Algorithm and Hardware Co-Design for Local/Edge Computing

Zhewei Jiang

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2020

Abstract

Algorithm and Hardware Co-Design for Local/Edge Computing

Zhewei Jiang

Advances in VLSI manufacturing and design technology over the decades have created many computing paradigms for disparate computing needs. With concerns for transmission cost, security, latency of centralized computing, edge/local computing are increasingly prevalent in the faster growing sectors like Internet-of-Things (IoT) and other sectors that require energy/connectivity-autonomous systems such as biomedical and industrial applications.

Energy and power efficient are the main design constraints in local and edge computing. While there exists a wide range of low power design techniques, they are often underutilized in custom circuit designs as the algorithms are developed independent of the hardware. Such compartmentalized design approach fails to take advantage of the many compatible algorithmic and hardware techniques that can improve the efficiency of the entire system. Algorithm hardware co-design is to explore the design space with whole stack awareness.

The main goal of the algorithm hardware co-design methodology is the enablement and improvement of small form factor edge and local VLSI systems operating under strict constraints of area and energy efficiency. This thesis presents selected works of application specific digital and mixed-signal integrated circuit designs. The application space ranges from implantable biomedical devices to edge machine learning acceleration.

Table of Contents

List of Tables v

List of Figures vi

Acknowledgments x

Introduction 1

Chapter 1: A Low Power Unsupervised Spike Sorting Accelerator Insensitive to Clustering Initialization in Sub-optimal Feature Space 5

1.1 Motivation 5

1.2 Algorithm 8

1.2.1 Dataset 8

1.2.2 Informative Sample Screening 10

1.2.3 Density-based Centroid Seeding 12

1.2.4 Centroid based Clustering 15

1.3 Accelerator Architecture 15

1.4 VLSI Implementation 17

1.5 Summary 20

Chapter 2: 1.74-μW/ch, 95.3%-Accurate Spike-Sorting Hardware based on Bayesian Decision 21

2.1 Motivation 21

2.2 Algorithm and Implementation 22

2.2.1 Feature Space Distribution 23

2.2.2 Cluster Validness Check 23

2.2.3 Decision-Tree-like Sorting 24

2.3 Measurements and Comparisons 26

Chapter 3: A sub-microwatt 96-Channel Neural Spike Processor for a Movement-Intention-Decoding Brain-Computer-Interface Implant 30

3.1 Motivation 30

3.2 System Overview 31

3.3 Spike Sorting 34

3.3.1 Supervised Training of the Constrained Bayesian Boundary Sorting 34

3.3.2 Accuracy and Cost Evaluations 37

3.4 Intention Decoding 38

3.4.1 Ensemble Observation Kalman Filter 38

3.4.2 Theoretical Basis 42

3.4.3 EOKF Evaluation 43

3.5 NSP Processor Architecture 45

3.6 Prototype Measurement 48

3.7 Summary 50

Chapter 4: XNOR-SRAM: In-Memory Computing SRAM Macro for Binary/Ternary Deep Neural Networks 52

4.1 Motivation 52

4.2 IMC OVERVIEW 53
4.2.1 Multi-bit weights in IMC designs 53
4.2.2 Binary weights in IMC design 53
4.2.3 Multi-bit activation in IMC designs 54
4.3 XNOR-SRAM Macro Design and Optimization 54
4.3.1 XNOR-SRAM Bitcell Design 55
4.3.2 XNOR-SRAM Operation and Analysis 58
4.3.3 Transfer Function and ADC Optimization 60
4.4 Measurement Results 62
4.4.1 Energy Consumption and Performance Measurements 62
4.4.2 Variability and Compensation 63
4.4.3 The Statistical Model of XNOR-SRAM and Voltage Scaling 65
4.4.4 Strategy for Mapping DNNs onto XNOR-SRAM arrays 68
4.4.5 DNN Accuracy Characterization 68
4.5 Summary 72

Chapter 5: C3SRAM: An In-Memory-Computing SRAM Macro Based on Robust Capacitive Coupling Computing Mechanism 74
5.1 Motivation 74
5.2 Architecture and Operation 75
5.2.1 Memory Array Operation 75
5.2.2 ADC Operation 81
5.2.3 Signal Switching Order 83
5.3 Algorithm Hardware Specification 84

5.3.1 Activation Bit Precision 84
5.3.2 Partial Convolution Quantization Levels 86
5.4 Measurement and Analysis 87
5.4.1 Energy and Throughput 87
5.4.2 Transfer Function 89
5.4.3 Variability Measurement 92
5.4.4 Evaluation on Neural Network Tasks 95
5.5 Summary 96
Conclusion 98
References 106

List of Tables

1.1 Performance comparisons. 20

3.1 Spike sorting accuracy. 37

3.2 Comparison of number of calculations in EOKF and standard KF (20-neuron cortical map). 42

3.3 Comparison to prior BCI processors. 51

4.1 V_{RBL} variance of a single column at 1.0V and 0.6V supply extracted from post-layout Monte-Carlo simulations. 64

4.2 Measured MLP (for MNIST) and CNN (for CIFAR-10) accuracy summary using XNOR-SRAM at 0.6V supply. 73

5.1 Comparison to prior IMC works. 91

5.2 Measured MLP (for MNIST) and CNN (for CIFAR-10) accuracy summary using C3SRAM. 96

List of Figures

1.1 A general brain-computer interface operation flowchart. 6

1.2 Algorithm flow chart. The black, red, blue, and green boxes represent steps that are common operation and scheduling, phase 1, phase 2, and phase 3, respectively. . 9

1.3 (a) Histogram of spike peak amplitudes; (b) log scale contour plot of spike density in feature space, peak/trough amplitudes as X/Y axes; (c) histogram of trough amplitude. 11

1.4 The error in feature distribution decreases as more spikes are collected. 13

1.5 The centroids convergence is shown to be more reliable in the algorithm with screening procedure (a) than without the procedure (b). 14

1.6 Spike sorting accuracy at various initialization schemes. 16

1.7 Block Diagram of the accelerator. 16

1.8 The layout of the accelerator in a 65nm CMOS. The footprint is 226x226 μm^2. . . . 18

1.9 Accelerator frequency based on critical path delay at various VDD levels. 19

1.10 (a) Power breakdown by type; (b) leakage power breakdown by source. 19

2.1 Spike sorting accelerator block diagram. 23

2.2 Index Pair lookup for peak feature. 24

2.3 CAM architecture. 25

2.4 Adjacency Checker. 25

2.5 Module power gating grouping. 26

2.6 Power gating scheme. 27

2.7 Balloon latch schematics. 27

2.8 Balloon latch operation flow. 28

2.9 Die photo. 28

2.10 Sorting accuracy comparison. 29

2.11 Power vs. spiking rate. 29

3.1 Proposed prosthetic BCI task flow. 32

3.2 Visualization of spike sorting via Bayesian approximate decision tree. 35

3.3 Feature space segmented block identification coding. 36

3.4 Instantaneous spiking rates weighted in ensemble average for the observation based estimate of a kinematic state. 40

3.5 Multivariat regression of ensemble neurons reduce the effect of noise. 44

3.6 State prediction error of ensemble prediction (left) and single neuron prediction (right) across angles and speed. 45

3.7 The architecture of the proposed NSP. 46

3.8 Conveyor style queue of a 4-detector example. 47

3.9 Die photo and area breakdown. 48

3.10 Spike rate dependency of the NSP power. 49

3.11 Power and performance of the NSP. 50

4.1 (a) XAC operation illustration and (b) the proposed XNOR-SRAM macro architecture. 56

4.2 XNOR-SRAM bitcell design and XNOR-ACC operation with ternary inputs/activations and binary weights. Bitwise ternary-XNOR output from each bitcell forms pull-up/-down paths on the RBL voltage, which represents the XAC value. 57

4.3 PU/PD paths for V_{RBL} with binary activations. 59

4.4 PU/PD paths for V_{RBL} with ternary activations. 59

4.5 XAC is mapped to V_{RBL}. The confined linear quantization scheme is shown, along with the corresponding 10 reference voltages for the 11-level flash ADC. 61

4.6 The accuracy of the MLP trained for MNIST and that of the CNN for CIFAR-10 as a function of ADC levels. 62

4.7 (a) 65nm XNOR-SRAM prototype chip micrograph. (b) Power and area breakdown. 63

4.8 Data dependent XNOR-SRAM power. 64

4.9 Energy and frequency scaling with supply voltage. 65

4.10 Energy and delay comparison with digital baseline. 66

4.11 Measured transfer function and variability. 67

4.12 Body bias tuning for PMOS/NMOS mismatch. 68

4.13 Measured ADC output probability distribution as a function of XAC value at V_{DD} of 1.0V, 0.8V, 0.6V, and 0.5V. 69

4.14 Normalized transfer function at different V_{DD}. 70

4.15 Mapping convolutional neural networks to XNOR-SRAM-based in memory computing. 71

4.16 Measurement based simulation framework for CIFAR-10 accuracy evaluation using XNOR-SRAM macros. 72

5.1 Architecture of C3SRAM in-memory computing macro. 76

5.2 C3SRAM bitcell design and in-cell bMAC operand table. 77

5.3 Threshold voltage variability effects on charged capacitor voltage. 78

5.4 Capacitive coupling based in-memory computation of bMAC. 80

5.5 (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. 81

5.6 Operation of the double-sampling self-calibrating single-ended comparator. 82

5.7 Signal transition order for reducing analog non-idealities. 84

5.8 MLP on MNIST dataset inference accuracy losses at various levels of activation precisions. 85

5.9 MNIST and CIFAR-10 inference accuracies increase as quantization resolution of pre-activation partial sum (256 input) increases. 87

5.10 The bMAC distribution of MLP for MNIST and quantization in limited ADC range. 88

5.11 C3SRAM prototype chip micrograph. 89

5.12 C3SRAM energy and delay comparison with XNOR-SRAM and digital ASIC with traditional SRAM and ALU. 90

5.13 Left: measured power consumption breakdown between the three supplies powering bMAC compute (blue), partial sum accumulation (red), ADC (green). Right: area breakdown of the C3SRAM module. 90

5.14 Measured V_{MBL} transfer curve shows lower FSR from ideal curve due to charge sharing with ADC input capacitors. 92

5.15 (a) ADC output offset due to comparator gain stage mismatch, (b) V_{MBL} error variation measurement, (c) RMS error of the macro. 94

5.16 ADC power increases exponentially as ADC power supply increases; the trip point variation increases linearly. 95

Acknowledgements

I would like to thank my academic advisor, Prof. Mingoo Seok, for his assistance, support, and encouragement throughout my Ph.D. career at Columbia University. He set high standard in research, lead a great environment at the VLSI Lab, and has been a reliable source of technical guidance. I would like to thank Prof. Jae-sun Seo of Arizona State University, who I have collaborated with and has guided and shaped my research over the years. Many thanks to the other committee members of my thesis proposal and defense, Prof. Martha Kim, Prof. Dion Khodagholy, Prof. Xiaofan (Fred) Jiang, and Prof Charles Zukowski. They have graciously shared their valuable time with me and I greatly appreciate it.

I would like to thank the many outstanding people of VLSI Lab I've worked and collaborated with throughout my graduate career. Many thanks to Seongjong (Josh) Kim, Jiangyi Li, Joao Cerqueira, Minhao Yang, Pavan Chundi, and Sung Justin Kim. Beyond the VLSI lab, I'd like to thank Shihui Yin of Arizona State University who I have collaborated with for the last three years. Shihui's knowledge and meticulousness is indispensable in our work together. I am also grateful to my colleagues who I have learned from and enjoyed the company of. Many thanks to Teng Yang, Doyun Kim, Wei Jin, Tianchan Guan, Dongkwun Kim, Weiwei Shan, Bo Zhang, Daniel Kim, Ashish Shukla, and Peiye Liu.

I would also like to thank Prof. Richard Garner, Prof. Sean Bentley, and Prof. Gregory Mercurio of Adelphi University for not only shaping my academic career but also for their personal friendships.

Beyond academia, I would like to thank Dr. Vivek Joshi, Dr. 'Rumi' Muhammed Ahosan

Ul Karim, Dr. Xi Cao of GlobalFoundries for their guidance and mentorship during my internship at the Memory Solution Group. They imparted perspectives, insights, and advises to me that was and will continue to be valuable to my research career for my Ph.D. and beyond.

All of this was only made possible by the continued support by my mother, Fang Zhang. I am thankful for her love and support.

Zhewei Jiang

April 2020

Introduction

VLSI technology over the decades have created many computing paradigms for disparate computing needs in power, form factor, deployment environment, application, etc. In addition to the traditional pursuit of large scale and centralized computing, edge computing have increased in prominence in recent times, driven largely by the fast growing Internet-of-Things (IoT) in the consumer electronics sector. Similarly, local computing saw the same technological enablement in fields that demand energy or connectivity autonomy such as in biomedical and industrial applications.

Local and edge computing often operate under stringent resource constraints, the chief of which is power or energy efficiency. For many of these applications, the computing tasks and their associated parameters are limited in a way that embedding general purpose computing devices into such systems is inefficient and potentially infeasible. Such systems would benefit by unitizing application specific hardware, i.e. custom circuit that performs only the desired algorithms. Compartmentalized design methodology for algorithm and hardware takes system agnostic approach and develop algorithm independent of hardware, and to a less extent vice versa. Algorithm/hardware co-design is the design methodology that explores the large design space from algorithm and hardware perspectives simultaneously. Design efforts across multiple design stacks are often iterative, co-design methodologies are efficiency tools developed to address that, aiming to provide higher performance gain per design effort. Algorithm/hardware co-design has functionality and non-functionality specifications. Functionality spec simply refers to that the algorithm must be executable in the hardware. This is an open-ended problem with large design space. The non-

functionality specs refer to overall system performance like speed, power, and area. The co-design efforts are largely put toward the compliance of non-functionality specs.

At the most fundamental level, algorithm/hardware co-design is the optimization of an algorithm model that reflects hardware and system architectural behavior, with the inclusion of scheduling, data traffic model, and linkage between operators in the algorithm stack and processing element (PE) in the hardware stack.

The optimization method used in the presented projects is constraint allocation and partitioning. The process is as follows. First identify the baseline algorithm and the main constraints in the specification; then model the algorithm operators with the associated PE cost in the constraint domain; then experiment with applicable hardware techniques to allow re-allocation of constraint domain budget to in turn enable further modifications to PE/operator, scheduling, architecture; do so iteratively until design is optimized.

In the following chapter, 5 projects are presented, detailing designs achieved via constraint allocation based algorithm/hardware co-design.

In the first three chapters, the designs tackle implantable brain computer interface (BCI). For implantable hardware, the power and energy is the main design constraint, limited by battery life, charging mechanism, and most importantly the biological environment it interacts with. The projects make use of existing algorithms that were hardware-agnostic as they were not designed for implantable devices. Hence, there are many vectors for major hardware-aware augmentations through constraint allocation and partitioning.

Chapter 1 tackles the spike sorting problem for hardware with a power budget fit for implant. The baseline algorithm selected is the sequential leader algorithm. As with many clustering algorithms, there's an initialization challenge which can leads to high transient stage storage and computation cost. To re-allocate the resources, the transient memory units and high per-input runtime are targeted for optimization. The design implements a modified leader algorithm with an added screening stage. The screening stage samples the input data feature space and only performs cluster training using informative inputs, drastically reducing the memory and computations

necessary for centroid convergence.

Chapter 2 presents another spike sorting hardware. This project can be considered another iteration of the Chapter 1 design. By experimenting with power-gating memory elements, the sorting algorithm's constraint model is changed accordingly to trigger a constraint re-allocation. Schedule-based algorithm modifications is the new design goal. Given the algorithm/hardware for input feature distribution from the previous design iteration, a Bayesian boundary based sorting algorithm is devised to achieve improvements in energy and power by minimizing the per-input computing run-time.

Chapter 3 presents an end-to-end BCI system inclusive a Bayesian boundary based spike sorter from Chapter 2 and a partial neural intention decoding hardware. The baseline algorithm for neural decoding is the Kalman Filter. There are several vectors for modifications as there are on-chip/off-chip constraints in this project, giving it more freedom in constraint re-allocation. The major resource-consuming elements of the implant are the multiplier PE, wireless communication, and data traffic jam. In experimenting with a novel modified form of Kalman Filter, the algorithm allows data stream partition to reduce earlier than the baseline version, moving majority of high cost computations off-chip without incurring any on-chip cost, this in turn enables the replacement of multiplier PE with simpler adder, exploiting the discrete input of the on-chip portion of the neural decoding, which in turn solves the data traffic jam issue through the event-driven adder staggering the computations.

In the later two chapters, the in-memory-computing macros are designed to serve as components for larger ASIP systems handling the more flexible application of machine learning. In this fast growing sector, the energy and power limitation is imposed by the scaling of the already power hungry computations and memory wall. The co-design optimization goal is directed at the main computing primitive of machine learning tasks.

Chapter 4 presents a mixed-signal in-memory-computing macro which is capable of performing binary multiply-and-accumulate operation on the entire content of a memory module without explicit access. Chapter 5 presents another mixed-signal in-memory-computing macro which op-

erate using a more robust and power efficient computing mechanism of capacitive coupling.

Chapter 1: A Low Power Unsupervised Spike Sorting Accelerator Insensitive to Clustering Initialization in Sub-optimal Feature Space

Online unsupervised spike sorting or clustering is an integral component of implantable closed-loop brain-computer-interface (BCI) systems. Robust clustering performance against various non-idealities such as poor initialization and order-of-arrival of inputs are desirable while meeting the minimal area and power requirements for implants. We explore an online and unsupervised spike-sorting algorithm utilizing a low-overhead feature screening process that improves feature discriminability in the use of sub-optimal features for reducing hardware complexity. Based on the algorithm, an accelerator architecture that performs feature screening and clustering is devised and implemented in a 65-nm high-V_{TH} CMOS, largely improving clustering accuracy even with poor clustering initialization. In the post-layout static timing and power simulation, the power consumption and the area of the accelerator are found to be 2.17 μW/ch and 0.052 μm^2/ch, respectively, which are 53% and 25% smaller than the previous designs, while achieving the required throughput of 420 sorting/s at the supply voltage of 300mV.

1.1 Motivation

Advances in the BCI research have shown promising results in prosthetic, clinical, and other applications in laboratory settings [1, 2, 3]. In order to improve the practicality of these platforms, it is critical to develop a closed-loop BCI system which can be entirely integrated on a chip, and can autonomously operate with minimal need for user interference, thereby necessitates all processes to execute in a real-time unsupervised manner.

Figure 1.1 illustrates the operation flow of a typical invasive BCI system. The front-end begins with an implanted electrode array that probes the extracellular action potentials (i.e. multi-unit

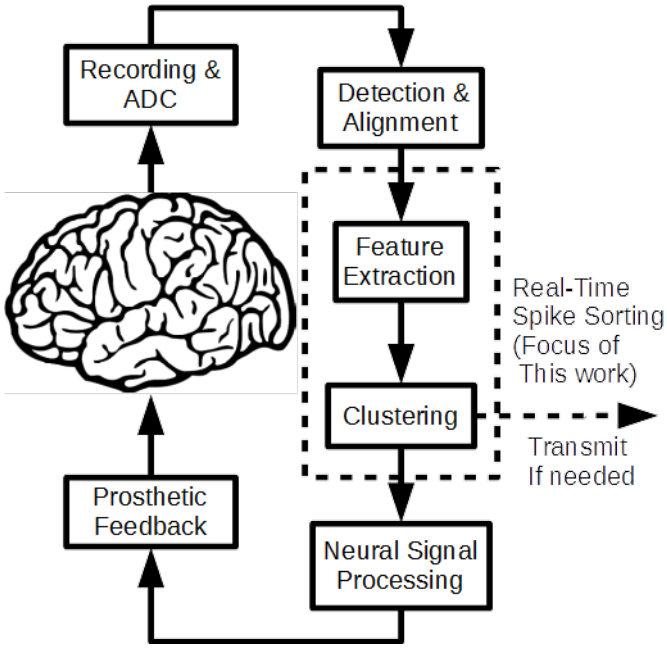


Figure 1.1: A general brain-computer interface operation flowchart.

activities) near the target neurons. The analog waveforms are then digitized, followed by spike detection and alignment. After this, features of spikes are extracted for reducing the data dimensionality, and discriminable values in a chosen feature space are computed for clustering purpose. Then, each spike is assigned to a cluster by its waveform. Sorted spikes can be used in various ways depending on the application; the cluster indexes can be transmitted, or used to compute spike rates of targeted neurons which in turn can be inputs for various neural signal processing (e.g. regressions) to derive neural models for prosthesis, or, in a trained system, inputs to be translated into neuro-prosthetic intents and commands.

