V
Energy Efficiency	0.5 TOPS/W @ 1V 2.6 TOPS/W @ 0.55V

Figure 7.7: Prototype chip micrograph and performance summary.

7.3.4 Design Support and Limitations

In this design, other network layers such as 1×1 convolution and batch normalization are not currently supported. Weight decay and other regularization techniques are not included either. The loss function used in our hardware is the squared hinge loss, and other loss functions such as cross-entropy are left for future work. A more flexible SIMD processor is required to support these functions. Other popular optimizers, e.g. Adam, AdaGrad, etc., are not currently supported. To support these algorithms, the weight update rules should be more flexible and more memory storage should be allocated for 2nd-order momentum of weight gradients.



(a) 5-layer CNN for MNIST dataset

(b) 7-layer CNN for CIFAR-10 dataset

Figure 7.8: Measured accuracy convergence of DNNs. (CNN for MNIST: 8C3-8C3-MP2-16C3-16C3-MP2-FC10; CNN for CIFAR-10: 16C3-16C3-MP2-32C3-32C3-MP2-64C3-64C3-FC10.)

7.4 Measurement Results

The prototype chip is implemented in 65nm CMOS (Fig. 7.7). We trained 7-/5-layer CNNs for CIFAR-10/MNIST datasets for 50 epochs with the prototype chip (Fig. 7.8) programmed with 29/21 lines of custom layer-level instructions, respectively. Momentum factor m=0.5 and J=10 for both CNNs. Square hinge loss is used as the loss function for training. Since the on-chip memory can only hold data for 4 and 10 images in one training iteration for CIFAR-10 and MNIST CNNs, the mini-batch size is set to 40 ($4\times J$) and 100 ($10\times J$), respectively. The weights were randomly initialized. Input images are loaded to the beginning of the input memory through a scan chain. Measurements on training both CNNs show on-par validation accuracies compared to the FP32 baselines by GPUs (Fig. 7.8).

At 1.0/0.55V, the CNN learning processor consumes 299/10.45 mW power at 294/53 MHz clock frequency, achieving energy efficiency of 0.50/2.60 TOPS/W (Fig. 7.9). The average energy efficiency for training the 7-layer CNN for CIFAR-10 is

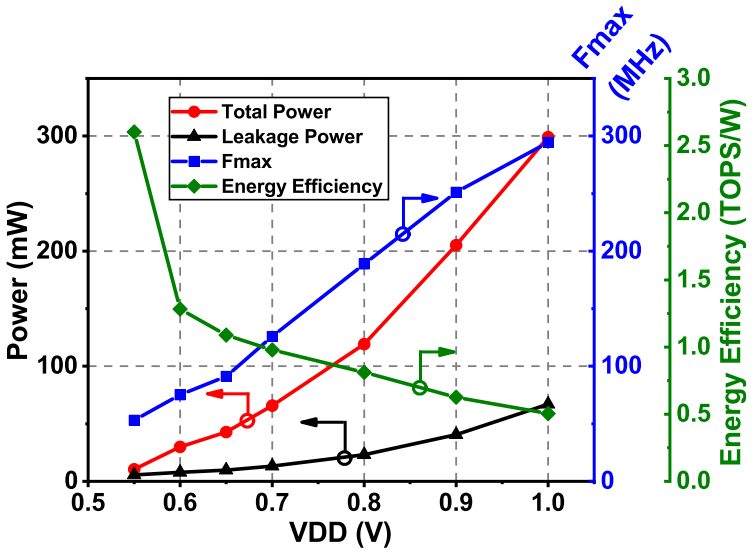


Figure 7.9: Power/energy measurements with voltage and frequency scaling.

1.74 TOPS/W at 0.55V. Table 7.2 shows the comparison with prior works. With >1 MB on-chip memory capacity, our 65nm CNN learning processor with FX16 precision demonstrates energy efficiency improvements of 20-50% over ASICs with FP16 precision [126, 127] and >5× over 16nm Nvidia Titan XP GPU with FP32 precision.