Feature-extraction and clustering are integral parts that are desirable to be performed in an unsupervised manner. Performing supervised training after placing an implant on a patient can require a significant amount of characterization efforts for creating and training with a patient specific dataset. In addition, the characteristics of spikes that an electrode array captures can vary over time, which may require regular expensive maintenance for re-training in case of long-term implants.

It is also critical to achieve online operation. The wireless communication between implants

and any external devices is expensive in power consumption. If all the detected spikes need to be wirelessly transmitted for offline feature-extraction and clustering, the power can be prohibitive [4, 5]. Online feature extraction and clustering have been shown to significantly reduce the required data rate for wireless communication, resulting in large energy savings in the system level [6].

Existing methods of feature extraction range from computationally intensive methods, e.g. principle component analysis (PCA), discrete wavelet transform (DWT), Bayesian algorithm, to computationally efficient ones which often extract time-domain features, e.g. peak-to-peak amplitude, spike width, zero-crossing feature, spike energy. Computationally intensive features like principle components and wavelet coefficients provide better inter-cluster discriminability but their power consumption and area, when implemented in a VLSI circuit form, can be prohibitive for resource-constrained implants.

On the other hand, time-domain waveform characteristics are intuitive to use and also inexpensive to extract but are not optimal in differentiating spike shapes [7]. Low discriminability negatively impacts clustering reliability by accentuating inherent non-idealities in many cluster algorithms such as sensitivity to initialization, input order-of-arrival, and disparate cluster membership. However, if we can mitigate those problems, i.e., improving clustering performance in sub-optimal feature space, the low complexity can help to significantly scale down power consumption and hardware footprint.

Therefore, we focus on developing a low-complexity yet robust multi-phase algorithm using sub-optimal features, and its efficient mapping onto VLSI accelerator architecture, for unsupervised and online feature extraction and clustering. The algorithm integrates three sub-algorithms, namely informative sample screening, density-based centroid seeding, and centroid-based clustering. The VLSI accelerator architecture of the algorithm is designed and implemented in a 65nm high threshold-voltage (V_{TH}) CMOS. The post-layout static timing and power simulation shows that the accelerator consumes 53% less power (2.17 μW/ch) and 25% smaller silicon footprint (0.05μm^2/ch), as compared to the previous work [8], while easily meeting the required throughput (420 clustering/s) at the supply voltage (VDD) of 300 mV and the clock frequency of 140 kHz.

The chapter is organized as follows: Section 1.2 introduces the proposed algorithm; Section 1.3 details the VLSI architecture; Section 1.4 presents the circuit implementation, the delay and power consumption and the design, and the comparisons to the previous works; and Section 1.5 summarizes the chapter.

1.2 Algorithm

This section explores a 3-phase unsupervised online spike sorting algorithm for improving clustering performance at low hardware complexity using sub-optimal features. Figure 1.2 shows the flow chart for the entire algorithm. The overview of the algorithm is: phase 1 informatively screens incoming samples based on a density-based metric, which is computed through constructing distribution histograms on feature space axes. The first phase concludes when a statistically significant amount of data is collected to faithfully represent the density distribution of the dataset. Phase 2 uses those selected informative samples to create centroids (i.e., centroid seeding). Cluster centroids are initialized and updated at the locally densest areas in the feature space. The second phase concludes when cluster centroids converge. Finally, in phase 3, spikes are assigned a cluster index via centroid-based clustering. No feature extraction is performed.

1.2.1 Dataset

The algorithm and accelerator architecture is designed based on the dataset of single channel extracellular neuronal signals which were recorded in the sensory thalamus of vibrissa pathway in anesthetized rats. Neuronal signals were amplified, band-pass filtered (300–5 kHz), digitized in 8-bit resolution at 40 kHz/channel and collected using a 32-channel data-acquisition system (Plexon, Dallas, Tx). The dataset contains a total of 40k spikes originating from 4 neurons and has a signal-to-noise ratio of 2.9dB. A single spike consists of 32 samples. We also use the two other datasets which contain 40k spikes from 2 and 3 neurons, respectively, but mostly use the first dataset since it gives the lowest clustering success rate across all the experiments that we perform.

Detail procedures for creating the datasets were described previously [9, 10], approved by the

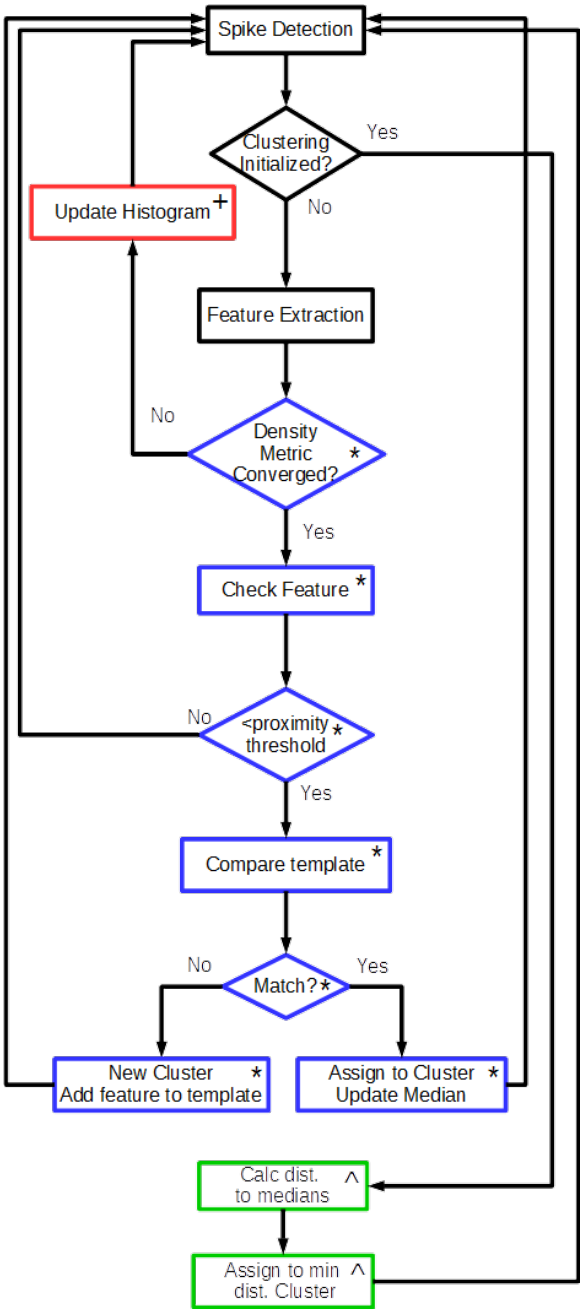


Figure 1.2: Algorithm flow chart. The black, red, blue, and green boxes represent steps that are common operation and scheduling, phase 1, phase 2, and phase 3, respectively.

Institutional Animal Care and Use Committee at Columbia University. Briefly, female rats (225 –300 g; Sprague Dawley) were sedated with 2% vaporized isoflurane and anesthetized with sodium pentobarbital (50 mg/kg, i.p., initial dose). Body temperature of the animals was maintained at 37 °C by a servocontrolled heating blanket (FHC, Bowdoinham, ME). After initial anesthesia, the animal was mounted on a stereotactic device (RWD Life Science, China) in preparation for the surgery and subsequent recordings. After the initial midline incision on the head, a small craniotomy (1 mm X 1 mm) was made on the left hemisphere over the ventroposterior medial nucleus (VPm) of the thalamus (2.0–4.0 mm caudal to the bregma, 2.0–3.5 mm lateral to the midline). A tungsten microelectrode was slowly advanced into VPm using a hydraulic micropositioner (Kopf Instruments, Tujunga, CA).

1.2.2 Informative Sample Screening

Initialization sensitivity is a critical problem for centroid-based clustering such as k-means since they are heuristic [11] such that random initialization schemes can lead to drastically different results on the same data. Furthermore, fully implemented distance minimization is NP hard [12], hence reliable initialization are essential for enabling online unsupervised spike sorting.

For this purpose, we propose to perform informative sample screening in phase 1. The screening starts with establishing the knowledge of feature distributions. Specifically, using the early set of incoming spikes, histograms for features are generated, which are then used to determine if spikes are informative or not. Those informative spikes are passed to the phase 2 for the density-based cluster seeding, which serve as a critical foundation for the low-complexity centroid-based clustering in phase 3.

We select peak/trough amplitudes as spike features which are simple to extract but have large noise component in the feature space. Figure 1.3 (b) contour plot shows one dataset's feature space distribution. The features have multimodal distributions when projected onto feature space axes. As illustrated in the density histograms in Figure 1.3 (a) and (c), the low SNR of the dataset, however, contributes to the high variance in the distinguishable modes. High overlaps in modal

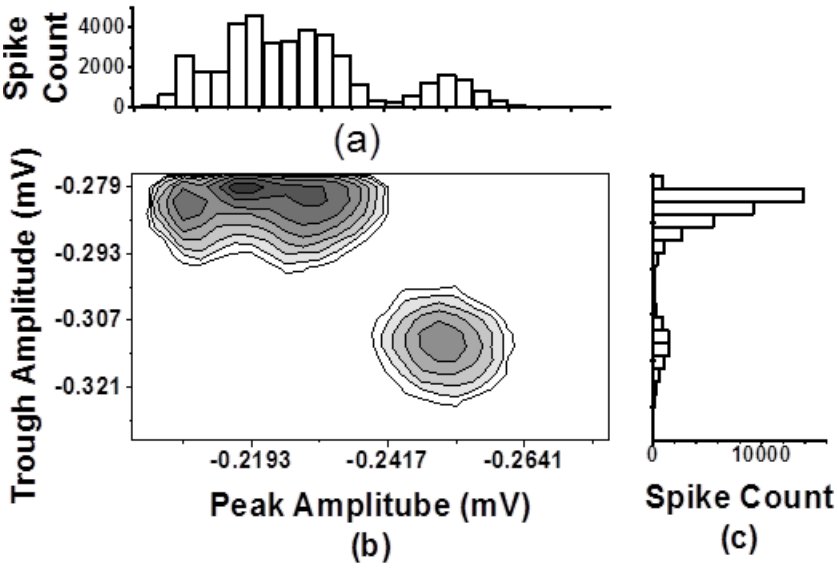


Figure 1.3: (a) Histogram of spike peak amplitudes; (b) log scale contour plot of spike density in feature space, peak/trough amplitudes as X/Y axes; (c) histogram of trough amplitude.

distribution represent poor inter-cluster separation. This makes the dataset ideal for evaluation of the proposed screening algorithm.

The distribution histograms of the peak and trough amplitudes are used as the density metric, devised with low hardware overhead by projecting each data points' amplitude features onto the axes which are segmented into bins with width ϵ. The bins' width ϵ is selected to match the proximity parameter of the density-based cluster seeding in phase 2. (Subsection 1.2.2 for details). The density is represented by the count of spikes in the bin and for each input, the peak and trough values are divided by bin size ϵ and used as index to access and increment the density values (eq. 1.1 and 1.2). The screening rule relies on the multimodal density distributions in the feature space having distinguishable peaks in at least one dimension.

$$Density(\frac{max(spk_{in})}{\epsilon}) + + \tag{1.1}$$

$$Density(\frac{min(spk_{in})}{\epsilon}) + + \tag{1.2}$$

$$error_n = 20log\frac{rms(D_n - D_{final})}{rms(D_{final})} \tag{1.3}$$

If spikes have both peak and trough amplitudes that belong to target bins, they are chosen as informative samples. Target bins are bins that have local maximal density which corresponds to the peak of mode in the histogram. Fig. 3(b) shows that contours of highest density are located at coordinates where the bins are local maxima. The data shown in the Figure 1.3 is for the illustrative purpose. The full dataset has large ambiguity on border point assignment which might not be linearly separable.

The distribution histogram is found to be well converged, confirming the effectiveness of the informative sample screening. As shown in Figure 1.4 which shows the distribution error as more spikes are used to create the distribution, the error of the density metric reaches -60dB with normalized true density distribution at around 700 spikes. The error is defined as eq. 3. D_n and D_{final} are normalized distribution functions generated from n samples and the entire dataset respectively.

The approximate convergence period, i.e. the time to reach reliable density metric, is approximately 2 second based on the assumption maximum firing rate at 70 spikes per second from 6 neurons as specified by [8]. The 1-to-2 phase change can be scheduled by the use of a counter.

1.2.3 Density-based Centroid Seeding

In the second phase, the proposed algorithm performs density-based centroid seeding, which estimates centroids using the informative samples screened in the phase 1. These parameters are passed to the phase 3 for centroid based clustering.

The proposed seeding part of the algorithm performs as such: screening process looks up each incoming spikes' feature in the density histogram, if both peak and trough belongs in local maximally valued bins, the spike is an informative sample. If it is the first spike found in a particular

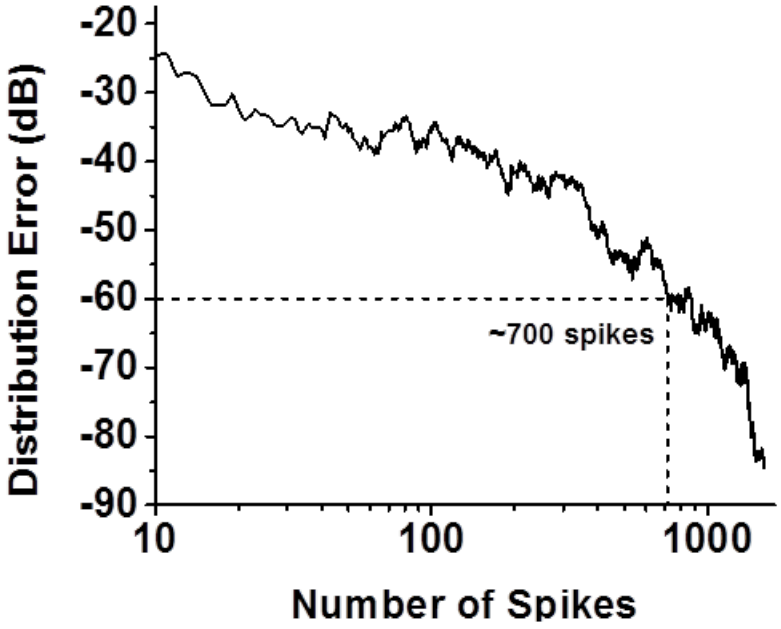


Figure 1.4: The error in feature distribution decreases as more spikes are collected.

intersection, the spike waveform (not its features) is saved as the centroid of the cluster. All subsequent spikes with features matching the same bins are used to update the existing cluster median. Spikes that do not pass screening are discarded. After a period of time, the centroid for each cluster converges. And the converged median waveforms are used as seeds for the centroid-based clustering in phase 3.

The proposed density-based centroid seeding is developed based on a reduced DBSCAN (density-based spatial clustering in applications with noise) with the efforts to reduce computational complexity as well as the number of user-setting parameters.

The standard DBSCAN is an offline algorithm with O(n log n) runtime, and requires two user setting parameters: proximity threshold ϵ and minPts, the minimum number of data points within ϵ-neighborhood to be considered density-reachable [13]. Setting appropriate minPts parameter requires prior knowledge of feature variance, which is non-trivial and costly to compute.

Density based informative sample selection allows the centroid seeding algorithm to execute without the parameter minPts. As mentioned previously, the proximity threshold ϵ matches the

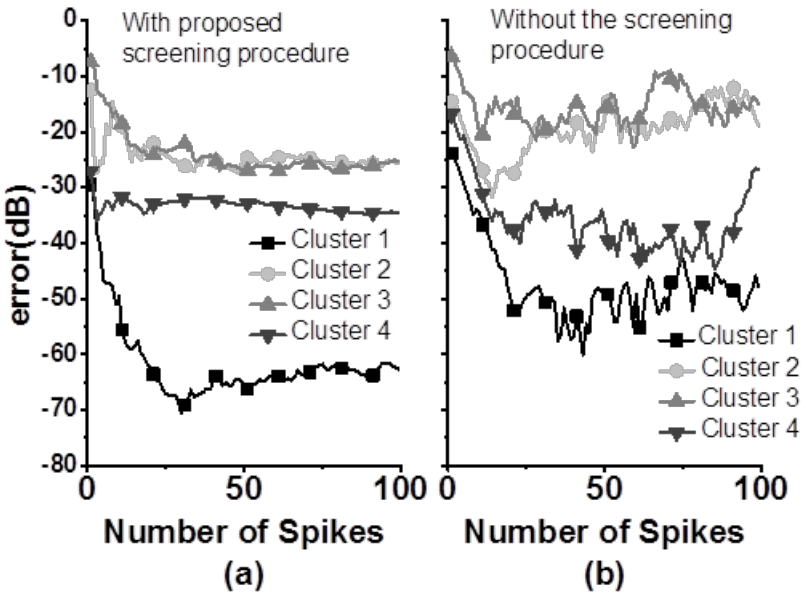


Figure 1.5: The centroids convergence is shown to be more reliable in the algorithm with screening procedure (a) than without the procedure (b).

bin width defined for the screening rule. As a result, all spikes in the same informative-sample-populated bins are of the same cluster, and no two such bins are adjacent in the feature space as they represent modal peaks in density distribution. Hence, there exists precisely one cluster centroid in or near each informative-samples-populated bin, provided that the features present distinguishable modes for each cluster.

In addition, the proposed algorithm has O(n) runtime. It does not need to visit all the points in its ϵ-neighborhood while DBSCAN (O(n log n) runtime) does. Rather, for each incoming spike that passes screening, it is immediately assigned to be a member of the cluster associated with the bins because the density-reachability is implicitly satisfied based on the distribution curve (modal peak).

Figure 1.5 illustrates the screening procedure's effect on centroid convergence error: (a) when using informative sample, (b) when using all incoming spikes. The convergence behavior of the proposed methods is shown to be more insensitive to initialization with on average 16dB lower error.

1.2.4 Centroid based Clustering

Phase 3 has a simple winner-take-all scheme where the input spike is assigned to the cluster with the least distance to centroid. This clustering algorithm uses the converged median waveforms (seeds) found in the phase 2. The number of the centroids is also provided by the phase 2. Given that all centroids are initialized at less than -20dB deviation from the true centroids (Figure 1.5(a)), minimization of intra-cluster distance is implicitly met by assigning each incoming spike to the closest centroid. The distance metric we use is the L1 metric. L1 distance covers the full dimensions of the data. As pointed out in the previous work on on-chip spike sorting ASIC based on the sequential leader algorithm [8], L1 metric can provide noise tolerance and lower hardware cost comparing to the Euclidean distance. The L1 metric can be computed as shown in eq. 1.4 :

$$d_{L1} = \sum_{n=1}^{N_{spk}} |spk_{in}(n) - spk_{centroid}(n)| \tag{1.4}$$

Finally, Figure 1.6 shows the performance of the algorithm. The first column shows an average of classification accurary of 57.3% with randomly initialized centroids, algorithm based on sequential leader (used in [8]) with predetermined target for cluster number; second column shows an average classification accuray of 92.2% with the best case initialization, i.e. the algorithm operates without informative sample screening just as that of the first column but is given an input order that initialized centroids near the true centroids; the third column shows an average of 98.6% accurary rate with the propose algorithm, demonstarting superior insensitivity to order-of-arrival and the highest success rate among three.

1.3 Accelerator Architecture

For the proposed algorithm, we devise the area and power efficient accelerator architecture (Figure 1.7). For phase 1, the features are used to aggregate the density metric, stored in a dedicated memory block (Memory: Density Metric in Figure 1.7). Memory address is feature values truncated to 6 bits MSB to represent ϵ value of 4, i.e. feature valued 128 increments density value

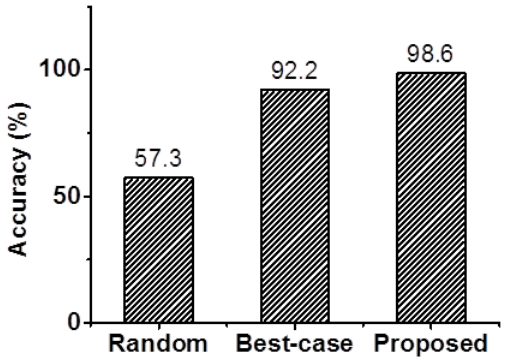


Figure 1.6: Spike sorting accuracy at various initialization schemes.

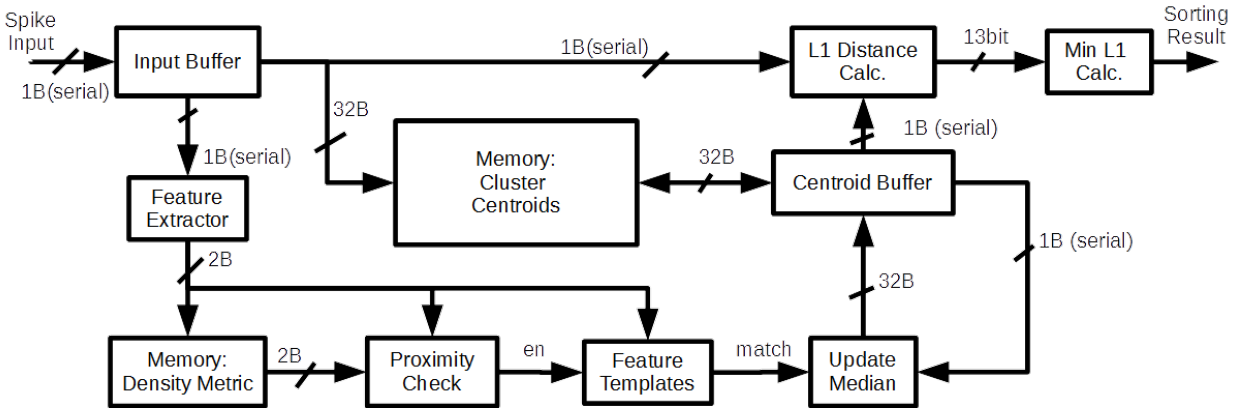


Figure 1.7: Block Diagram of the accelerator.

in bin 32. The bin size is set to 1/64 of feature range to exploit binary encoding.

In phase 2, the features are used to read from the density metric memory values of the bin and its neighbors. Proximity check block then determines whether the input is an informative sample. If informative, the features are looked up in template memory block; if present, the waveform is used to update the centroid saved in cluster centroid memory block, if not present, the features are saved in template memory and the waveform is saved in centroid memory. Computations are performed on the centroid buffer block to reduce read/write accesses.

In phase 3, L1 distance calculation block computes the distance between input and each centroid, and minimum L1 calculation block determines the cluster index the input belongs to. The index is the final output of the accelerator.

The main goal in the architecture design is to minimize power consumption and silicon footprint. For this, we ensure large memories in the accelerator are not directly clocked to lower switching power on the clock tree. Switching activities in memories are therefore limited to reading and writing access only. Given that the majority of the circuits are devoted to memories, it is extremely important to reduce their switching activities.

In the accelerator design, most of the circuit is idle (zero switching activities) at any clock cycle. As a result, leakage dominates the power consumption. We implemented all computational blocks serially in order to reduce leakage on the architecture level. Higher switching activity associated with serial design is insignificant in this accelerator implementation, with only schedule controller and clock network being affected. Serial implementation decreases the computational blocks' silicon footprint by a factor of 32, and also significantly reducing idle leakage.

In addition, we select the lowest operating frequency for low power consumption while still yielding the comparable throughput reported in [8] where a single channel DSP operating at 480 kHz can process 6 neurons firing at 70 spikes/s. Our design benefits from informed initialization to achieve the same throughput with 140 kHz clock, on account of lesser transient clusters.

1.4 VLSI Implementation

The accelerator is implemented in a 65nm CMOS. As previous works [8, 14] have identified leakage power as dominant in implantable chip power budget, the accelerator is synthesized with high-V_t devices for minimizing leakage. The low clock frequency requirement also allows us to use near/sub-threshold supply voltage to minimize power consumption. Therefore, in order to ensure the robustness of gates at the aggressively-scaled supply voltage, we only use a subset of the industrial standard cell library during the synthesis that are found to be robust at near/sub-threshold supply voltages. While low-voltage optimized SRAM can be used for reducing the footprint for various memories, we decide to use the latches in the standard cell library for high robustness.

The design is automatic-place-routed (APR-ed), and the layout view of the implemented accelerator is shown in Figure 1.8. This is a single channel spike sorter unit with no dependency on

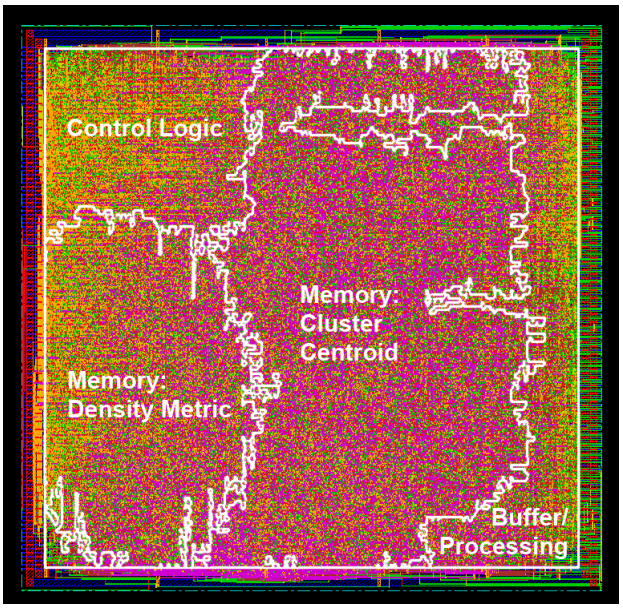


Figure 1.8: The layout of the accelerator in a 65nm CMOS. The footprint is 226x226 μm^2.

external computation blocks or parameter, with fixed front end configurations built-in. The area of the design is 0.052 mm^2.

We perform the static timing analysis using the libraries characterized at multiple supply voltages and the interconnect parasitic information generated by the APR tool. The accuracy of the timing and power characterization flow is calibrated and confirmed by comparing its result to the results of SPICE simulation for benchmark circuits.

As shown in Figure 1.9, at the supply voltage of 0.3V, the maximum clock frequency can be as high as 324 kHz, more than twice the required clock frequency of 140 kHz. At 0.3V, the power consumption is characterized to be 2.17µW.

As shown in Figure 1.10(a), power consumption breakdown by type indicate that the leakage power consists of 90% of all power consumptions. Leakage breakdown by module in (b) indicates that majority of the power consumed is still from memory blocks, with the centroid and histogram memories consuming 65% of the total power (47% and 18% respectively).

Finally, Table 1.1 details comparison with previous works in spike sorting DSP. To our best knowledge, the architecture in [8] is the only other design that implementes real-time unsupervised spike sorting. Our accelerator is a single channel classifier that is scaled linearly and inter-unit

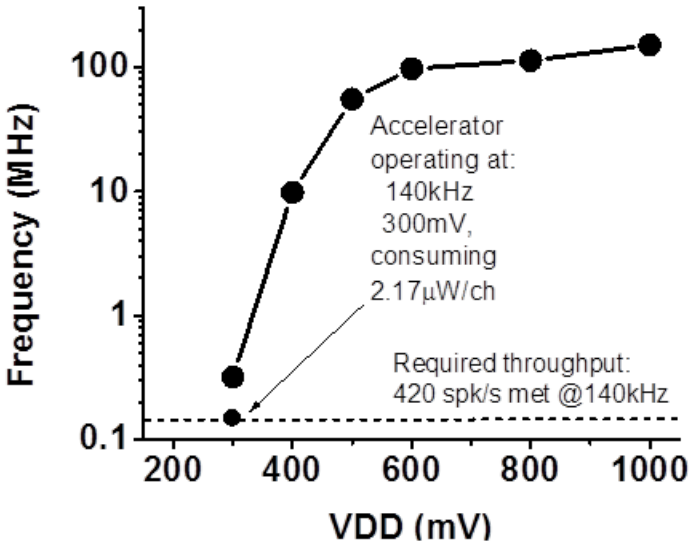


Figure 1.9: Accelerator frequency based on critical path delay at various VDD levels.

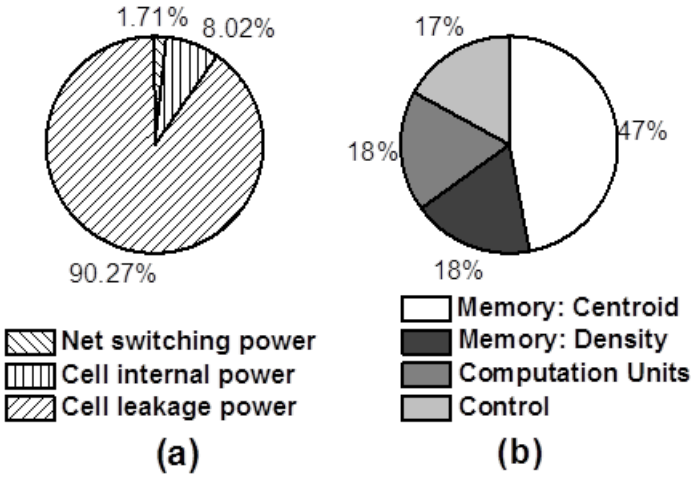


Figure 1.10: (a) Power breakdown by type; (b) leakage power breakdown by source.

Table 1.1: Performance comparisons.

	[15]	[4]	[14]	[8]	This work
No. of channels	32	128	64	16	1
Detection	Y	Y	Y	Y	N
Feature extraction	N	Y	Y	N/A	Y
Clustering	N	N	N	Y	Y
Data-rate reduction	12.5x	80x	11x	240x	85x
Clock frequency (MHz)	–	–	0.4	0.48	0.14
Throughput (sorting/ch/s)	N/A	N/A	N/A	420	420
Power (μW/channel)	75	100	2.03	4.68	2.17
Area (mm^2/channel)	0.11	1.58	0.06	0.07	0.05
Power density (μW/mm^2)	682	63.3	33.8	66.8	43.4
Process(nm)	500	350	90	65	65
Core voltage (V)	3	3.3	0.55	0.27	0.3

independent. The proposed design achieves considerably lower power consumption and silicon footprint.

1.5 Summary

In this chapter, we propose a hardware spike sorting accelerator that provides online, unsupervised clustering with initialization and order-of-arrival insensitivity. By screening data using proximity to centroid in feature space, we can improve the performance of the clustering algorithm as well as reduce hardware complexity. Efficient hardware architecture and circuit techniques are devised in order to reduce power consumption and silicon footprint. For the dataset instrumented from rats, the designed accelerator achieves 98% accuracy in spike sorting. The converged centroid error is also improved by 16dB. Implemented in a 65nm CMOS, the accelerator exhibits the footprint of 0.05μm^2 and the power consumption of 2.17μW, which are 25% and 53% better than the previous state-of-the-arts.

Chapter 2: 1.74-µW/ch, 95.3%-Accurate Spike-Sorting Hardware based on Bayesian Decision

This chapter presents algorithm/hardware co-design for real-time unsupervised spike sorting hardware for reducing power and improving sorting accuracy. We devise an algorithm based on Bayesian decision, which enables high accuracy while using noisy and simple time-domain features. Those simple features significantly reduce computation complexity, memory requirement, and thus the required number of cycles per sorting. The latter, coupled with the sparsity of spikes in time, makes the hardware idle for most of time, and thus we employ aggressive power gating and balloon latches to sleep most of the circuits and wake them up only when a spike is detected for maximal power savings.

2.1 Motivation

Online neural spike detection and sorting (clustering spikes by waveform features) is an important step between neural signal sensing and motor-intention decoding for closed-loop neural prosthetic systems. Implanted electrodes sense activities of multiple neurons while decoding often needs single-unit spiking rates. Therefore, implantable hardware for real-time spike detection and sorting [8, 16] is critical to close a loop and to reduce communication cost between an implant and an external receiver [4, 17].

One of the state-of-the-art hardware for spike sorting implements the sequential leader algorithm which computes sums of differences between samples of incoming and centroid waveforms [8]. While the simple time-domain feature allows it consume low power (4.68µW/ch) and compact area (0.07mm^2/ch), the hardware suffers from low clustering accuracy (78.2% while sorting 96k spikes) mainly due to the use of less robust time-domain features.

In this chapter, we pursue algorithm/hardware co-design to increase clustering accuracy and minimize power dissipation. We propose a novel sorting algorithm that can self-learn a decision metric based on Bayesian boundary [18] from incoming spike waveforms. While the algorithm also uses very simple time-domain feature (i.e., min and max values of a spike), the self-learning capability enables the algorithm to robustly sort spikes. When mapped onto hardware architecture, indeed, the simple time-domain features can largely reduce computation and memory requirement, allowing the hardware to finish sorting a spike in 2 clock cycles, while the conventional sequential leader algorithm can take hundreds of cycles. The reduced computing complexity coupled with spike sparsity in time allows us to have the hardware in a sleep mode for most of the time via power gating switches and balloon latches (BL).

We prototype test chips for the proposed sorter having one channel, which can sort 420 spikes/s at the sorting accuracy of 95.3% while consuming 1.74μW at VDD of 0.6V, both significantly improved from the prior arts [8].

2.2 Algorithm and Implementation

The sorter performs spike sorting in a per-channel basis. As shown in Figure 2.1, the hardware starts with a threshold based spike detector at the input. The detector is preconfigured with a threshold derived from channel noise, and operates on the continuous sample stream from the ADC. The features used in the spike-sorting task are the maximum (peak) and minimum (trough) value of each spike waveform. A feature extractor extracts the peak and trough of action potentials of each spike waveform for the following training or sorting process.

The theoretically optimal sorting solution in the 2-dimensional feature space is a set of free-form Bayesian boundaries from clusters' distribution intersects. This approach is, however, far too costly to on-implant processing as it requires large memory and complex computations. Instead, we project two of single-feature Bayesian distribution boundaries to the 2-feature space, forming a set of grid-constrained boundaries, effectively a 1.5-dimension feature space for a relaxed hardware requirement.

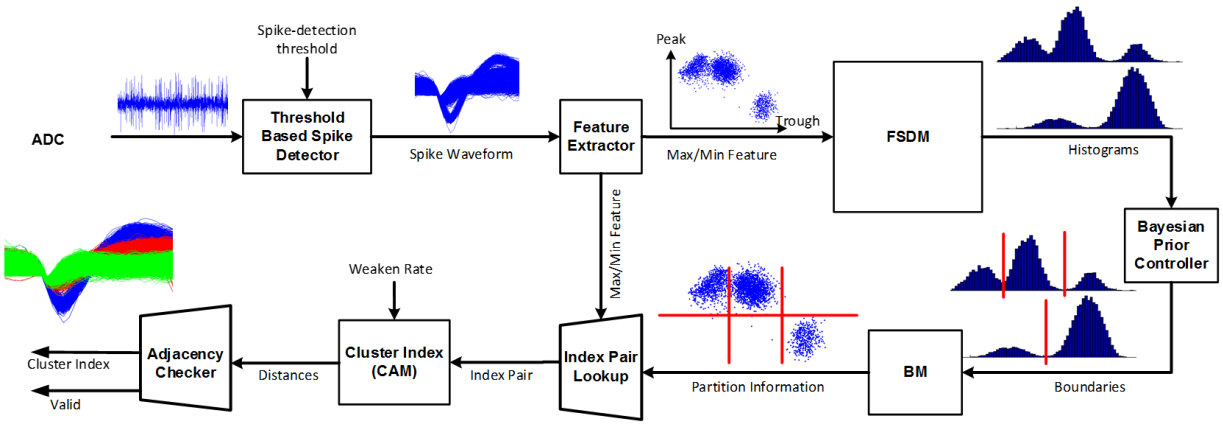


Figure 2.1: Spike sorting accelerator block diagram.

2.2.1 Feature Space Distribution

The first step of the training process is to identify Bayesian decision boundary. For each detected spike, the hardware updates the two histograms of the peak and trough values (Figure 2.1) in the Feature Space Distribution Memory (FSDM). After the specified number of spikes are used for updating histograms, the Bayesian Prior Controller traverses the FSDM for finding the local minima of each feature distribution and store them as Bayesian decision boundaries in the Boundary Memory (BM). The boundaries are used to orthogonally partition the feature space, with each partition identifiable by a pair of indexes.

2.2.2 Cluster Validness Check

After the boundaries are found, each feature space partition is to be assigned a status as either a cluster or unnecessary segmentation. This is done by updating the confidence level of the cluster status of each partition in the feature space. The specific steps are as follow. First, the Index Pair Lookup (Figure 2.2) compares the features of an incoming spike with the Bayesian boundaries in the BM. This locates the specific partition that the spike belongs in. The pair of indexes is then fed to a CAM (Figure 2.3). If the CAM finds the pair of indexes in it, it increases the confidence level of the entry by setting the associated 2b indicator (00-vacant, 01-outlier, 10-weak cluster, 11-strong cluster). If no match, it places the index pair in the first vacant entry, with indicator set

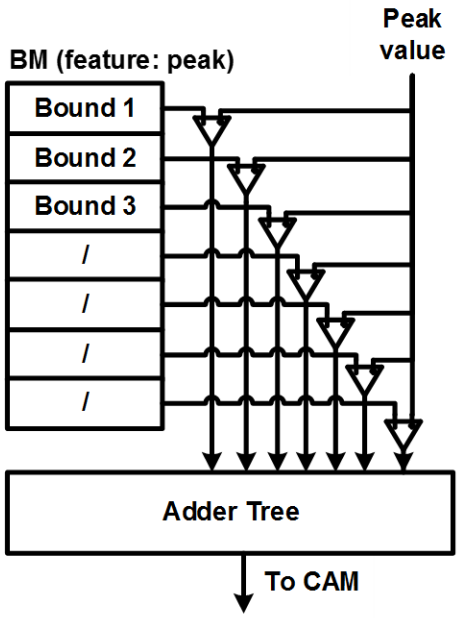


Figure 2.2: Index Pair lookup for peak feature.

to 01. The controller periodically decreases all entries' indicators once per N spikes to remove outliers from strongly-recognized clusters (Figure 2.3).

2.2.3 Decision-Tree-like Sorting

The spike sorting process can start after a specified amount of training. It performs the same computation for finding an index pair, but the hardware no longer updates the CAM indicators. The result of the CAM which represents distances to partitions enters the Adjacency Checker, which then finds the closest partition that is a valid cluster. The checker performs min function with a vector mask that selects clusters only with 10 and 11 indicators. The partition index is found via the comparison paths (Figure 2.4).

The algorithm is similar to a decision tree. The branching conditions are Bayesian boundaries found during training. However, the cluster status update serves as a pruning process on the decision tree in a manner that leaves it in a partial coverage of the feature space. The pruned branches are recovered in the deployment stage via a least distance comparison.

The proposed architecture can greatly reduce memory requirement and computational com-

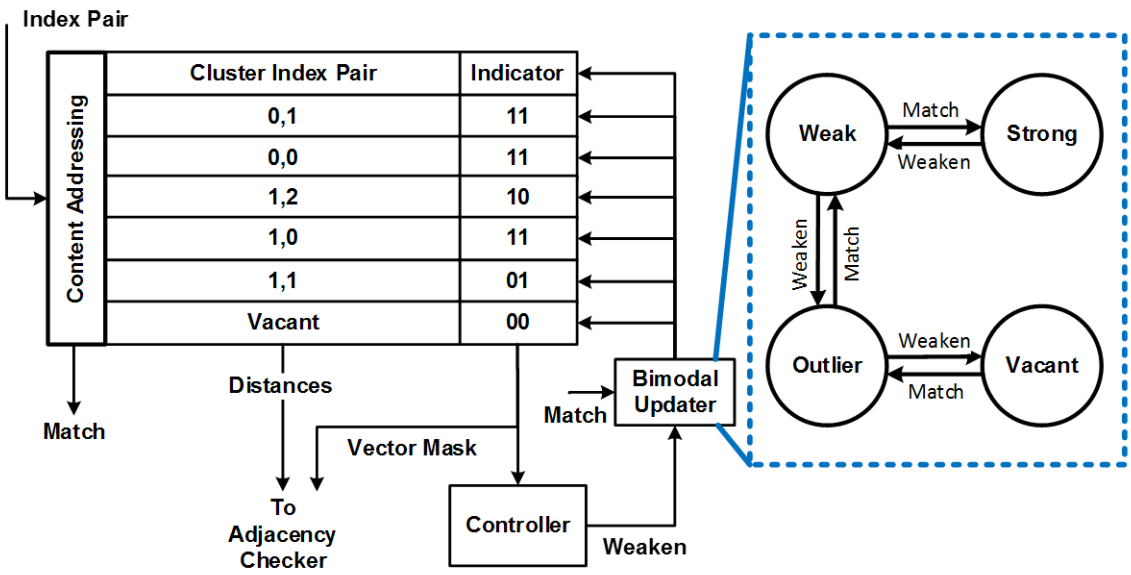


Figure 2.3: CAM architecture.