7.5 Conclusion

In this work, we present a programmable CNN learning processor in FX16 precision with a custom ISA and layer-level instructions. A two-level FIFO array and input storage scheme is employed for convolutional data reuse and flexible zero padding support. A dual-read-mode cyclic weight storage scheme is implemented to enable non-transpose/transpose-mode weight access without explicit transpose operations during training. The 65nm prototype chip achieves peak/average energy efficiency of 2.6/1.74 TOPS/W at 0.55V.

Table 7.2: Comparison with Prior Works.

	[125]	[126]	[127]	Titan XP GPU	This Work
Technology (nm)	14	65	65	16	65
Models	CNN MLP RNN	MLP	CNN MLP RNN	CNN MLP RNN	CNN
Supply (V)	0.58-0.9	0.67-1.1	0.78-1.1	–	0.55-1
Area (mm2)	9	16	16	471	16.9
Clock (MHz)	750-1500	5-200	50-200	1582	53.1-294.4
SRAM (MB)	2	0.44	0.36	>3	1.12
PE Precision	FP16	FP16 (Feature) FX16/8/4 (Weight)	FP16/8	FP64/32/16	FX16
Peak Perf. (GOPS)	1500 (16b)	204 (16b)	300 (16b)	12,150 (32b)	151
Power (mW)	–	2.4-196	43.1-367	250,000	10.45-299
Energy Efficiency (TOPS/W)	–	2.16 (16b)	1.74 (16b) 3.48 (8b)	0.48 (32b)	0.50-2.60

Chapter 8

CONCLUSION

In this dissertation, a comprehensive study of energy-efficient circuit and architecture designs for intelligent systems is presented.

The dissertation first presented an ultra-low-power ECG processor that performs both biometric authentication and personal cardiac monitoring. Aided by output layer removal, low precision, and Lasso-based compression, a total of 390× NN memory is reduced compared to software. We achieved <2.5% EER for a 645-subject in-house ECG database, consuming 1.06 μW power for real-time ECG authentication. For arrhythmia detection, the proposed ECG processor achieved 93.13%/89.78% sensitivity/specificity for 42 subjects in MIT-BIH arrhythmia database. The proposed ECG processor enables secure access and cardiac monitoring in wearable devices with stringent power/area constraints.

Next, two SRAM-based in-memory computing macro designs and one multi-array in-memory computing architecture titled Vesti are presented. The 256×64 XNOR-SRAM and C3SRAM macros achieve energy efficiency of 403 TOPS/W and 672 TOPS/W, respectively. The Vesti accelerator is fully designed and laid out in 65nm CMOS, achieving energy consumption of $<$ 20 nJ for MNIST classification and ¡ 40 μJ for CIFAR-10 classification at 1.0 V supply. More recently, a programmable C3SRAM based in-memory computing accelerator named "PIMCA" is proposed, which consists of 108 256x128 C3SRAM macros. A customized instruction set assembly is designed to program the 6-stage pipeline of the accelerator. The 28nm prototype chip achieves system-level energy efficiency of 437/62 TOPS/W at 40 MHz, 1V supply for DNNs with 1b/2b precision.

Finally, in addition to the DNN inference accelerators, a programmable CNN learning processor in FX16 precision with a custom ISA and layer-level instructions is presented. A two-level FIFO array and input storage scheme is employed for convolutional data reuse and flexible zero padding support. A dual-read-mode cyclic weight storage scheme is implemented to enable non-transpose/transpose-mode weight access without explicit transpose operations during training. The 65nm prototype chip achieves peak/average energy efficiency of 2.6/1.74 TOPS/W at 0.55V.

In summary, a number of circuit-level and architecture-level hardware innovations are presented, exploiting the reduced algorithm complexity with pruning and low-precision quantization techniques. Hardware and software co-design plays a key role in achieving the goal of energy-efficient computing. In particular, macro-level and system-level SRAM based in-memory computing works presented in this dissertation show that SRAM based IMC is one of the promising hardware solutions for energy-efficient intelligent systems.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, pp. 1097–1105.

[2] 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, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778.

[3] K. J. Geras, S. Wolfson, Y. Shen, N. Wu, S. G. Kim, E. Kim, L. Heacock, U. Parikh, L. Moy, and K. Cho, "High-resolution breast cancer screening with multi-view deep convolutional neural networks," *CoRR*, vol. abs/1703.07047, 2017.