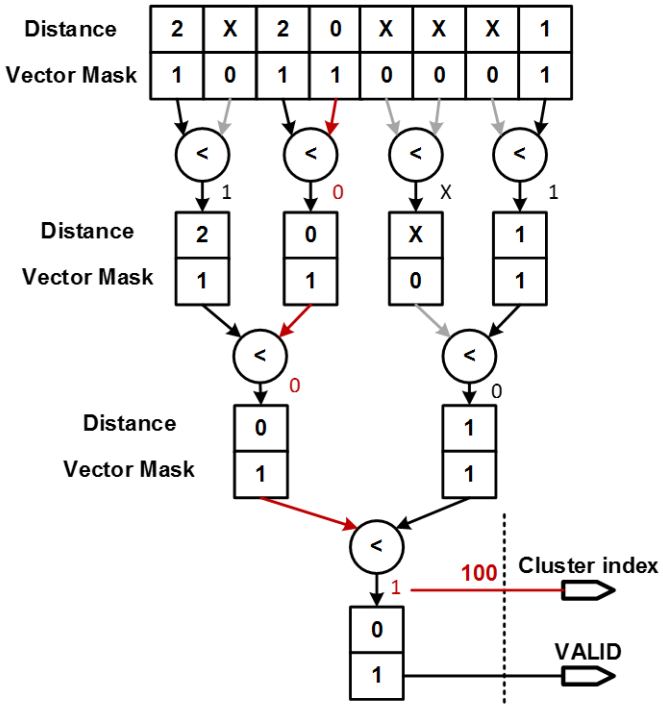


Figure 2.4: Adjacency Checker.

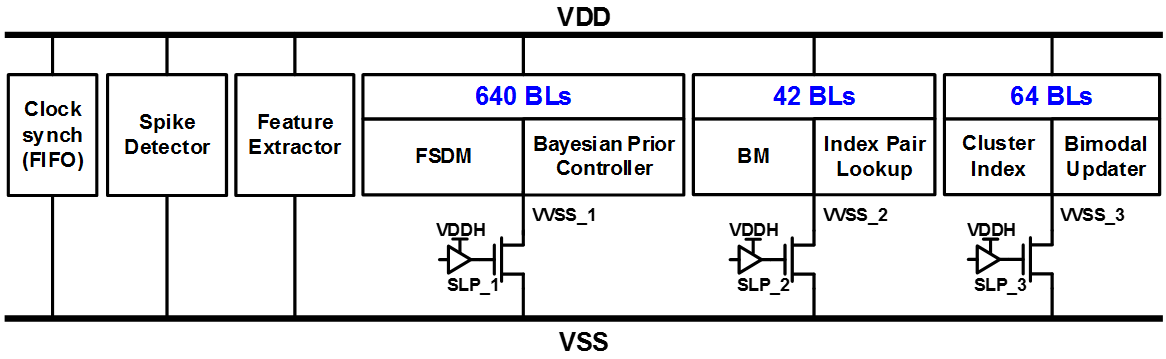


Figure 2.5: Module power gating grouping.

plexity. While the conventional sequential leader algorithm needs to store a full waveform per each cluster centroid, the proposed architecture stores only a pair of indexes and a few boundaries per each cluster. Assuming 48 samples/spike, 8b ADC resolution, 4 clusters/ch, and 2b/index, and 6b/boundary, the former requires 1,536 (= 48 * 8 * 4) bits while the latter can require only 36 (= 6 * 2 + 4 * 6) bits. Similarly, the proposed architecture requires eight 6b comparisons and single 6b equality operation per sorting while the sequential leader algorithm needs 48 8b additions and 45 14b additions per sorting.

Thanks to the low computational complexity, the proposed architecture takes only 2 clock cycles for sorting a spike. As incoming spikes are sparse in time, the hardware can be idle for most of the time. In order to minimize the power dissipation during idle time, as shown in Figure 2.5, the hardware aggressively employ power gating switches (PGS) and BLs. Therefore it can have most of the modules in a sleep mode and wake them up only when detecting a new spike (Figure 2.6). We design a BL in thick oxide devices for leakage savings (Figure 2.7). The control signals for PGS are bootstrapped to 1V via level converters for reducing wakeup time (T_{2WKU}). Figure 2.8 shows the operating waveforms of a BL.

2.3 Measurements and Comparisons

We prototype test chips for the proposed sorter in a 65nm high-V_t(Figure 2.9). The macro is of a single channel as online training has high area cost. In order to integrate the online training

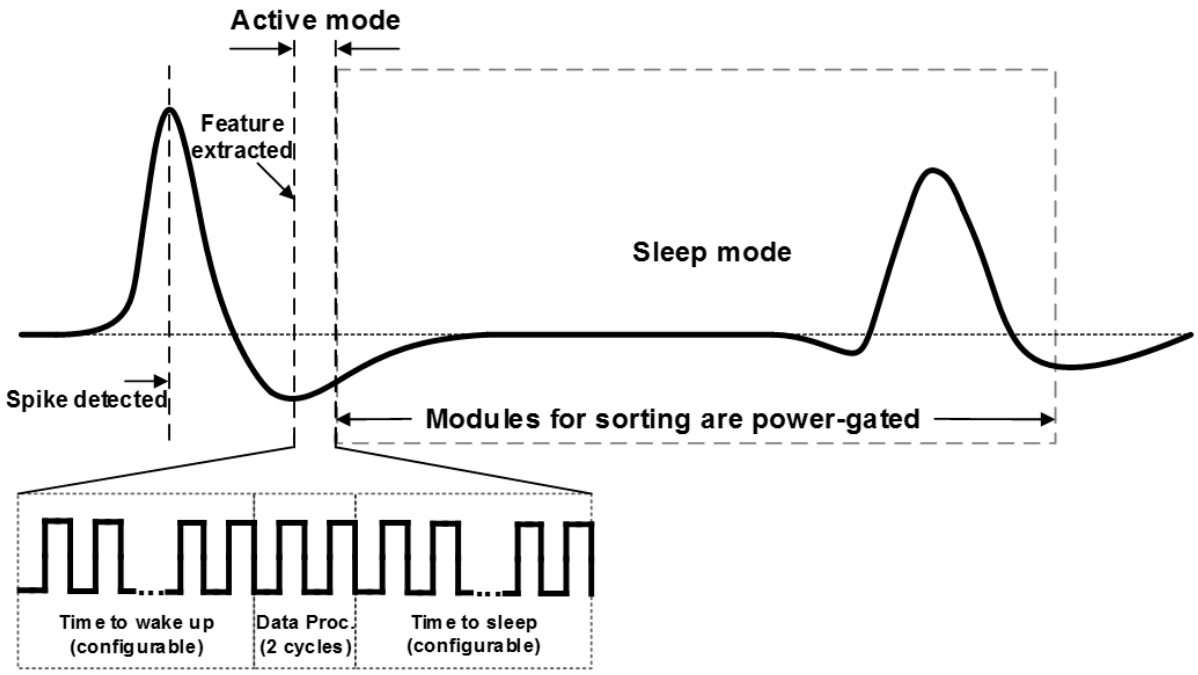


Figure 2.6: Power gating scheme.

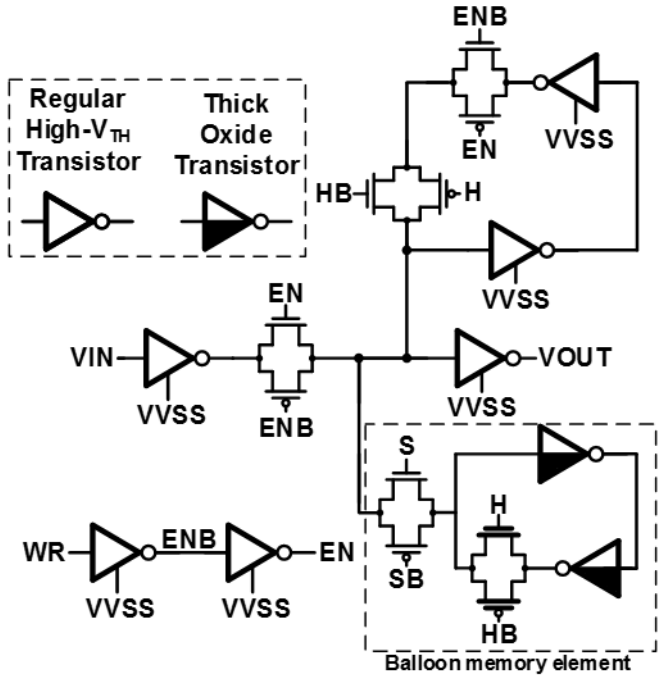


Figure 2.7: Balloon latch schematics.

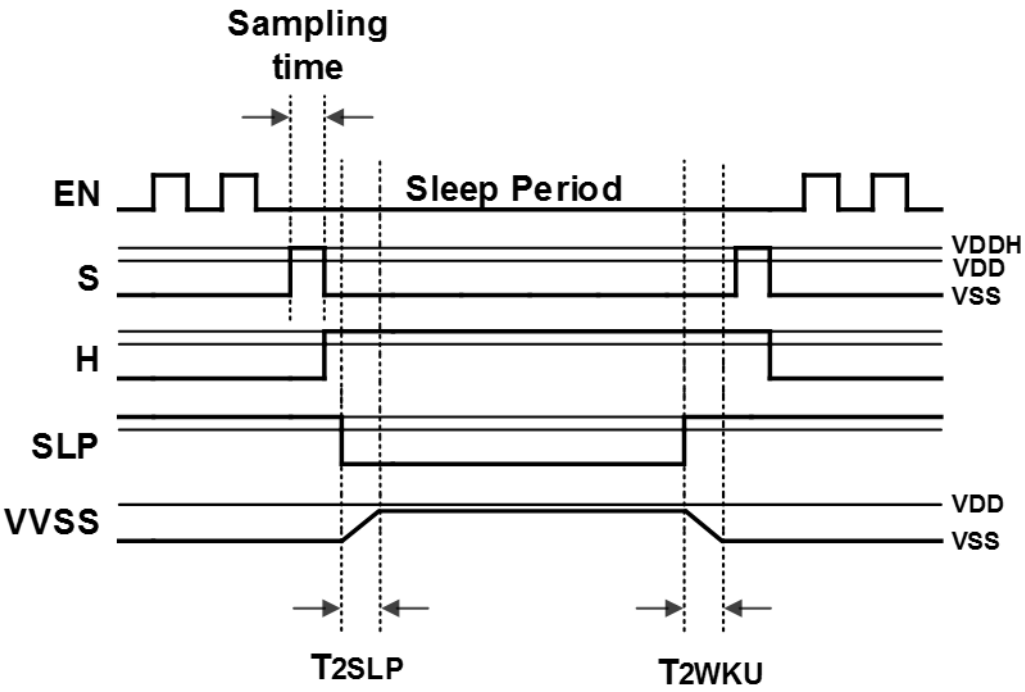


Figure 2.8: Balloon latch operation flow.

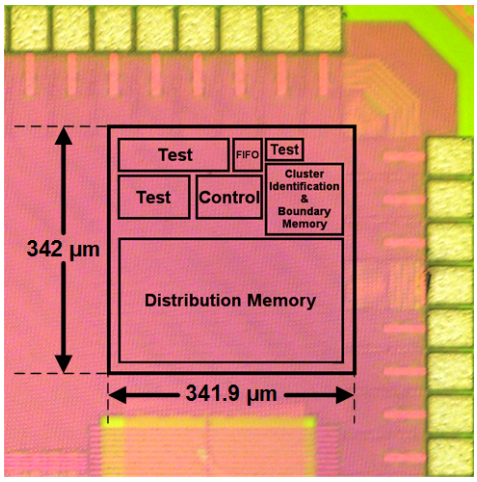


Figure 2.9: Die photo.

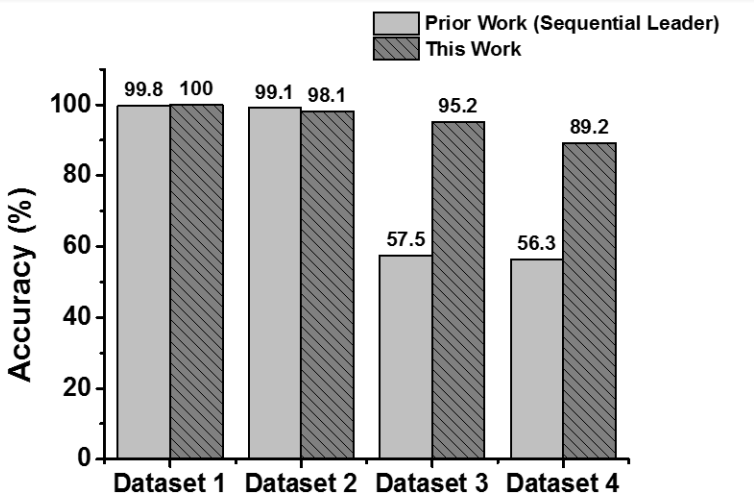


Figure 2.10: Sorting accuracy comparison.

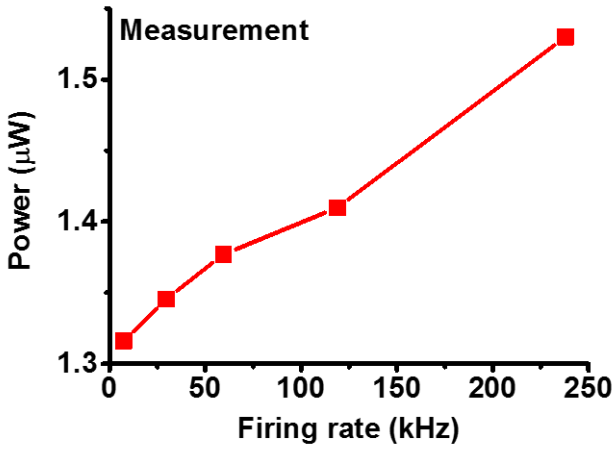


Figure 2.11: Power vs. spiking rate.

hardware in an implant, we designed the system to perform the training process for each channel in a time-multiplexed manner. The proposed architecture achieves the average accuracy of 95.3% in sorting 96k spikes in four datasets. For the same waveforms, the sequential leader algorithm can achieve the accuracy of only 78.2% (Figure 2.10). At VDD of 0.6V and the throughput of 420 spikes/s/ch, the proposed hardware consumes 1.74 μW, 2.7X smaller than the prior art's power dissipation of 4.68 μW/ch at the same throughput. Note that the power consumption scales with throughput (Figure 2.11). The proposed architecture takes 0.116mm^2 .

Chapter 3: A sub-microwatt 96-Channel Neural Spike Processor for a Movement-Intention-Decoding Brain-Computer-Interface Implant

This chapter presents sub-microwatt end-to-end neural signal processing hardware for deployment-stage real-time upper-limb movement intent decoding. This module feature intercellular spike detection, sorting, and decoding operations for a 96-channel prosthetic implant. We design the algorithms for those operations to achieve minimal computation complexity while matching or advancing the accuracy of state-of-art BCI sorting and movement decoding. Based on those algorithms, we devise the architect of the neural signal processing hardware with the focus on hardware reuse and event-driven operation. The design achieves among the highest level of integration, reducing wireless data rate by more than four orders of magnitude. The chip prototype in a 180-nm high-V_t, achieving the lowest power dissipation of 0.78μW for 96 channels, 21× lower than the prior art at a comparable/better accuracy even with the partial decoding function integration.

3.1 Motivation

Advances in brain-computer-interface (BCI) research is aiding the development of prosthesis for patients with limited mobility. Prosthesis can be categorized as passive or active. While passive prosthesis only provides structural support for patients, active prosthesis can perform the patients' intended motor function, autonomously or controlled. BCI can aid active prosthesis by mapping their neural activities to the intended movements and actuating them. Hence, BCI systems are invaluable in limited mobility rehabilitative services [19, 20, 21, 22, 23].

A prosthetic BCI operates by measuring neural activity and inferring the intended movement based on a learned model (cortical map) that relates the neural behavior to the movement intention. Neural activities useful for motor intention decoding can be sampled directly from residual muscle

activation near the prosthesis site or deep within the motor cortex in the brain. Any neural signals encoding motor intention can support BCI prosthesis if the encoding scheme can be reliably modeled. For locked-in patients without residual muscle activation, only central nervous signals can aid the prosthesis. Extracellular spiking activity from pre-motor or motor cortex is currently the state of the arts for upper limb movement decoding [21], outperforming non-invasive systems based on signals such as EEG [22] or MEG [23]. Unlike EEG or MEG, the detection of spiking neural signal requires surgical procedure to implant probes and peripheral support devices. The invasiveness places additional physical design constraint, e.g. power consumption, to the already complex algorithmic challenges.

In this chapter, we present a full stack design and the prototype of a prosthetic Neural Spike Processor (NSP). The NSP decodes neural spike information into direction and velocity of intended muscle movement, enabling rehabilitative services for patients. We modify and improve existing neural decoding algorithms to implement our design.

The remainder of the chapter is organized as follows. In Section 3.2, we first briefly present the scope and tasks involved in the BCI system. In Section 3.3, we detail our spike sorting algorithm. In Section 3.4, we present our neural decoding approach and evaluate the accuracy and cost of our hardware. In Section 3.5, we present the architecture of the NSP chip. In Section 3.6, we present the prototype and measurements. Finally, Section 3.7 summarizes the chapter.

3.2 System Overview

Figure 3.1 shows the processing stages of the targeted extracellular spike BCI system. The first stage is an implanted electrode array. The electrode array senses extracellular potentials, which originate from surrounding neurons. The signals are then filtered with band-pass or low-pass filters. Following the filtering stage, analog-to-digital converters (ADC) digitize the extracellular voltage signals and produce multi-channel digital data streams for later stages of the BCI system. The NSP initiates after the ADC stage of the BCI and terminates before the transmission of partially decoded neural data off chip.

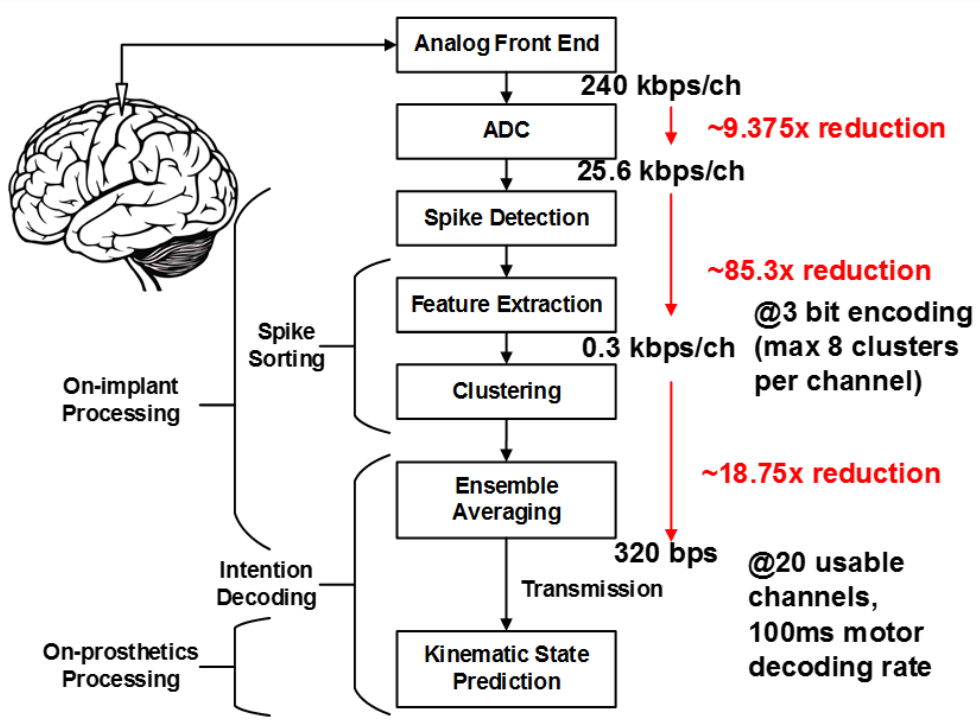


Figure 3.1: Proposed prosthetic BCI task flow.

The motor intention-decoding task is delegated to both on-chip hardware processing at the implant site (NSP) and off-chip software processing at the prosthesis site. The on-implant computation includes spike detection, spike sorting (which includes feature extraction and clustering), and partial computation of intention decoding which estimate the movement state via ensemble-regressed spiking events. The near-prosthesis computation concludes the rest of intention decoding, notably the Kalman filter (KF) operation, which finalize the kinematic state prediction.

The main physical constraints of an implantable BCI device is power efficiency. The temperature sensitivity of the implant site tissues can render the targeted neurons useless when exposed to high power implant. Among the tasks performed by the implant, data transmission from implant to prosthesis is the dominate power consumer for existing BCI implants [8]. Hence, our NSP design is driven by the objective of achieving the highest data rate reduction while performing the minimum amount of computation on implant.

Based on the prior works [19, 20, 24], we optimize and improve the algorithms for spike sorting and intention decoding to reduce the on-chip computational complexity while improving

the decoding accuracy as compared to the unmodified algorithms. Consider a typical 96-electrode array sensing at 8-bit resolution at 30 kHz for prosthetic BCI, the data rate is nearly 3MB/s without any on-implant processing (Figure 3.1). If the full data streams are transmitted off-chip entirely, the required power would make the system unsuitable for long term deployment. Therefore, we implement spike detection, feature extraction, sorting, and the first part of decoding on the implant device. With the on-implant processing capability, we can reduce the wireless data rate by more than four orders of magnitude.

The functions of the NSP is as follows. The processor receives the 96 channels of streams and detects neural activations (spikes) by thresholding the action potential. The NSP then performs spike sorting by their waveform shape. The sorted neurons are then analyzed for their spiking behavior. The neurons' efficacy in kinematic information encoding determines their usage in the following stages. The motor intention decoding uses a cortical model to map instantaneous spiking rates from the selected neuron population to kinematics. Instantaneous spiking rates are in practice spike counts in time bins, which are typically around 100 ms. The cortical map is attained by regressing training data, typically at the same step when we identify the neurons that encode significant kinematics information. Finally, we use filtering techniques such as the Kalman filter (KF) to refine the kinematic estimates.

This calibration procedure requires in-patient experiments, and is typically very hard to fully automate in the deployment system. The typical calibration process is as following. A lock-in patient with a BCI implant is asked to imagine a preprogramed movement while the BCI system asserts a small amount of the control over the visual feedback of said movement (screen or prosthesis). Gradually, the BCI's decoded results are asserted more and more influence over the preprograming of prosthesis until the movement is entirely driven by the BCI system. The cortical map developed by outpatient analysis is used to make the initial motor prediction. The rigorous training of the BCI decoding features relies heavily on the neural plasticity of the patient's neurons. The calibration step is much more of a rehabilitative reconfiguration process on the motor cortex neurons than a setup step on an out-of-the-box working machine. While the decoding suc-

cess is dependent on the patient's neural plastic health, the initial decoding input greatly affects the viability of the BCI system.

3.3 Spike Sorting

Each extracellular electrode measures the activity of neurons in its proximity; hence, a sorting process is required to differentiate spikes by their originating neurons. The basis for sorting is the spike waveform's shape, under the assumption that multiple neurons are at various distances to the electrode and they have unique internal ionic gating states. The varying distances through the brain tissues to the electrode have varying filtering effect; the ionic gating states affect the spike shape such as relaxation and pre-spiking depreciation. The spike sorting process uses selected waveform markers to identify the individual neurons. Spike sorting provides substantial savings in power. Assuming the 10-bit identifier for 96 electrode channels (3-bit identifier for each channel), 4 clusters per channel, and an average spiking rate of 50 Hz, spike sorting can achieve 100X reduction in data transmission (Figure 3.1). Our design goal concerning spike sorting is to implement algorithm and hardware that consumes minimal energy and area at high accuracy.

3.3.1 Supervised Training of the Constrained Bayesian Boundary Sorting

For the NSP development, we designed the similar spike sorter based on the constrained Bayesian boundary model in Chapter 2 yet uses the offline-training model for improving accuracy. The accuracy of the online-trained model is weaker for several reasons. One factor is the suboptimal feature selection. The independent nature of the single-feature Bayesian boundaries over-segments the feature space. Another is that the pruning process leaving only partial coverage of the feature space. Some of this problem is recovered by the adjacency checking of the feature to the known branching nodes. Still, it is not fully reliable since not all nodes are Bayesian optimal for that particular modality.

Our supervised-trained sorter uses two voltage samples directly selected from the waveform, but not necessarily the peak and trough values as used in the unsupervised training version. This

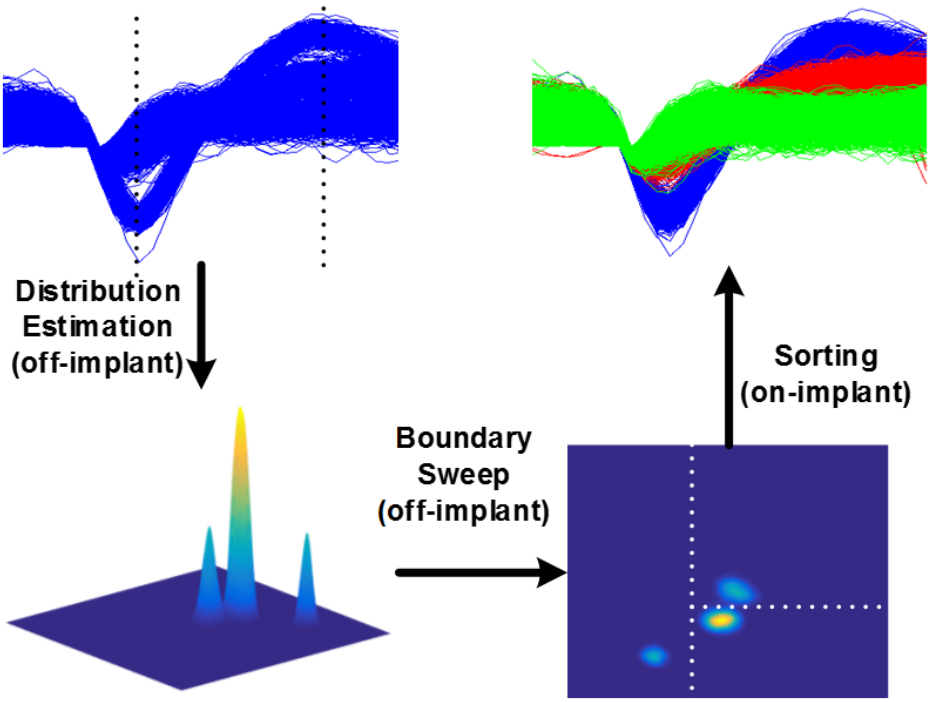


Figure 3.2: Visualization of spike sorting via Bayesian approximate decision tree.

change is motivated by the followings. The use of raw waveform samples performs near optimal in our simulation but it exhibits large dimensionality, and thereby increasing power dissipation [8]. To improve accuracy while keeping the dimensionality low, we determine which two sample indexes used for the feature during the offline training process. We perform a parameter sweep to find the best performing pair of indexes, across all the spike waveforms used in the training process, as the decision tree's feature space.

We construct the decision tree in offline training (Figure 3.2). First, we estimate the distribution of data points with Gaussian kernels. We then sweep for boundaries under the orthogonality constraint. The boundary information (direction, order, and values) is stored in implants. We limit the maximum number of neurons per channel to four, which is anyway rarely exceeded. In the case of more than four clusters present in a channel, the decoding performance is not affected with a heuristic that no more than three neurons from one channel are selected for intention decoding.

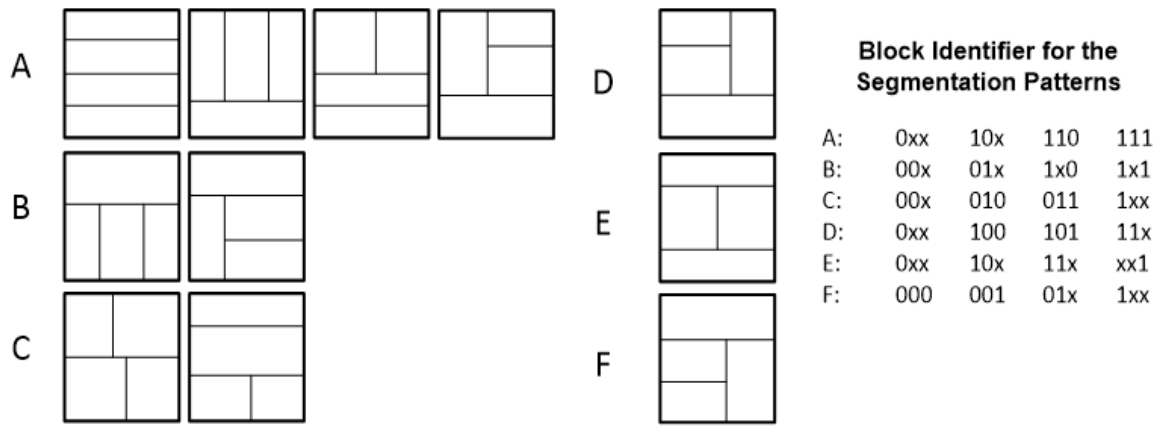


Figure 3.3: Feature space segmented block identification coding.

Under the constraint limiting to four clusters per channel, eleven unique segmentation patterns exist in the feature space (Figure 3.3). Here, the diagonally symmetric patterns are considered a single class. Each spike, as a point in the space, can be located by simple comparison with the segmentation boundaries (these are the decision tree branching conditions). Its cluster is identified by the 3-bit comparison results. The comparisons against each of the boundary, combing with the feature organization, produce a unique marker that identifies it as a particular cluster within a channel.

The 11 segmentation patterns are organized into six groups. For all patterns in the same group, the four blocks share the same boundary comparison identifier. This also means the order of the features being used for of decision tree branching condition are in the same order. This allows an encoding scheme that reduces memory needed to characterize the channel as compared to the online-trained implementation in [24]. Online training require allocation of memory for the full 4x4 grid since the optimal segmentation pattern is unknown. This requires three boundaries along each dimension, and a large hash table to store 16 blocks' information (cluster validness, its associated weight for intention decoding). The proposed offline variant thus has more than 50% reduction in memory requirement.

Table 3.1: Spike sorting accuracy.

	Datasets (Number of neurons)			
Decision metric	D1(2)	D2(3)	D3(4)	D4(4)
L1 Norm	99.97%	99.25%	91.39%	89.00%
Const. Boundary	99.99%	99.19%	91.61%	89.49%

3.3.2 Accuracy and Cost Evaluations

We use the data measured from mice [9] to evaluate the sorting accuracy of our proposed algorithm. The dataset (D1, D2, D3, and D4) contains channels having signals from two to four neurons. The sampling rate is 40 kHz, band-pass filtered between 300 Hz and 5 kHz.

We categorize prior works on on-chip spike sorting by their decision metric, namely distance to template [8, 25, 26] or boundary [24]. Due to the high area and power cost of a multiplier, all of the distance-based sorters use L1 distance metric (the sum of absolute difference of two vectors) on time-domain features. In 3.1, we compare our constrained Bayesian boundary sorting to the L1 norm and find that ours achieves a comparable sorting accuracy with the L1 norm, both using two features.

The decision boundary based on L1 norm is not constrained to orthogonal grids in the feature space, which can theoretically outperform the proposed constrained boundary model. However, as the features are voltage samples taken directly from the spike waveform, its non-idealities (noise) account for its sorting performance. Since spike's peak, trough, and relaxation slope are primarily driven by different ion pumps, different parts of a spike waveform have different variances. L1 metric does not consider this phenomenon; thus, its sorting accuracy is negatively affected.

The advantage of the decision tree is that it requires much less on-chip memory and computation than the distance based technique. The memory required in the presented sorter per channel is 3 bytes for storing boundary information, 4 bits for segmentation pattern (total 28 bits) for n-cluster per channel ($n < 5$). The total is 336 bytes for the 96-channel NSP. By comparison, a 2-feature

L1 metric requires 2n bytes (64 bits when n=4), marking 768 bytes for the 96-channel system. In term of the computation complexity, the 2-feature L1 requires 3n 8b additions/subtractions and n-1 8b comparisons, while our proposed model only needs three 8b comparisons and one 3bit 4-entry CAM read.

3.4 Intention Decoding

3.4.1 Ensemble Observation Kalman Filter

In this subsection, we present Ensemble Observation Kalman filter (EOKF). This filter not only improve decoding accuracy, but also reduce on-chip computation workload, and minimize the data rate in transmission to off implant.

The standard KF for motor intention decoding has the following form:

$$x_{k+1} = Ax_k + w_k \tag{3.1}$$

$$z_k = Hx_k + q_k \tag{3.2}$$

The term x is the state, i.e. position, velocity, etc. A is the state transition matrix; w is the state noise (typically zero-mean Gaussian variable); z is observation, i.e., binned spiking rates; H is the observation matrix, i.e. cortical map; q is observation noise (zero mean Gaussian variable); subscript k is the time step.

Eq. 3.1 describes the state transition in a Markov chain. It simply provides a movement constraint from one moment to another. The constraint acts as a dampener to avoid overly aggressive prosthetic movement. Eq. 3.2 describes the more interesting behavior as it formulates the cortical mapping between kinematic state and neural activity. Given an estimated state, Eq. 3.2 reconstructs the expected spiking rates from the selected population.

The standard KF updates its state and parameters iteratively, beginning with an a priori estimate

of next kinematic state via the Markov process. A priori estimate is given as:

$$x_k^- = Ax_{k-1}^- \tag{3.3}$$

The standard KF's posterior estimate combines the passive estimate from state transition and the weighted error between expected observation and actual observation in the following equation:

$$x_k = x_k^- + K_k(z_k - Hx_k^-) \tag{3.4}$$

The error weights are known as Kalman gain K, computed as:

$$K_k = P_k^- H^T (HP_k^- H^T + Q)^{-1} \tag{3.5}$$

The Kalman gain is updated with the error covariance matrix P and the measurement error matrix Q. Eq. 3.5 represents a substantial computation load, since the number of neurons selected for motor intention decoding is relatively large, typically between 20 to 50 [21, 27], making H P-k HT at least a 20×20 matrix. In particular, there exist no closed-form solution to inverse such large matrix. Employing a numerical method for this inversion problem inevitably increases power and area overhead for an implant.

To reduce this computational complexity, we propose an inverse form of observation-to-state transition as a committee machine, essentially changing eq. 3.2 to:

$$x_k = Ez_k + q_k \tag{3.6}$$

in which the expected state is constructed with ensemble spiking rates, weighted by E. Figure 3.4 illustrates this. After this operation, the cortical map H now becomes trivial (identity matrix) in EOKF, therefore reducing the computational workload henceforth. The proposed filter has the same a priori estimate as the standard KF, as shown in 3.4.

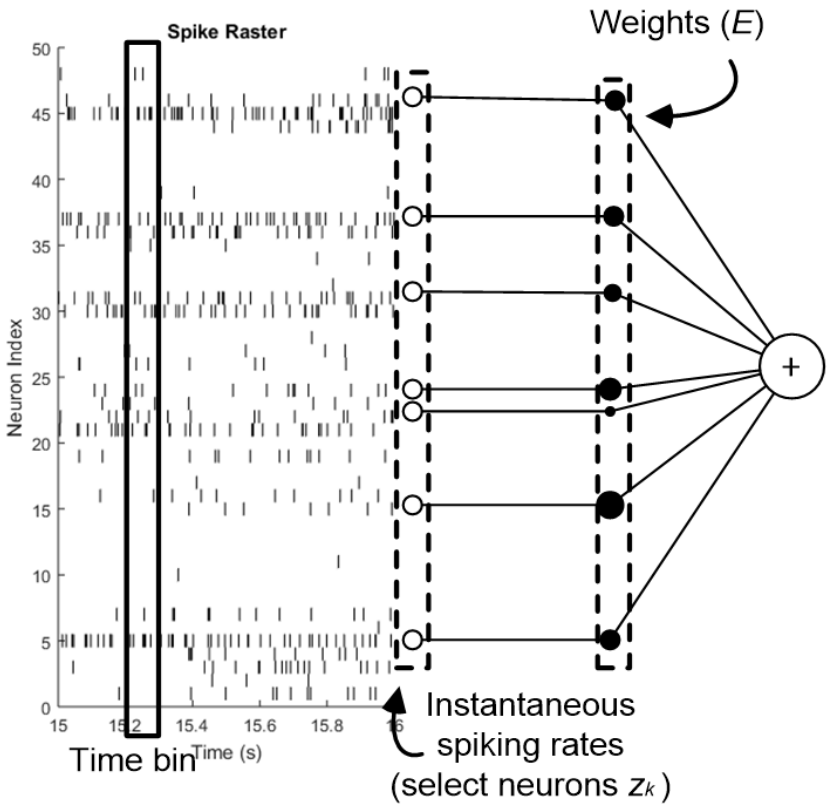


Figure 3.4: Instantaneous spiking rates weighted in ensemble average for the observation based estimate of a kinematic state.

In the EOKF, eq. 3.5, 3.6 are reduced to:

$$x_k = x_k^- + K_k(Ez_k - x_k^-) \tag{3.7}$$

$$K_k = P_k^-(P_k^- + Q)^{-1} \tag{3.8}$$

With the new smaller filter (i.e., eqs. 3.4, 3.7, 3.8), the lowest data bandwidth of the entire BCI system is located at computation of Ez_k. This term has the equivalent data rate as the final decoder output (x_k). Therefore, we perform only the computation of Ez_k and all the other computations are offloaded to prosthetics sites where power and area budget is much greater. This partition also includes the computation of eq. 3.4 in the prosthetics site since there is no data dependency.

The posterior error covariance matrix updates in the standard KF in the following form.

$$P_k^- = AP_k^{-1}A^T + W \tag{3.9}$$

$$P_k = (I - K_kH)P_k^- \tag{3.10}$$

The error covariance estimate is first updated with state transition and state error variance W in eq. 3.9. The posterior error covariance incorporates the observation error in the form of Kalman gain in eq. 3.10. In the proposed filter, eq. 3.10 is reduced to eq. 3.11.

$$P_k = (I - K_k)P_k^- \tag{3.11}$$

As mentioned, the posterior error covariance matrix width is now the number of state elements, 3 (x/y directions, and velocity), instead of the number of neurons.

Finally, in 3.2, we compare the number of multiplications, additions, and divisions in the proposed EOKF and the standard KF. We assume to select 20 neurons for intention decoding. We can find 1-2 orders of magnitude reduction in the operations, even for the case of including the

Table 3.2: Comparison of number of calculations in EOKF and standard KF (20-neuron cortical map).

	Number of Calculations (Mult/Add/Div))	
Equations	Standard Kalman	EOKF
3.4	4/2/-	4/2/-
3.5/3.7	80/80/-	46/46/-
3.6/3.8	32180/32060/1180	10/9/4
3.9	8/8/-	8/8/-
3.10/3.11	88/84/-	8/8/-

computations both on implants and prosthetics site.

3.4.2 Theoretical Basis

The standard Kalman filter (hereafter referred to as Kalman filter) has been one of the state-of-art decoding methods in cortical spiking based motor intention BCI. Its observation (state) model is straightforward. The estimate computation is based on the preferred direction property of select neurons. The observation based state estimate, i.e. instantaneous velocity, is modeled as a weighted sum of target neurons' time-binned spiking rate. The weight is derived solely from error covariance.

The Kalman Filter's weighting basis does not regard the neuron's firing behavior across intended movement directions, using the same error variance in all cases. This assumes that a neuron's spiking rates maintains the same signal-to-noise ratio regardless if the intended movement is in its preferred direction. In actuality, the spiking rate variance during a state of excitation, inhibition, and idle are not constant, thus the linear regression of single neuron observation have uneven reliability depending on the a priori estimate. Furthermore, even if the spiking variance is always constant, high firing rate (preferred direction) is more reliable simply due to higher SNR.

The more biologically realistic model has been validated by some studies that have fitted the excitation response in cosine or wrap-around Gaussian functions [28]. This is the tuning curve of

the neuron. If the tuning curve model is used to improve the Kalman filter, it will take the form of a separate iterative process to modify the error covariance from the previous step to one that corresponds the current state transition prediction. This will require a large lookup table for the error covariance values from training experiments and would increase the cost of computation. Furthermore, neural models are non-stationary. Changes in neural behavior would require more drastic correction to the tuning curves if an adaptive model is needed [29].

The proposed EOKF adapts an entirely different approach to weighting neurons' predictions. By reducing the probability distributions of many individual observations to a single population ensemble observation, the EOKF uses a population vector model and determines the weights through multi-variate regression. Instead of the individual spiking rate modulated only by error covariance, it is further modulated by its own excitatory state, benefiting from the signal of a higher SNR. As shown in Figure 3.5, an example of 4 neurons firing when the movement is in direction of radian angle of -1. For neuron 1, the intended movement does not corresponds to excitation. However, the baseline firing is noisy and still has a high spiking rate. For neuron 2 to 4, the spiking rate is close to the tuning curve profile. The baseline Poisson noise dominated spiking from neuron 1 is thus compensated in an ensemble, since the weights acquired via multiple regression account for the behavior of excited neurons at the same time. Hence the weighted vector sum is a better observation than the single neurons'.

3.4.3 EOKF Evaluation

To evaluate the performance of our proposed EOKF, we use an upper-limb reaching data set [30] from the Database for Reaching Experiment and Models (DREAM) [31]. The task is the standard 2D equal-distance 8-target center-out-reach-and-return performed by a Rhesus monkey well trained in the experiment. Only the velocity vector of the hand movement in the x, y plane is used as kinematic state, same as the velocity-Kalman study [20]. The data set contains 194 trials of the 196-neuron spiking traces from the motor cortex.