[4] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke, "The Microsoft 2017 conversational speech recognition system," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 5934–5938.

[5] A. Y. Hannun, P. Rajpurkar, M. Haghpanahi, G. H. Tison, C. Bourn, M. P. Turakhia, and A. Y. Ng, "Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network," *Nature Medicine*, vol. 25, pp. 65–69, 2019.

[6] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, 2019, pp. 6105–6114.

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

[8] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm FDSOI," in *2017 IEEE International Solid-State Circuits Conference, ISSCC 2017, San Francisco, CA, USA, February 5-9, 2017*. IEEE, 2017, pp. 246–247.

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

[10] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[11] J. Song, Y. Cho, J. Park, J. Jang, S. Lee, J. Song, J. Lee, and I. Kang, "An 11.5TOPS/W 1024-MAC butterfly structure dual-core sparsity-aware neural processing unit in 8nm flagship mobile SoC," in *IEEE International Solid-State Circuits Conference (ISSCC)*, February 2019, pp. 130–132.

[12] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information and Processing Systems (NIPS)*, 2015, pp. 1135–1143.

[13] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *NIPS*, 2016.

[14] D. Kadetotad, S. Arunachalam, C. Chakrabarti, and J. Seo, "Efficient memory compression in deep neural networks using coarse-grain sparsification for speech applications," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, November 2016.

[15] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *International Conference on Learning Representations*, 2019.

[16] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, ser. Lecture Notes in Computer Science, vol. 9908. Springer, 2016, pp. 525–542.

[17] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 4107–4115.

[18] J. Choi, S. Venkataramani, V. Srinivasan, K. Gopalakrishnan, Z. Wang, and P. Chuang, "Accurate and efficient 2-bit quantized neural networks," in *Conference on Systems and Machine Learning (SysML)*, 2019.

[19] D. Shin, J. Lee, J. Lee, and H. Yoo, "DNPU: an 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," in *2017 IEEE International Solid-State Circuits Conference, ISSCC 2017, San Francisco, CA, USA, February 5-9, 2017*. IEEE, 2017, pp. 240–241.

[20] P. N. Whatmough, S. K. Lee, H. Lee, S. Rama, D. M. Brooks, and G. Wei, "A 28nm SoC with a 1.2GHz 568nJ/prediction sparse deep-neural-network engine with >0.1 timing error rate tolerance for IoT applications," in *2017 IEEE International Solid-State Circuits Conference, ISSCC 2017, San Francisco, CA, USA, February 5-9, 2017*. IEEE, 2017, pp. 242–243.

[21] S. Kim and M. Seok, "Variation-tolerant, ultra-low-voltage microprocessor with a low-overhead, within-a-cycle in-situ timing-error detection and correction technique," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 6, pp. 1478–1490, 2015.

[22] B. Moons, D. Bankman, L. Yang, B. Murmann, and M. Verhelst, "BinarEye: An always-on energy-accuracy-scalable binary cnn processor with all memory on chip in 28nm CMOS," in *IEEE Custom Integrated Circuits Conference (CICC)*, April 2018, pp. 1–4.

[23] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, "An always-on 3.8 μj/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 158–172, Jan 2019.

[24] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[25] D. Wan, F. Shen, L. Liu, F. Zhu, J. Qin, L. Shao, and H. Tao Shen, "TBN: Convolutional neural network with ternary inputs and binary weights," in *European Conference on Computer Vision (ECCV)*, September 2018.

[26] J. Zhang, Z. Wang, and N. Verma, "A machine-learning classifier implemented in a standard 6T SRAM array," in *2016 IEEE Symposium on VLSI Circuits, VLSIC 2016, Honolulu, HI, USA, June 15-17, 2016*. IEEE, 2016, pp. 1–2.

[27] Q. Dong, S. Jeloka, M. Saligane, Y. Kim, M. Kawaminami, A. Harada, S. Miyoshi, D. Blaauw, and D. Sylvester, "A 0.3V VDDmin 4+2T SRAM for searching and in-memory computing using 55nm DDC technology," in *2017 Symposium on VLSI Circuits*, June 2017, pp. C160–C161.

[28] 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, San Francisco, CA, USA, February 11-15, 2018*. IEEE, 2018, pp. 488–490.