In this study, the data is used for offline reconstruction of native movements. We train the filter

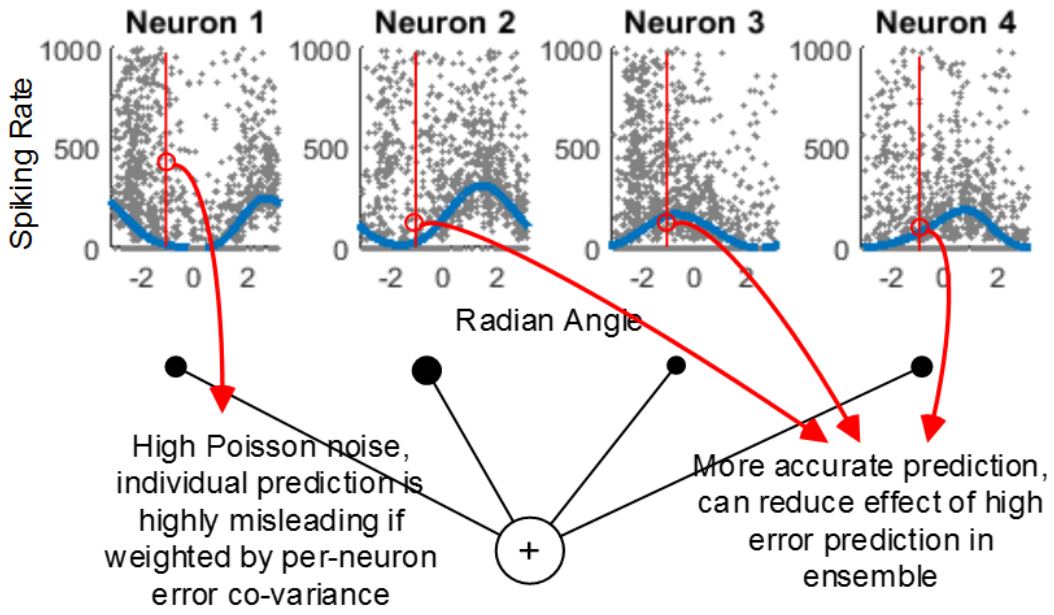


Figure 3.5: Multivariat regression of ensemble neurons reduce the effect of noise.

state parameters (transition matrices, error variances) with 80% of the data (randomly selected for each trial). Post-training decoding (reconstruction) is done on the remaining 20%.

The proposed EOKF outperforms the standard KF with lower trace reconstruction error, by 5% to 46% depending on trial selection. This outperformance comes from the advantage that the proposed EOKF always has better observation than the standard KF. In the proposed EOKF, the error variance of the observation-based estimate has an upper bound that is equal to the lowest error variance of its members, in which case, the committee is trivial having the single member determine the output.

The advantage of the committee machine also manifests in error variance consistency. Figure 3.6 demonstrates the regression residuals of an ensemble and a single neuron. Each dot represents a kinematic state instance. The residual is color-mapped to show that the accuracy of single neuron model (used in the standard KF) is less consistent across directions and movement speed than the ensemble model used in the proposed EOKF. The error variance matrices are assumed invariant in both the standard KF and EOKF; hence, the ensemble better fits the variance model due to its evenness.

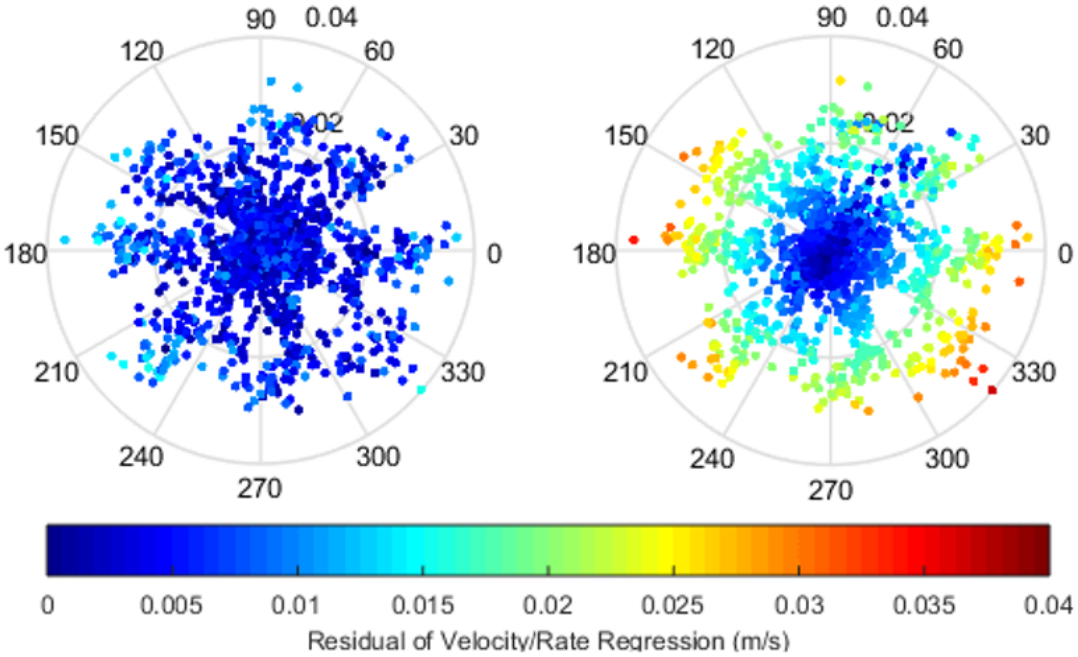


Figure 3.6: State prediction error of ensemble prediction (left) and single neuron prediction (right) across angles and speed.

3.5 NSP Processor Architecture

Based on the optimized sorting and decoding models, we prototyped the 96-channel Neural Spike Processor (NSP). The NSP architectural design focuses on resource sharing to minimize area and power. In addition, we exploit data sparsity inherent in neural spike activities to implement event-driven computation for lower resources. All stages after parallel spike detection can share hardware without a separate fast/slow clock domain, data stream multiplexers, or additional controller for time multiplexing.

Figure 3.7 shows the on-chip hardware architecture of the proposed NSP. It includes the modules for spike detection, spike sorting, and the ensemble regression part of the EOKF. The NSP starts with 96 non-overlapping spike detectors each of which integrates a simple feature extractor. These spike detectors deliver spike features directly to the sorter modules. We grouped 32 of the spike detectors to share a single set of sorter hardware. Each sorter hardware has the memory entries of boundaries for the 32 channels and those for neuron identifiers for the data transfer of

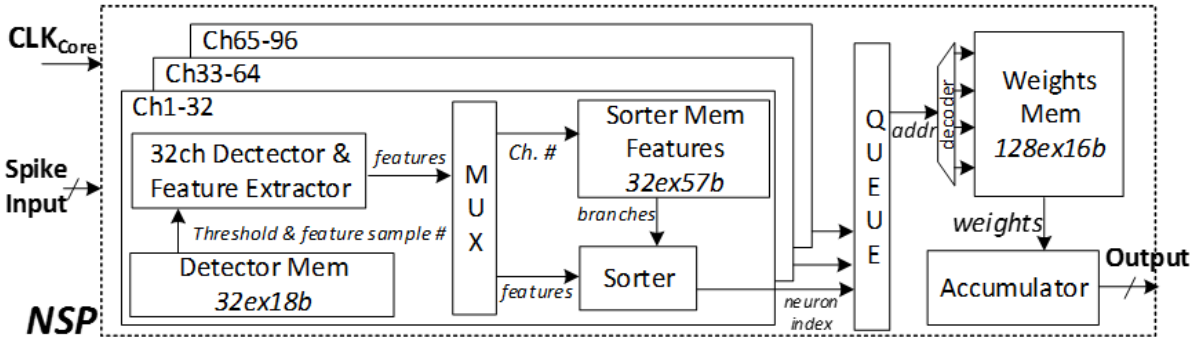


Figure 3.7: The architecture of the proposed NSP.

spike events to the following decoding module.

The spike waveform length mainly determines the group size, i.e., the number of spike detectors that share sorting hardware. In our design, each waveform has 32 samples. Therefore, as long as the sorter takes only 1 cycle to perform sorting (the case for the NSP) or fully pipelined and it employs the non-overlapping detection scheme, no channel can generate more than one spike event within 32 cycles and the sorter can sort every detected spike online without data buffering.

Specifically, at the event of a spike, the detector enters an event token (i.e., spike features) onto a conveyor style queue (Figure 3.8). The queue entry points from detectors have a simple stalling rule that gives priority to the token already on the conveyor, the stalled token then attempts to enter the queue in the subsequent clock cycles until a free spot is available. With the non-overlapping detection and the defined samples per spike, token stalling does not generate backpressure to earlier stages. The conveyor queue does not preserve the time order of spikes. However, the order is irrelevant downstream where only spiking rate is considered. Thus, this minimally affects the accuracy of the rate based decoding schemes that our EOKF belongs to. No spike is stalled beyond the time bin edges (nor would it be problematic as spike rate has relatively high Poisson noise) to affect the following rate computation.

The sorter modules, each responsible for 32 channels, consume event tokens in the order of the channel array if no collision is present in the data streams, but in cases of token stalling, channel order is compromised. As a result, a neuron index (i.e., address) must be included in the token

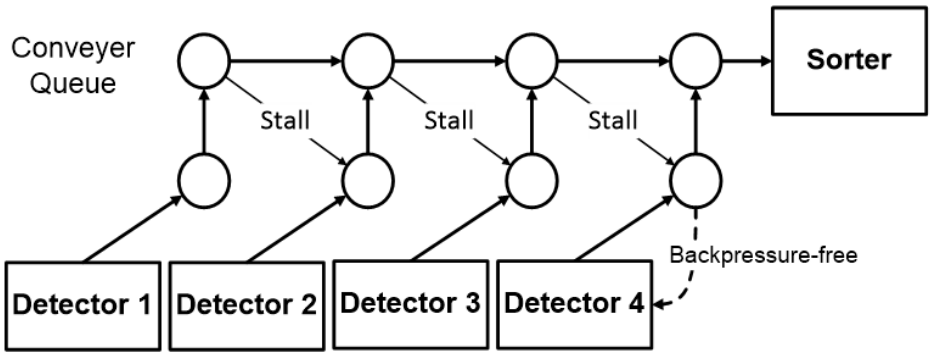


Figure 3.8: Conveyor style queue of a 4-detector example.

to retrieve correct boundary information used in sorting. The sorter checks the incoming features against boundaries stored in the Bayesian boundary memory. Note that we use the coding based on fixed segmentation patterns in the feature space (Section 3.3.1). Thus, the outputs of the sorter are the indexes (addresses) for the memory storing weights for decoding.

Finally, the architecture has a single ensemble-regression module (memory and accumulator). This decoding module does not require resource sharing since all data paths converge to a single register per state element in Ez_k. Instead of calculating spike rates and multiply them with weights (E), upon every spike event, we perform memory read for retrieving a coefficient and then accumulate it using a single adder. This is equivalent to multiply-accumulate since the instantaneous spiking rate is a count of spikes in a fixed-duration time bin. In place of the multiplier, each spike event immediately triggers an addition of its weight in the ensemble observation registers. This architecture can reduce silicon area and thus leakage power.

A typical challenge of event-driven implementation in place of a scheduled one is potential data collision hazards. In our architecture, we can have collisions among the three spike sorting modules when they try to access the ensemble-regression module at the same time. Unlike spike detection, the sorter has no hardware constraint for token generation. With a finite amount of buffer memory, therefore, token loss is possible. Practically, however, token loss is improbable since only a small subset of all sorted neuron channels (e.g., 20 50 in typical experiments) is selected for intention decoding. In our test, no token loss or even collision occurs even with all sorted neuron

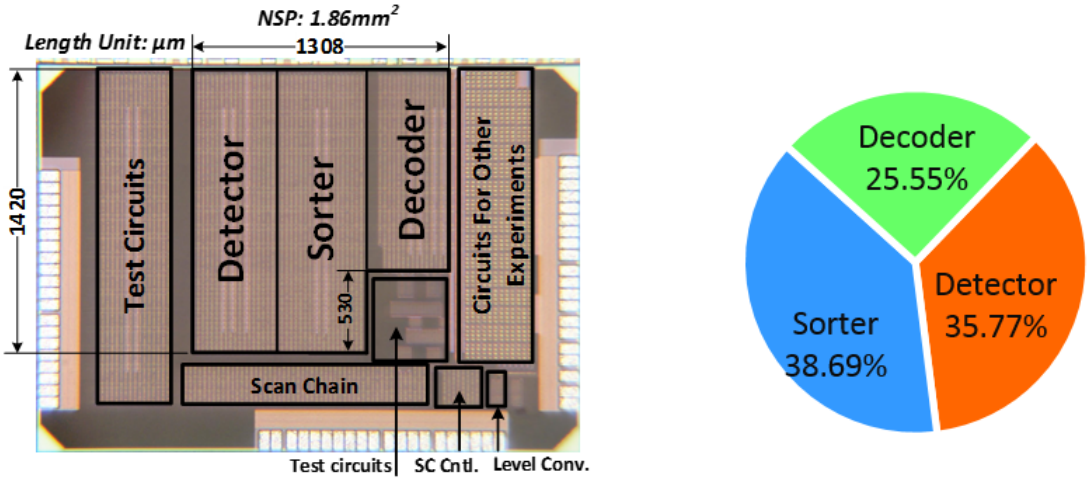


Figure 3.9: Die photo and area breakdown.

channels considered valid. In the unlikely event that a token is lost, its effect is minimal since the ensemble-regression module can easily tolerate small loss.

3.6 Prototype Measurement

We prototyped the 96-channel NSP in a 0.18 μm CMOS technology. The technology is chosen since leakage power is the major energy efficiency bottleneck in the NSP [32]. Figure 3.9 shows the chip die photo. The total area is only 1.86 μm^2. The 96 detectors, three sorters, and one EOKF decoder take the similar silicon footprints.

The power consumption of the NSP is data rate dependent thanks to its event-driven operation (Figure 3.10). Operations at each step may be terminated as soon as information of each spiking event's efficacy at changing the decoding output becomes available. Spiking event from electrode channels that do not factor in the ensemble is not active at all and spiking from useful channels but not in selected ensemble is stopped after sorting. The power consumption scales with patient movement intent.

The target clock frequency of the NSP is the same as the sample frequency of the ADC, which is 30 kHz in our system. Our detecting, feature extracting, sorting, and decoding models exhibit

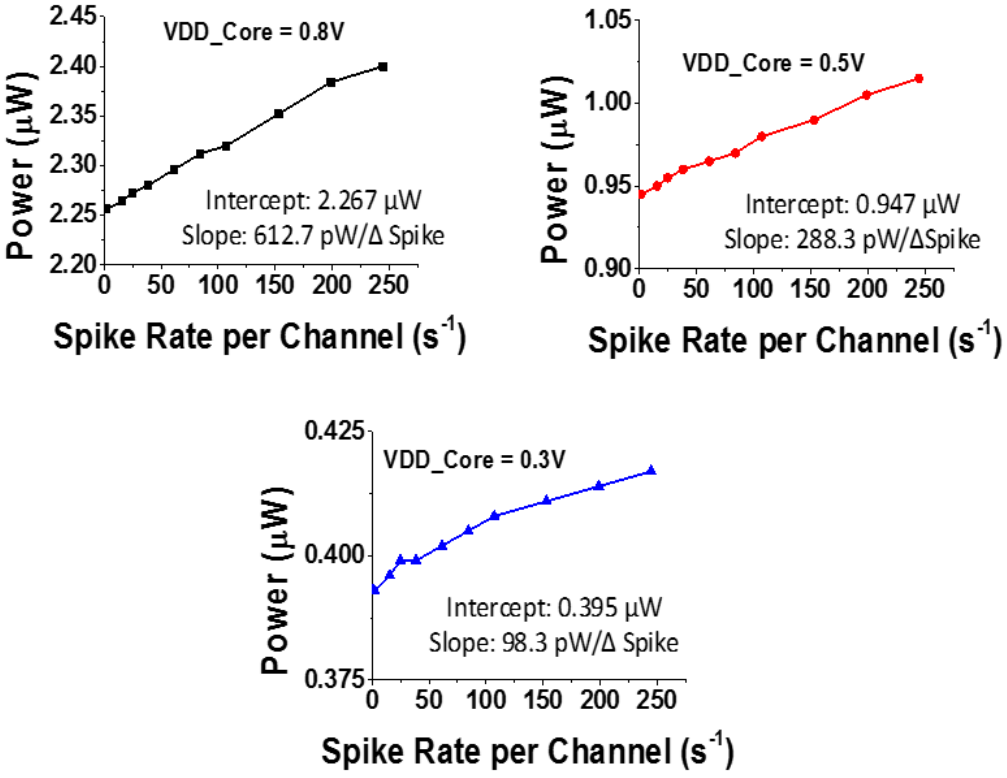


Figure 3.10: Spike rate dependency of the NSP power.

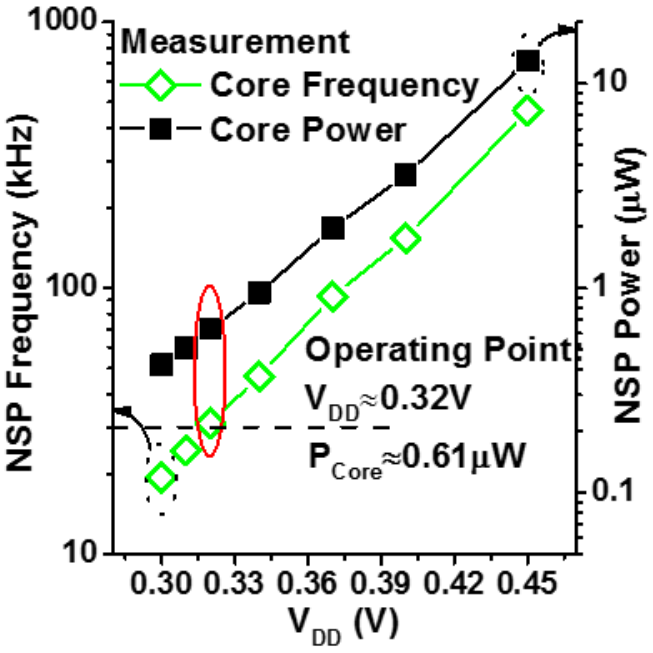


Figure 3.11: Power and performance of the NSP.

substantially small computational complexity and therefore can easily meet the 30-kHz timing requirement. This invites us to scale supply voltage to the subthreshold level of 0.32 V (Figure 3.11), at which the 96-channel NSP consumes only 0.61 μW.

In Table 3.3, we compare the NSP to the state-of-the-art BCI processors [24, 26, 8, 33, 34]. The proposed NSP achieves 21× smaller power dissipation than [26]. It also demonstrates the highest level of integration, namely the first end-to-end integration of neural signal processing at better accuracy over prior arts.

3.7 Summary

In this chapter, we present a nanowatt neural spike processor for a movement-intention-decoding neural interface implant. We devise/optimize algorithms and architecture for hardware and energy cost. Our design provides substantial resource savings from prior arts. We verified our algorithm using data driven testing for spike sorting and intention decoding, and with additional boundedness analysis for the proposed ensemble observation model. Our proposed hardware architecture en-

Table 3.3: Comparison to prior BCI processors.

	This work	[24]	[26]	[8]	[33]	[34]
Process (nm)	180	65	65	65	130(sim)	180
No. of Channels	96	96	128	16	32	64
Detection Algorithm	AT*	AT	ICD*	AT	NEO*	NEO
Supervised Sort	Y	N	Y	N	N	N
Sorting Algorithm	Design tree	Bayesian	K-means	O-sort	GSKM*	C-sort
Spike Dataset	[9]	[9]	[7](sim)	[35](sim)	[36]	[7](sim)
Accuracy	89-99%	95%	77-87%	75%	91%	67-93%
Partial Decoding	Y	N	N	N	N	N
Core VDD (V)	0.32	0.6	0.54	0.27	1.2	1.8
Core Power/ch (nW)	6.3	1740	175	108.8	750	231.3
Core Area/ch (mm^2)	0.0194	0.12	0.003	0.07	0.023	0.094

*AT: absolute threshold; ICD: integer coefficient detector; NEO: Nonlinear energy operator
*GSKM: Gap statstics K-means

ables effective hardware sharing and event-driven architecture, thereby substantially reducing area and power dissipation. The NSP not only achieves the highest level of functional integration from spike detection to the intention decoding but also marks a record power efficiency.

Chapter 4: XNOR-SRAM: In-Memory Computing SRAM Macro for Binary/Ternary Deep Neural Networks

In this chapter, We present XNOR-SRAM, a mixed-signal in-memory computing SRAM macro that computes ternary-XNOR-and-accumulate operations in binary/ternary deep neural networks (DNNs) without row-by-row data access. The XNOR-SRAM bitcell embeds circuits for ternary XNOR operations, which are accumulated on the read bitline (RBL) by simultaneously turning on all 256 rows, essentially forming a resistive voltage divider. The analog RBL voltage is digitized with a column-multiplexed 11-level flash ADC at the XNOR-SRAM periphery. XNOR-SRAM is prototyped in a 65-nm CMOS, and achieves the energy-efficiency of 403 TOPS/W for ternary-XNOR-and-accumulate operations with 88.8% test accuracy for CIFAR-10 dataset at 0.6V supply.