[29] S. K. Gonugondla, M. Kang, and N. R. Shanbhag, "A 42pJ/decision 3.12TOPS/W robust in-memory machine learning classifier with on-chip training," in *2018 IEEE International Solid-State Circuits Conference, ISSCC 2018, San Francisco, CA, USA, February 11-15, 2018*. IEEE, 2018, pp. 490–492.

[30] W. Khwa, J. Chen, J. Li, X. Si, E. Yang, X. Sun, R. Liu, P. Chen, Q. Li, S. Yu, and M. Chang, "A 65nm 4Kb algorithm-dependent computing-in-memory SRAM unit-macro with 2.3ns and 55.8TOPS/W fully parallel product-sum operation for binary DNN edge processors," in *2018 IEEE International Solid-State Circuits Conference, ISSCC 2018, San Francisco, CA, USA, February 11-15, 2018*. IEEE, 2018, pp. 496–498.

[31] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A mixed-signal binarized convolutional-neural-network accelerator integrating dense weight storage and multiplication for reduced data movement," in *2018 IEEE Symposium on VLSI Circuits, Honolulu, HI, USA, June 18-22, 2018*. IEEE, 2018, pp. 141–142.

[32] Z. Jiang, S. Yin, M. Seok, and J. Seo, "XNOR-SRAM: In-memory computing sram macro for binary/ternary deep neural networks," in *2018 IEEE Symposium on VLSI Technology*, 2018, pp. 173–174.

[33] P. Srivastava, M. Kang, S. K. Gonugondla, S. Lim, J. Choi, V. S. Adve, N. S. Kim, and N. R. Shanbhag, "PROMISE: an end-to-end design of a programmable mixed-signal accelerator for machine-learning algorithms," in *45th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2018, Los Angeles, CA, USA, June 1-6, 2018*. IEEE Computer Society, 2018, pp. 43–56.

[34] C. Eckert, X. Wang, J. Wang, A. Subramaniyan, R. R. Iyer, D. Sylvester, D. T. Blaauw, and R. Das, "Neural cache: Bit-serial in-cache acceleration of deep neural networks," in *45th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2018, Los Angeles, CA, USA, June 1-6, 2018*. IEEE Computer Society, 2018, pp. 383–396.

[35] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2016, pp. 4107–4115.

[36] X. Si, J. Chen, Y. Tu, W. Huang, J. Wang, Y. Chiu, W. Wei, S. Wu, X. Sun, R. Liu, S. Yu, R. Liu, C. Hsieh, K. Tang, Q. Li, and M. Chang, "A twin-8T SRAM computation-in-memory macro for multiple-bit CNN-based machine learning," in *IEEE International Solid-State Circuits Conference (ISSCC)*, February 2019, pp. 396–398.

[37] L. Chang, R. K. Montoye, Y. Nakamura, K. A. Batson, R. J. Eickemeyer, R. H. Dennard, W. Haensch, and D. Jamsek, "An 8T-SRAM for variability tolerance and low-voltage operation in high-performance caches," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 4, pp. 956–963, April 2008.

[38] J. Wang, X. Wang, C. Eckert, A. Subramaniyan, R. Das, D. Blaauw, and D. Sylvester, "A compute SRAM with bit-serial integer/floating-point operations for programmable in-memory vector acceleration," in *IEEE International Solid-State Circuits Conference (ISSCC)*, February 2019, pp. 224–226.

[39] J. Seo, B. Brezzo, Y. Liu, B. D. Parker, S. K. Esser, R. K. Montoye, B. Rajendran, J. A. Tierno, L. Chang, and D. S. Modha, "A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *IEEE Custom Integrated Circuits Conference (CICC)*, 2011, pp. 1–4.

[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, April 2017.

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

[42] J. Seo *et al.*, "A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *IEEE Custom Integrated Circuits Conference (CICC)*, 2011.

[43] Apple, "Watch Series 3," https://www.apple.com/apple-watch-series-3/.

[44] Samsung, "Gear S3," https://www.samsung.com/global/galaxy/gear-s3/.

[45] K. H. Lee and N. Verma, "A low-power processor with configurable embedded machine-learning accelerators for high-order and adaptive analysis of medical-sensor signals," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 7, pp. 1625–1637, July 2013.