4.1 Motivation

As deep convolutional neural networks (DCNNs) continue to demonstrate improvements in inference accuracies [37, 38, 39, 40, 41], deep learning is shifting towards edge computing. This development has motivated works on low-resource machine learning algorithms [42, 43, 44, 45, 46, 47, 48, 49], and their accelerating hardware [50, 51, 52, 53, 54]. The dominant operations in DCNNs are multiply-and-accumulate (MAC), which consumes the most 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 [51, 52, 53, 54, 55]. As a result, IMC has become increasingly appealing to DCNN acceleration.

IMC is the design approach that performs large-scale and highly-parallel computation inside the memory block without explicit row-by-row memory access. Recent IMC works [55, 56, 57,

58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69] show significant energy-efficiency and throughput advantages over conventional architectures.

4.2 IMC OVERVIEW

IMC refers to memory architectures that support computations that takes place inside the memory fabric, thus avoiding the energy intensive memory wall [70]. IMC works are not exclusively MAC accelerators. For instance, designs in [55, 64] and Neural Cache [65] support two-row logic operations. Some IMC works support machine learning algorithms other than neural networks, such as Ada-boost [57] and Random Forest [62]. In this section, we provide an overview of IMC designs for neural network acceleration.

4.2.1 Multi-bit weights in IMC designs

Conventionally, in a neural network, both its activations and weights are multi-bit values. Since the physical topology of memory bitcells are independent at the array level, multi-bit weight representation in IMC is implemented at circuit level instead of at architecture level. The following IMC works are designed to support multi-bit weight neural networks [67, 71, 72, 73]. For example, the Twin-8T design in [67] stores multi-bit weights in multiple SRAM cells, where these bitcells are numerically related to each other through the transistor width ratio in adjacent columns.

Alternatively, there are IMC designs based on emerging non-volatile memory technologies that can store multi-bit weight in a single bitcell, such as Phase Change Memory (PCM) [71, 72] and Resistive RAM (RRAM) [73]. However, these devices exhibit variability and nonlinearity limitations [74, 75].

4.2.2 Binary weights in IMC design

One of the chief algorithmic advancements that address the difficulty in multi-bit weight IMC design is the binary weight network (BWN) [44], in which the network weights are binarized but the input and output activations can remain multi-bits. Weight binarization relaxes the storage

constraint and makes storing weights straightforward.

A subset of BWNs called binary neural networks (BNN) binarize both weight and activation to +1 and -1 such that multiplications can be represented by simple XNOR [45, 46, 47]. While early BNNs are not very accurate, recent advances [48] show BNN can approximate high MAC precision via weight duplication akin to stochastic computing, achieving comparable accuracy with multi-bit DCNN at lower hardware cost. As a result, many IMC works [57, 58, 61, 62, 66, 68, 69], are designed to support BWNs or BNNs.

4.2.3 Multi-bit activation in IMC designs

Computing with multi-bit activations can be done in digital or analog domains. Digital representation can be done through multi-cycle operations. For the analog representation of input activations, IMC designs have to perform the digital-to-analog conversion (DAC) before actual computing can take place. Several IMC works have employed DAC techniques to preprocess the input activations, expressing the analog value in voltage or current.

Using the DAC circuits with voltage output, the Twin-8T IMC design [67] can represent up to four levels of input activations with four distinct voltage levels on the wordlines. XNOR-SRAM [69] can similarly support up to three voltage levels to perform ternary MAC computation. Voltage-level-based DACs generally have limited resolution.

Current-based DAC circuits have been employed in designs like [57] and Conv-SRAM [58]. Using pulse-width-modulation (PWM), the input activation is first converted into a pulse. Within the window of the generated pulse, a charging current is then applied to a capacitive element. It has two main design challenges: 1) the PWM needs to be linear and 2) the current source needs to be constant. Both of these challenges would require area/energy tradeoff.

4.3 XNOR-SRAM Macro Design and Optimization

We propose 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. 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 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).

4.3.1 XNOR-SRAM Bitcell Design

Figure 4.1 presents the XNOR-SRAM architecture, which can map convolutional and fully-connected layers of CNNs and multi-layer perceptrons (MLPs). 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.

Figure 4.2(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 XNOR mode (and memory mode read); T11 and T12 power-gate the PU/PD circuits when the corresponding column is disabled. Figure 4.2(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 Figure 4.2(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.

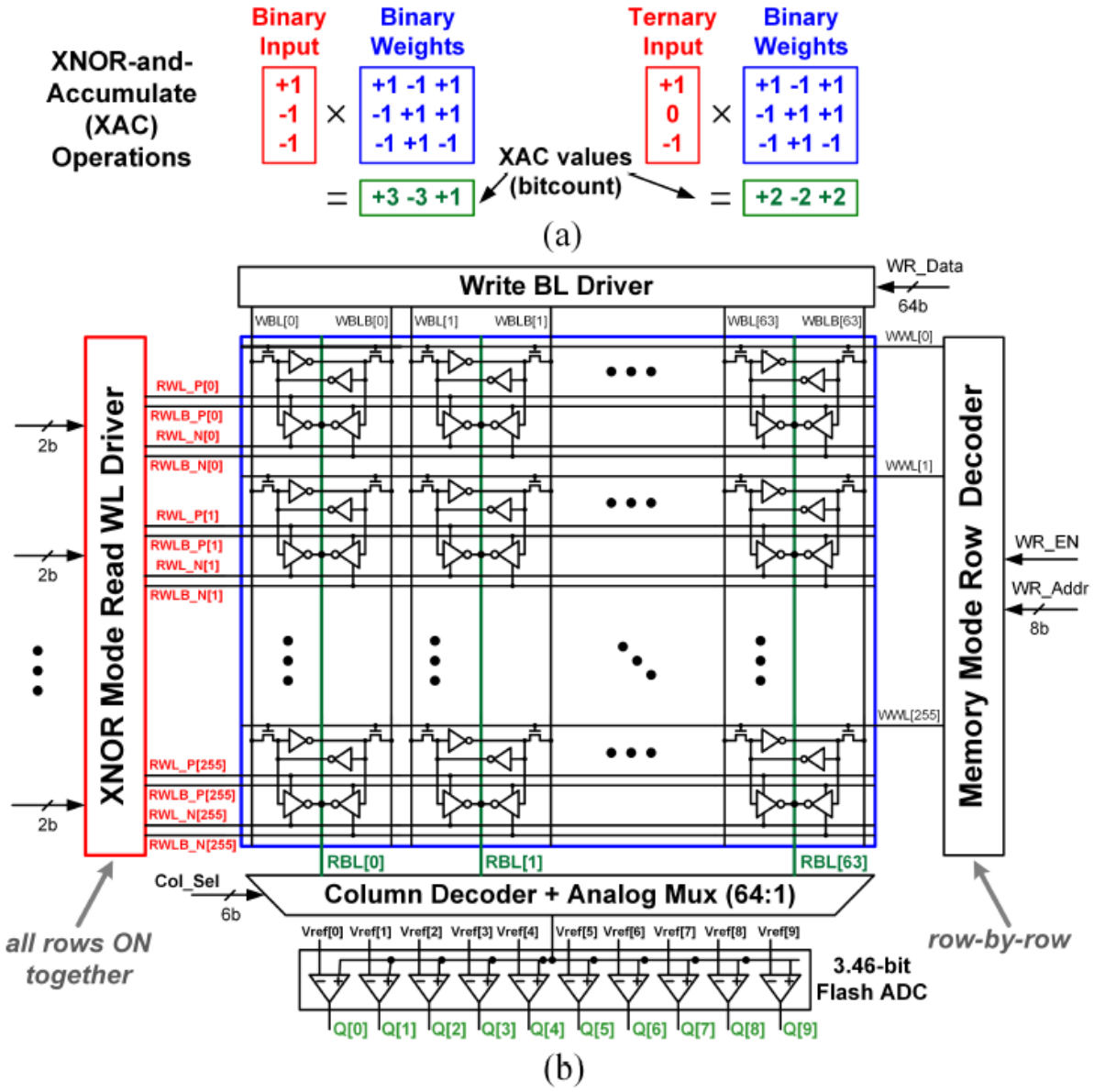
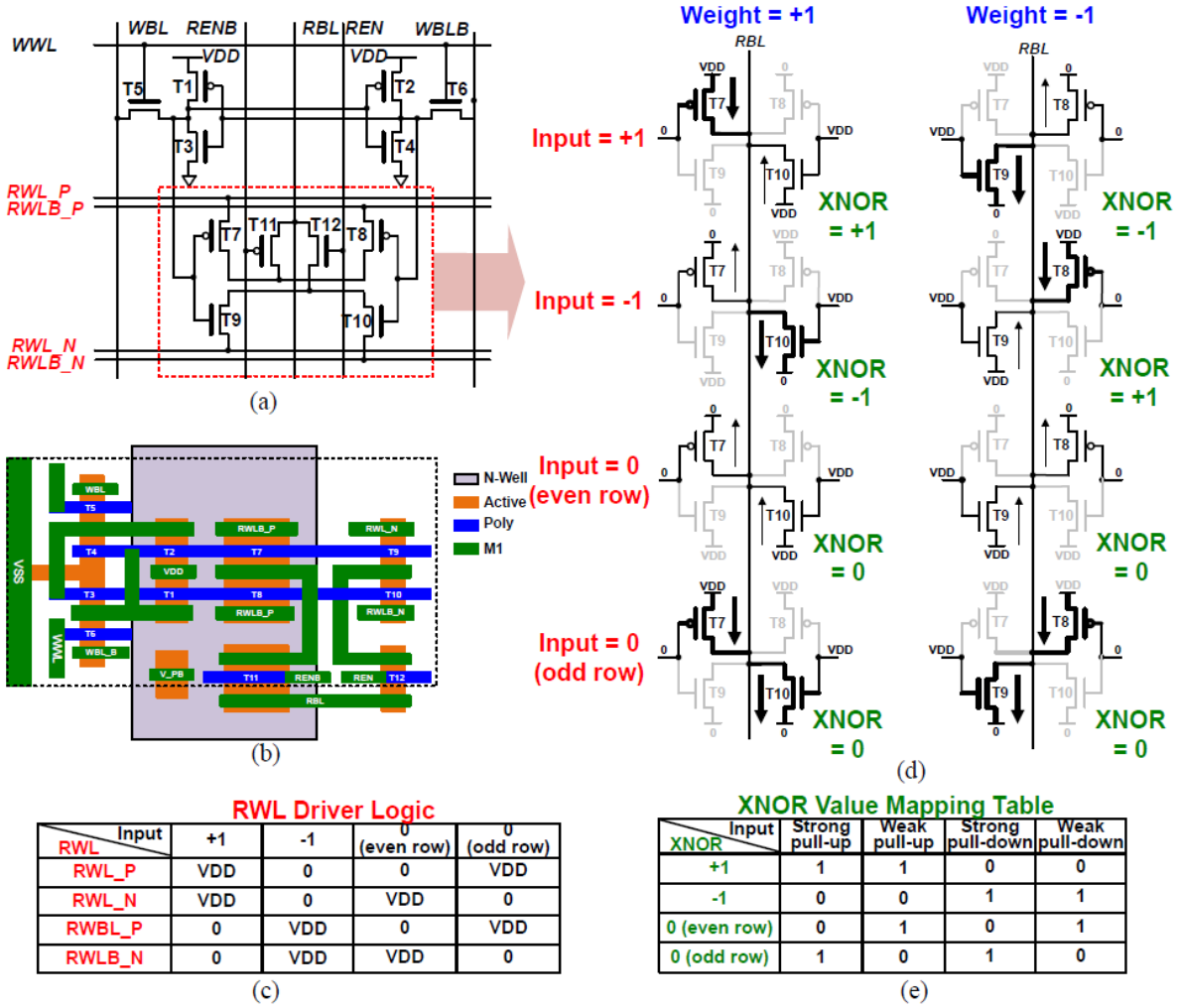


Figure 4.1: (a) XAC operation illustration and (b) the proposed XNOR-SRAM macro architecture.



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 4.2: XNOR-SRAM bitcell design and XNOR-ACC operation with ternary inputs/activations and binary weights. Bitwise ternary-XNOR output from each bitcell forms pull-up/-down paths on the RBL voltage, which represents the XAC value.

4.3.2 XNOR-SRAM 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 Figure 4.2(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 subject to PVT variation. Our design addresses this by making the bitcell array PMOS body bias tunable.

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 (4.1), 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 (4.2). As described in Figure 4.2(c), the bitwise XNOR output of '+1' and '-1' results in two PU and two PD paths, respectively. This is illustrated in Figure 4.3, and V_{RBL} can be represented as (4.3) with the resistive divider. Using (4.1) and (4.2), V_{RBL} can be formulated as (4.4), 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.

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 (4.4). 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 ((4.5)). Since those z bitcells do not contribute to the XAC output, the equations for the XAC value are identical for the

$$N = u + d, \tag{4.1}$$

$$XAC = u - d, \tag{4.2}$$

$$V_{RBL} = \frac{2u}{2u + 2d}, \tag{4.3}$$

$$V_{RBL} = \frac{XAC + N}{2N}, \tag{4.4}$$

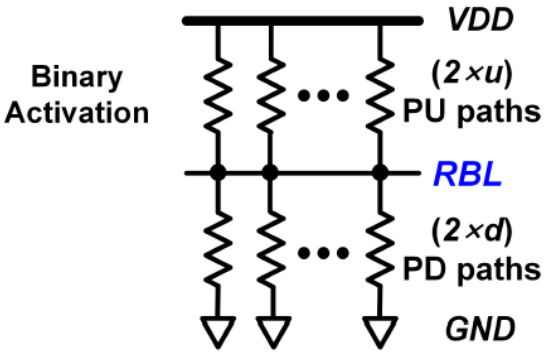


Figure 4.3: PU/PD paths for V_{RBL} with binary activations.

binary and ternary activation case (i.e., (4.2) and (4.6)). To maintain the same linear relationship between V_{RBL} and N as in (4.4), the z bitcells should contribute z PU and z PD circuits. This is illustrated in Figure 4.4.

$$N = u + d + z, \tag{4.5}$$

$$XAC = u - d, \tag{4.6}$$

$$V_{RBL} = \frac{2u + z}{2u + 2d + 2z}, \tag{4.7}$$

$$V_{RBL} = \frac{XAC + N}{2N}, \tag{4.8}$$

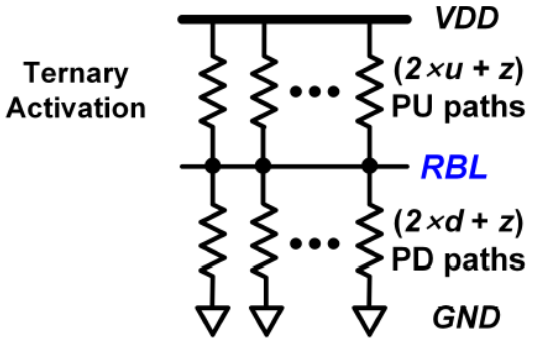


Figure 4.4: PU/PD paths for V_{RBL} with ternary activations.

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 Figure 4.2(c). Note that having z cells to exhibit no PU and PD paths (i.e., turning off '0' activation rows) will make (4.8) deviate from equation (4.4), 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 (Figure 4.2(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.

4.3.3 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 [76], 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, 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 (Figure 4.5), 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 (Figure 4.5). Note that the non-linearity of the PD and PU resistance makes the V_{RBL} transfer function non-linear, placing

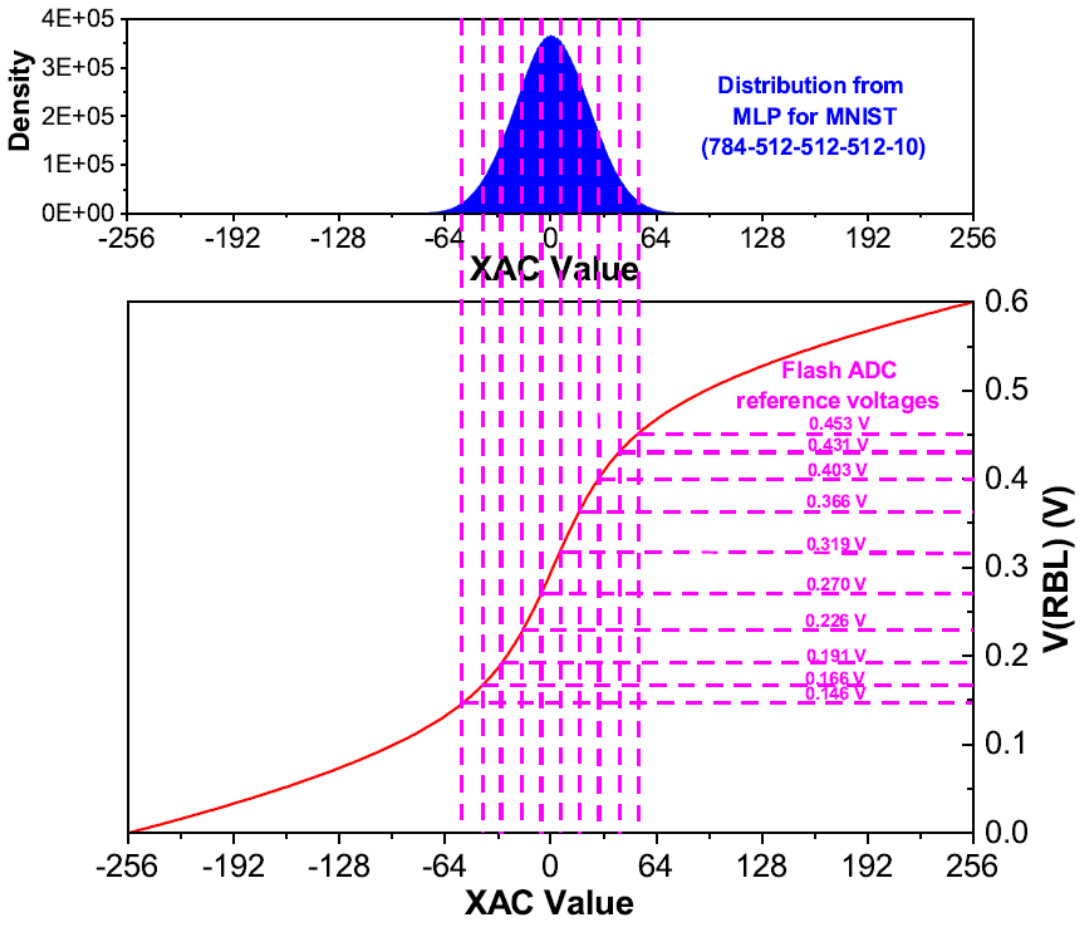


Figure 4.5: XAC is mapped to V_{RBL}. The confined linear quantization scheme is shown, along with the corresponding 10 reference voltages for the 11-level flash ADC.

Vrefs non-uniformly, as shown in Figure 4.5.

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). Figure 4.6 shows the simulation 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 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.

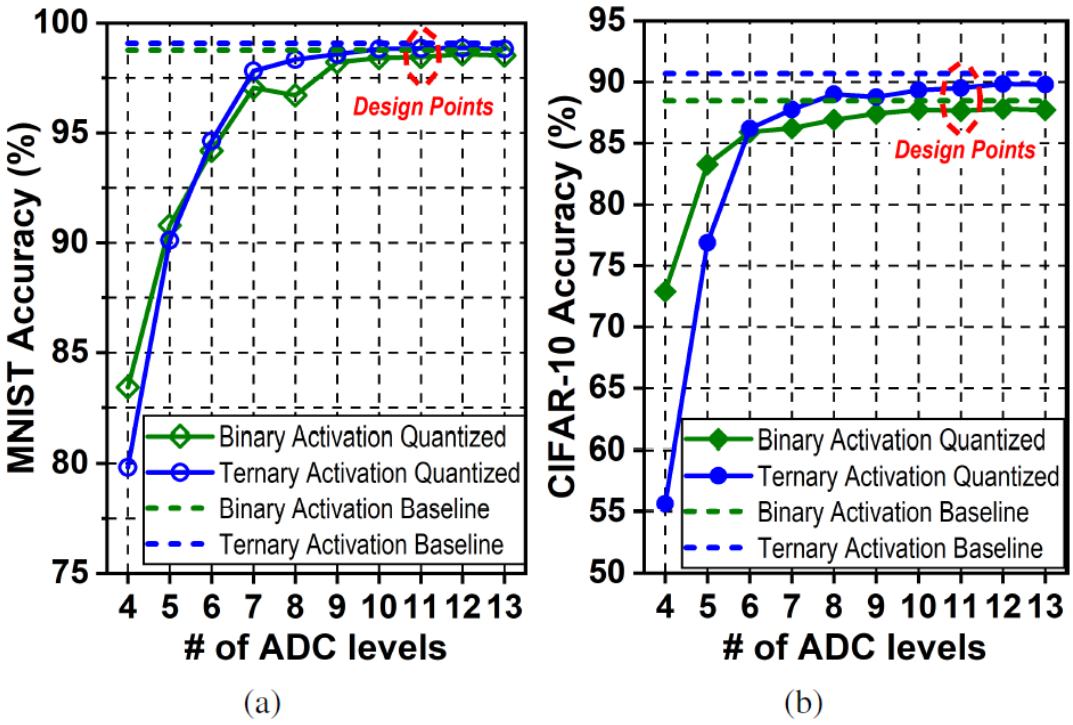


Figure 4.6: The accuracy of the MLP trained for MNIST and that of the CNN for CIFAR-10 as a function of ADC levels.