[46] H. Kim, R. F. Yazicioglu, T. Torfs, P. Merken, H. J. Yoo, and C. V. Hoof, "A low power ECG signal processor for ambulatory arrhythmia monitoring system," in *IEEE Symposium on VLSI Circuits*, June 2010, pp. 19–20.

[47] Y. P. Chen, D. Jeon, Y. Lee, Y. Kim, Z. Foo, I. Lee, N. B. Langhals, G. Kruger, H. Oral, O. Berenfeld, Z. Zhang, D. Blaauw, and D. Sylvester, "An injectable 64 nW ECG mixed-signal SoC in 65 nm for arrhythmia monitoring," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 375–390, Jan 2015.

[48] S. Y. Hsu, Y. Ho, P. Y. Chang, C. Su, and C. Y. Lee, "A 48.6-to-105.2 μW machine learning assisted cardiac sensor SoC for mobile healthcare applications," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 4, pp. 801–811, April 2014.

[49] iRhythm, "Zio XT Patch," http://irhythmtech.com/zio-services.php.

[50] AliveCor, "KardiaBand," https://store.alivecor.com.

[51] Samsung, "Simband," https://www.simband.io.

[52] Token, "The Ring," https://tokenring.com/the-ring.

[53] J. Lee, S. Noh, K. R. Park, and J. Kim, "Iris recognition in wearable computer," *Biometric Authentication*, vol. 3072, pp. 475–483, 2004.

[54] J. Chauhan, H. J. Asghar, A. Mahanti, and M. A. Kaafar, "Gesture-based continuous authentication for wearable devices: The smart glasses use case," in *Applied Cryptography and Network Security*, 2016, pp. 648–665.

[55] A. D. C. Chan, M. M. Hamdy, A. Badre, and V. Badee, "Wavelet distance measure for person identification using electrocardiograms," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 2, pp. 248–253, February 2008.

[56] I. Odinaka, P. H. Lai, A. D. Kaplan, J. A. O'Sullivan, E. J. Sirevaag, S. D. Kristjansson, A. K. Sheffield, and J. W. Rohrbaugh, "ECG biometrics: A robust short-time frequency analysis," in *IEEE International Workshop on Information Forensics and Security*, December 2010, pp. 1–6.

[57] A. Page, A. Kulkarni, and T. Mohsenin, "Utilizing deep neural nets for an embedded ECG-based biometric authentication system," in *IEEE Biomedical Circuits and Systems Conference (BioCAS)*, October 2015, pp. 1–4.

[58] S. J. Kang, S. Y. Lee, H. I. Cho, and H. Park, "ECG Authentication System Design Based on Signal Analysis in Mobile and Wearable Devices," *IEEE Signal Processing Letters*, vol. 23, no. 6, pp. 805–808, June 2016.

[59] R. D. Labati, E. Munoz, V. Piuri, R. Sassi, and F. Scotti, "Deep-ECG: Convolutional neural networks for ECG biometric recognition," *Pattern Recognition Letters*, March 2018.

[60] S. Yin, M. Kim, D. Kadetotad, Y. Liu, C. Bae, S. J. Kim, Y. Cao, and J. sun Seo, "A 1.06 μW smart ecg processor in 65nm CMOS for real-time biometric authentication and personal cardiac monitoring," in *IEEE Symposium on VLSI Circuits*, 2017, pp. C102–C103.

[61] The MIT-BIH Normal Sinus Rhythm Database, doi: 10.13026/C2NK5R.

[62] T. S. Lugovaya, "Biometric human identification based on electrocardiogram," Master's thesis, Faculty of Computing Technologies and Informatics, Electrotechnical University "LETI", Saint-Petersburg, Russian Federation, 2005.

[63] G. B. Moody and R. G. Mark, "The impact of the MIT-BIH arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001.

[64] Y. Liu, X. Feng, C. Zhang, C. Bae, and S.-J. Kim, "Electrocardiogram (ECG) authentication method and apparatus," US Patent Application 20 170 188 971A1. [Online]. Available: https://patents.google.com/patent/US20170188971A1/en

[65] C. Choi, Y. Kim, and K. Shin, "A PD control-based QRS detection algorithm for wearable ECG applications," in *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, August 2012, pp. 5638–5641.

[66] L. Wieclaw, Y. Khoma, P. Fałat, D. Sabodashko, and V. Herasymenko, "Biometric identification from raw ECG signal using deep learning techniques," in *IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 1, September 2017, pp. 129–133.

[67] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[68] J. Bombardieri, "Systolic pipeline architectures for symmetric convolutions," *IEEE Transactions on Signal Processing*, vol. 40, no. 5, pp. 1253–1258, May 1992.

[69] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society (Series B)*, vol. 58, pp. 267–288, 1996.

[70] A. Goldberger, L. Amaral, L. Glass, J. Hausdorff, P. C. Ivanov, R. Mark, J. Mietus, G. Moody, C. Peng, and H. Stanley, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, June 2000.

[71] Texas Instruments, "ADS1292R," http://www.ti.com/product/ADS1292R.

[72] K. K. M. Shreyas, S. Rajeev, K. Panetta, and S. S. Agaian, "Fingerprint authentication using geometric features," in *IEEE International Symposium on Technologies for Homeland Security (HST)*, April 2017, pp. 1–7.

[73] A. Uka, A. Roçi, and O. Koç, "Improved segmentation algorithm and further optimization for iris recognition," in *IEEE EUROCON - 17th International Conference on Smart Technologies*, July 2017, pp. 85–88.

[74] A. Iwasa, M. Hwa, A. Hassankhani, T. Liu, , and S. M. Narayan, "Abnormal heart rate turbulence predicts the initiation of ventricular arrhythmias," *Pacing and Clinical Electrophysiology*, vol. 28, no. 11, pp. 1189–1197, 2005.

[75] U. Ofoma, F. He, M. L. Shaffer, G. V. Naccarelli, and D. Liao, "Premature cardiac contractions and risk of incident ischemic stroke," *Journal of the American Heart Association*, vol. 1, no. 5, October 2012.

[76] P. Albrecht, "ST segment characterization for long term automated ecg analysis," Ph.D. dissertation, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1983.

[77] N. Verma, H. Jia, H. Valavi, Y. Tang, M. Ozatay, L. 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.

[78] J. Max, "Quantizing for minimum distortion," *IRE Transactions on Information Theory*, vol. 6, no. 1, pp. 7–12, March 1960.

[79] I. Hubara, https://github.com/itayhubara/BinaryNet.pytorch.

[80] H. Jia, Y. Tang, H. Valavi, J. Zhang, and N. Verma, "A microprocessor implemented in 65nm cmos with configurable and bit-scalable accelerator for programmable in-memory computing," *arXiv preprint arXiv:1811.04047*, 2018.

[81] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169–198, 1999.

[82] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7132–7141.

[83] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[84] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[85] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[86] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015, pp. 1–9.

[87] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, ser. JMLR Workshop and Conference Proceedings, vol. 37. JMLR.org, 2015, pp. 448–456.

[88] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *CoRR*, vol. abs/1609.07061, 2016.

[89] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, ser. JMLR Workshop and Conference Proceedings, vol. 37. JMLR.org, 2015, pp. 1737–1746.

[90] M. Courbariaux, Y. Bengio, and J. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 2015, pp. 3123–3131.

[91] S. Zhou, Z. Ni, X. Zhou, H. Wen, Y. Wu, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *CoRR*, vol. abs/1606.06160, 2016.

[92] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, 2017, pp. 345–353.

[93] T. Guan, X. Zeng, and M. Seok, "Recursive binary neural network learning model with 2.28b/weight storage requirement," *CoRR*, vol. abs/1709.05306, 2017.

[94] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[95] M. Kang, S. K. Gonugondla, and N. R. Shanbhag, "A 19.4 nJ/decision 364k decisions/s in-memory random forest classifier in 6T SRAM array," in *43rd IEEE European Solid State Circuits Conference, ESSCIRC 2017, Leuven, Belgium, September 11-14, 2017*. IEEE, 2017, pp. 263–266.

[96] W. Chen, K. Li, W. Lin, K. Hsu, P. Li, C. Yang, C. Xue, E. Yang, Y. Chen, Y. Chang, T. Hsu, Y. King, C. Lin, R. Liu, C. Hsieh, K. Tang, and M. Chang, "A 65nm 1mb nonvolatile computing-in-memory reram macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors," in *2018 IEEE International Solid-State Circuits Conference, ISSCC 2018, San Francisco, CA, USA, February 11-15, 2018*. IEEE, 2018, pp. 494–496.

[97] X. Si, W. Khwa, J. Chen, J. Li, X. Sun, R. Liu, S. Yu, H. Yamauchi, Q. Li, and M. Chang, "A dual-split 6T SRAM-based computing-in-memory unit-macro with fully parallel product-sum operation for binarized DNN edge processors," *IEEE Trans. on Circuits and Systems*, vol. 66-I, no. 11, pp. 4172–4185, 2019.

[98] M. Kang, S. Lim, S. K. Gonugondla, and N. R. Shanbhag, "An in-memory VLSI architecture for convolutional neural networks," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 3, pp. 494–505, 2018.

[99] 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," *J. Solid-State Circuits*, vol. 54, no. 6, pp. 1789–1799, 2019.

[100] S. Yin, Z. Jiang, M. Kim, T. Gupta, M. Seok, and J. Seo, "Vesti: Energy-efficient in-memory computing accelerator for deep neural networks," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 28, no. 1, pp. 48–61, 2020.

[101] G. Gambardella, J. Kappauf, M. Blott, C. Doehring, M. Kumm, P. Zipf, and K. A. Vissers, "Efficient error-tolerant quantized neural network accelerators," in *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, DFT 2019, Noordwijk, Netherlands, October 2-4, 2019*. IEEE, 2019, pp. 1–6.

[102] G. W. Burr, R. M. Shelby, S. Sidler, C. di Nolfo, J. Jang, I. Boybat, R. S. Shenoy, P. Narayanan, K. Virwani, E. U. Giacometti, B. N. Kurdi, and H. Hwang, "Experimental demonstration and tolerancing of a large-scale neural network (165,000 synapses) using phase-change memory as the synaptic weight element," *IEEE Transactions on Electron Devices*, vol. 62, no. 11, pp. 3498–3507, Nov 2015.

[103] S. Kim, M. Ishii, S. Lewis, T. Perri, M. BrightSky, W. Kim, R. Jordan, G. W. Burr, N. Sosa, A. Ray, J. . Han, C. Miller, K. Hosokawa, and C. Lam, "NVM neuromorphic core with 64k-cell (256-by-256) phase change memory synaptic array with on-chip neuron circuits for continuous in-situ learning," in *2015 IEEE International Electron Devices Meeting (IEDM)*, Dec 2015, pp. 17.1.1–17.1.4.

[104] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "PRIME: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *43rd ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2016, Seoul, South Korea, June 18-22, 2016*. IEEE Computer Society, 2016, pp. 27–39.

[105] F. Parveen, Z. He, S. Angizi, and D. Fan, "Hielm: Highly flexible in-memory computing using stt mram," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2018, pp. 361–366.

[106] M. Zabihi, Z. I. Chowdhury, Z. Zhao, U. R. Karpuzcu, J. Wang, and S. S. Sapatnekar, "In-memory processing on the spintronic CRAM: from hardware design to application mapping," *IEEE Trans. Computers*, vol. 68, no. 8, pp. 1159–1173, 2019.

[107] A. Chen and M. Lin, "Variability of resistive switching memories and its impact on crossbar array performance," in *2011 International Reliability Physics Symposium*, 2011, pp. MY.7.1–MY.7.4.

[108] G. Tayfun and Y. Vlasov, "Acceleration of deep neural network training with resistive cross-point devices," *CoRR*, vol. abs/1603.07341, 2016.

[109] C. Chen, M. Q. Le, and K. Y. Kim, "A low power 6-bit flash ADC with reference voltage and common-mode calibration," *J. Solid-State Circuits*, vol. 44, no. 4, pp. 1041–1046, 2009.

[110] J. C. Sancho and D. J. Kerbyson, "Analysis of double buffering on two different multicore architectures: Quad-core opteron and the cell-be," in *22nd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008, Miami, Florida USA, April 14-18, 2008*. IEEE, 2008, pp. 1–12.

[111] B. Liu, H. Li, Y. Chen, X. Li, Q. Wu, and T. Huang, "Vortex: variation-aware training for memristor x-bar," in *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*. ACM, 2015, pp. 15:1–15:6.

[112] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 2015, pp. 1117–1125.

[113] B. Moons and M. Verhelst, "A 0.3-2.6 TOPS/W precision-scalable processor for real-time large-scale convnets," in *2016 IEEE Symposium on VLSI Circuits, VLSIC 2016, Honolulu, HI, USA, June 15-17, 2016*. IEEE, 2016, pp. 1–2.

[114] S. Yin, S. K. Venkataramanaiah, G. K. Chen, R. Krishnamurthy, Y. Cao, C. Chakrabarti, and J. sun Seo, "Algorithm and hardware design of discrete-time spiking neural networks based on back propagation with binary activations," in *IEEE Biomedical Circuits and Systems Conference (BioCAS)*, October 2017.

[115] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch, C. di Nolfo, P. Datta, A. Amir, B. Taba, M. D. Flickner, and D. S. Modha, "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proceedings of the National Academy of Sciences*, vol. 113, no. 41, pp. 11 441–11 446, 2016.

[116] N. Chatterjee, M. O'Connor, D. Lee, D. R. Johnson, S. W. Keckler, M. Rhu, and W. J. Dally, "Architecting an energy-efficient DRAM system for GPUs," in *2017 IEEE International Symposium on High Performance Computer Architecture, HPCA 2017, Austin, TX, USA, February 4-8, 2017*. IEEE Computer Society, 2017, pp. 73–84.

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

[118] X. Si, J. Chen, Y. Tu, W. Huang, J. Wang, Y. Chiu, W. Wei, S. Wu, X. Sun, R. Liu, S. Yu, R. Liu, C. Hsieh, K. Tang, Q. Li, and M. Chang, "A twin-8T SRAM computation-in-memory unit-macro for multibit CNN-based AI edge processors," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 1, pp. 189–202, 2020.

[119] R. Guo, Y. Liu, S. Zheng, S. Wu, P. Ouyang, W. Khwa, X. Chen, J. Chen, X. Li, L. Liu, M. Chang, S. Wei, and S. Yin, "A 5.1pJ/neuron 127.3μs/inference rnn-based speech recognition processor using 16 computing-in-memory SRAM macros in 65nm CMOS," in *2019 Symposium on VLSI Circuits*, 2019, pp. C120–C121.

[120] J. Yue, Z. Yuan, X. Feng, Y. He, Z. Zhang, X. Si, R. Liu, M. 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.

[121] 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.

[122] J. Choi, S. Venkataramani, V. V. Srinivasan, K. Gopalakrishnan, Z. Wang, and P. Chuang, "Accurate and efficient 2-bit quantized neural networks," *Proceedings of Machine Learning and Systems*, vol. 1, 2019.

[123] B. Moons *et al.*, "ENVISION: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm FDSOI," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2017.

[124] J. Lee *et al.*, "UNPU: A 50.6TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *IEEE International Solid State Circuits Conference (ISSCC)*, 2018.

[125] B. Fleischer *et al.*, "A scalable multi-TeraOPS deep learning processor core for AI training and inference," in *Symposium on VLSI Circuits*, 2018.

[126] C. Kim *et al.*, "A 2.1TFLOPS/W mobile deep RL accelerator with transposable PE array and experience compression," in *IEEE International Solid State Circuits Conference (ISSCC)*, 2019.

[127] J. Lee *et al.*, "LNPU: A 25.3TFLOPS/W sparse deep-neural-network learning processor with fine-grained mixed precision of FP8-FP16," in *IEEE International Solid State Circuits Conference (ISSCC)*, 2019.

[128] S. Gupta *et al.*, "Deep learning with limited numerical precision," in *International Conference on Machine Learning (ICML)*, 2015.

[129] Y. Lecun *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, Nov. 1998.

[130] Q. Shang *et al.*, "Single-port SRAM-based transpose memory with diagonal data mapping for large size 2-D DCT/IDCT," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 22, no. 11, Nov. 2014.