4.4 Measurement Results

We prototyped the proposed XNOR-SRAM macro in a 65-nm CMOS (Figure 4.7(a)). The area and power breakdowns are presented in Figure 4.7(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 resistive voltage divider formed for XAC evaluation.

4.4.1 Energy Consumption and Performance Measurements

We measure 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 (Figure 4.8). 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 to XAC value near 0 poses the worst case for energy-

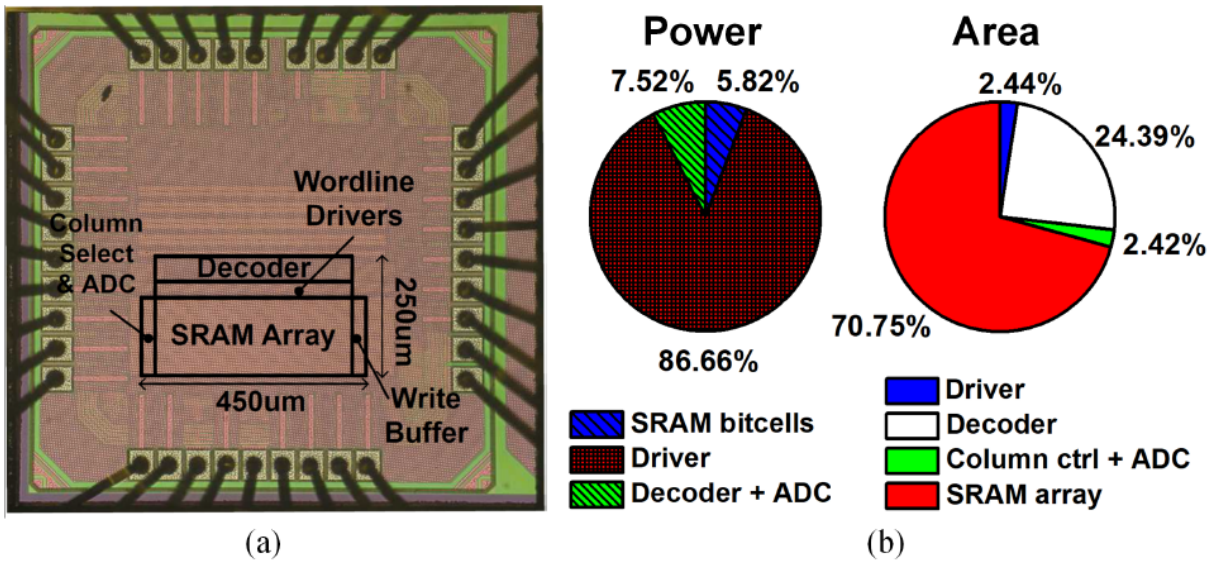


Figure 4.7: (a) 65nm XNOR-SRAM prototype chip micrograph. (b) Power and area breakdown.

efficiency. 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. Figure 4.9 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.

4.4.2 Variability and Compensation

Figure 4.11 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 +3σ 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 providing a knob to tune the body bias for PMOS transistors in the XNOR-SRAM array at marginal area/power penalty (Figure 4.12).

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

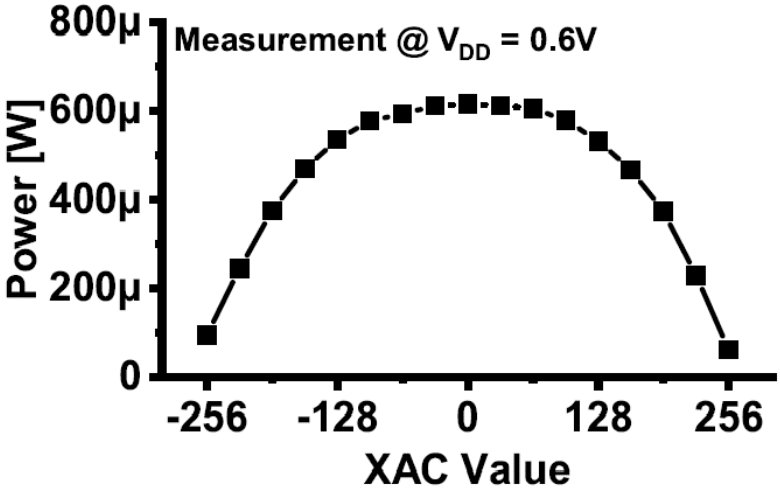


Figure 4.8: Data dependent XNOR-SRAM power.

Table 4.1: V_{RBL} variance of a single column at 1.0V and 0.6V supply extracted from post-layout Monte-Carlo simulations.

	Δ V_{RBL} (mV)		
VDD (V)	from mismatch	from IR drop	from both
1.0	2.71	36.6	36.8
0.6	7.09	6.06	9.33

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 compensation of the ADC offset, there are two remaining major 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.

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

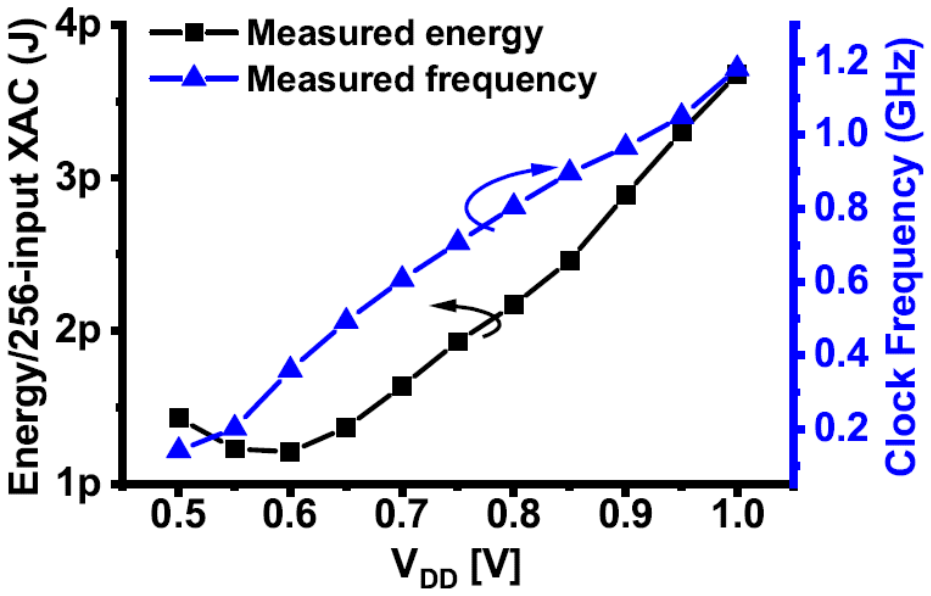


Figure 4.9: Energy and frequency scaling with supply voltage.

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 4.1 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.

4.4.3 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 for each XAC value, we estimated the probability distribution of the ADC output as function of the XAC value (Figure 4.13). We iterated this experiment at four different supply voltages of 1.0V, 0.8V, 0.6V, and 0.5V.

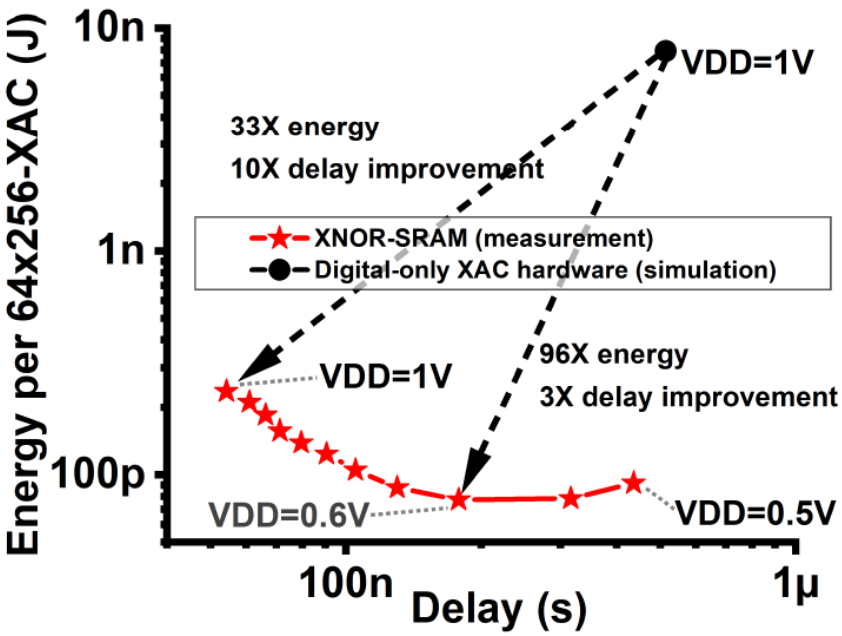


Figure 4.10: Energy and delay comparison with digital baseline.

Counter-intuitively, Figure 4.13 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

$$V_{RBL}(X, V_{DD}) = \bar{V}_{RBL}(X, V_{DD}) \pm \Delta V_{RBL}(X, V_{DD}), \tag{4.9}$$

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 combinations 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.

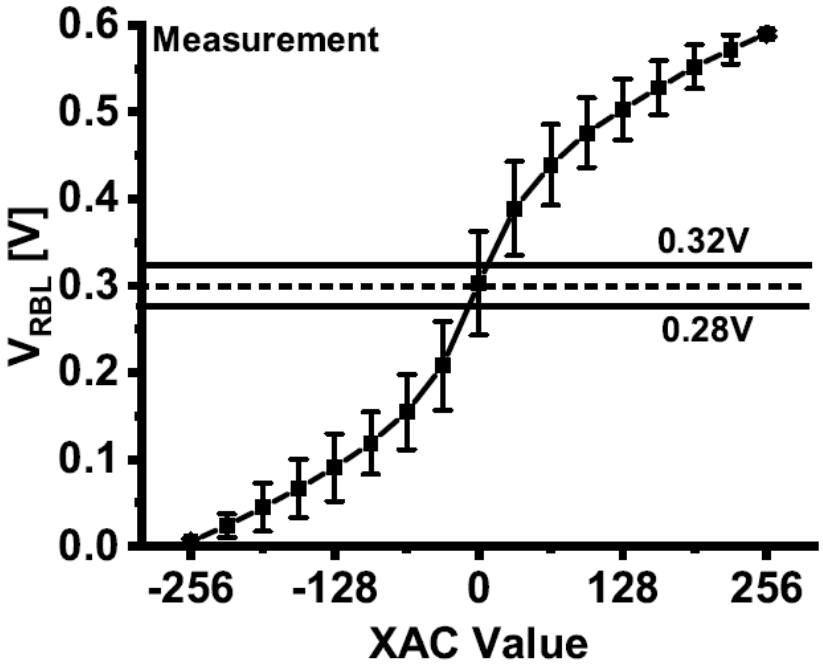


Figure 4.11: Measured transfer function and variability.

Voltage scaling of RBL voltage variance

As shown in Table 4.1, 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.

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 4.14. 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. 4.13.

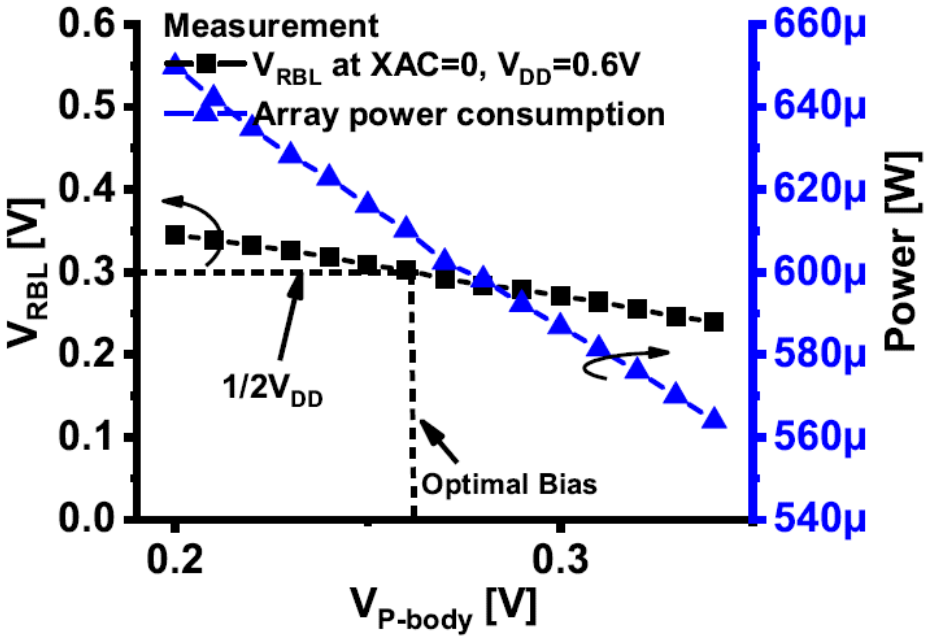


Figure 4.12: Body bias tuning for PMOS/NMOS mismatch.

4.4.4 Strategy for Mapping DNNs onto XNOR-SRAM arrays

A weight-stationary mapping scheme optimized for data reuse is adapted in our experiments. The mapping of FC layer weights in XNOR-SRAM 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 3x3x256 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 3x3 kernel filters is shown in Figure 4.15.

4.4.5 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 [45] has six convolutional layers and three fully-connected (FC) layers: 128C3-

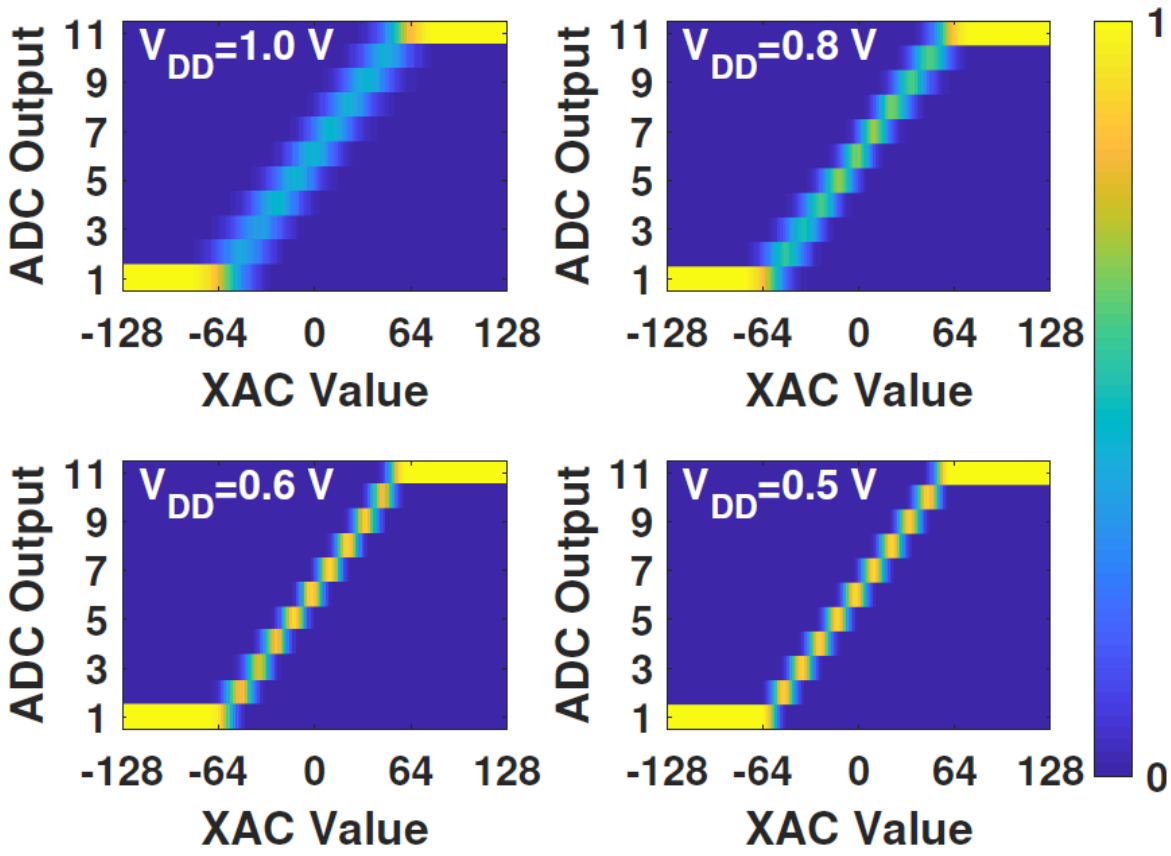


Figure 4.13: Measured ADC output probability distribution as a function of XAC value at V_{DD} of 1.0V, 0.8V, 0.6V, and 0.5V.

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 [45] 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. 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[77].

The MNIST accuracy results were obtained entirely from measurements, while the CIFAR-

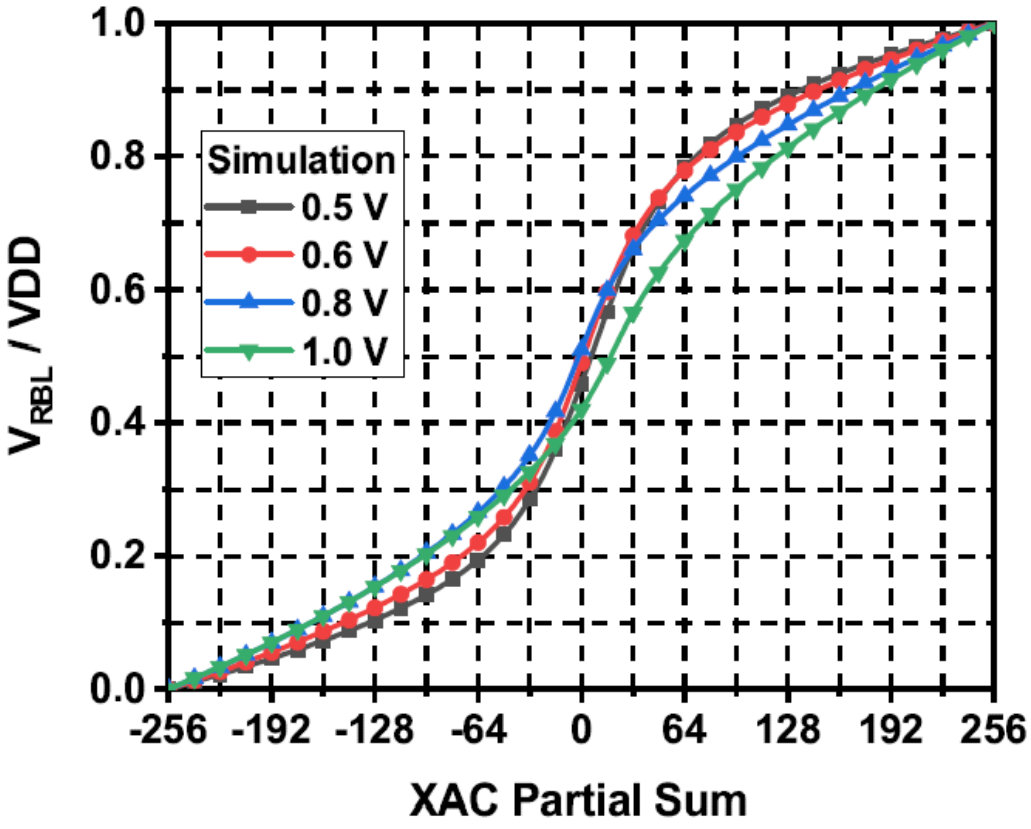


Figure 4.14: Normalized transfer function at different V_{DD}.

10 accuracy results were obtained from our measurement based simulation framework (illustrated in Figure 4.16) due to limited scan chain throughput of the prototype chip. Employing the same methodology used to generate the probability distribution in Figure 4.13, 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 bit-count, then this mapping was kept for all the 10k CIFAR-10 test images. The 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 [45]. Our Python simulation program repeated 20 runs with different random seeds, and the average accuracy values are reported.

Table 4.2 summarizes the measured accuracy results of MLP for MNIST and VGG/ResNet-

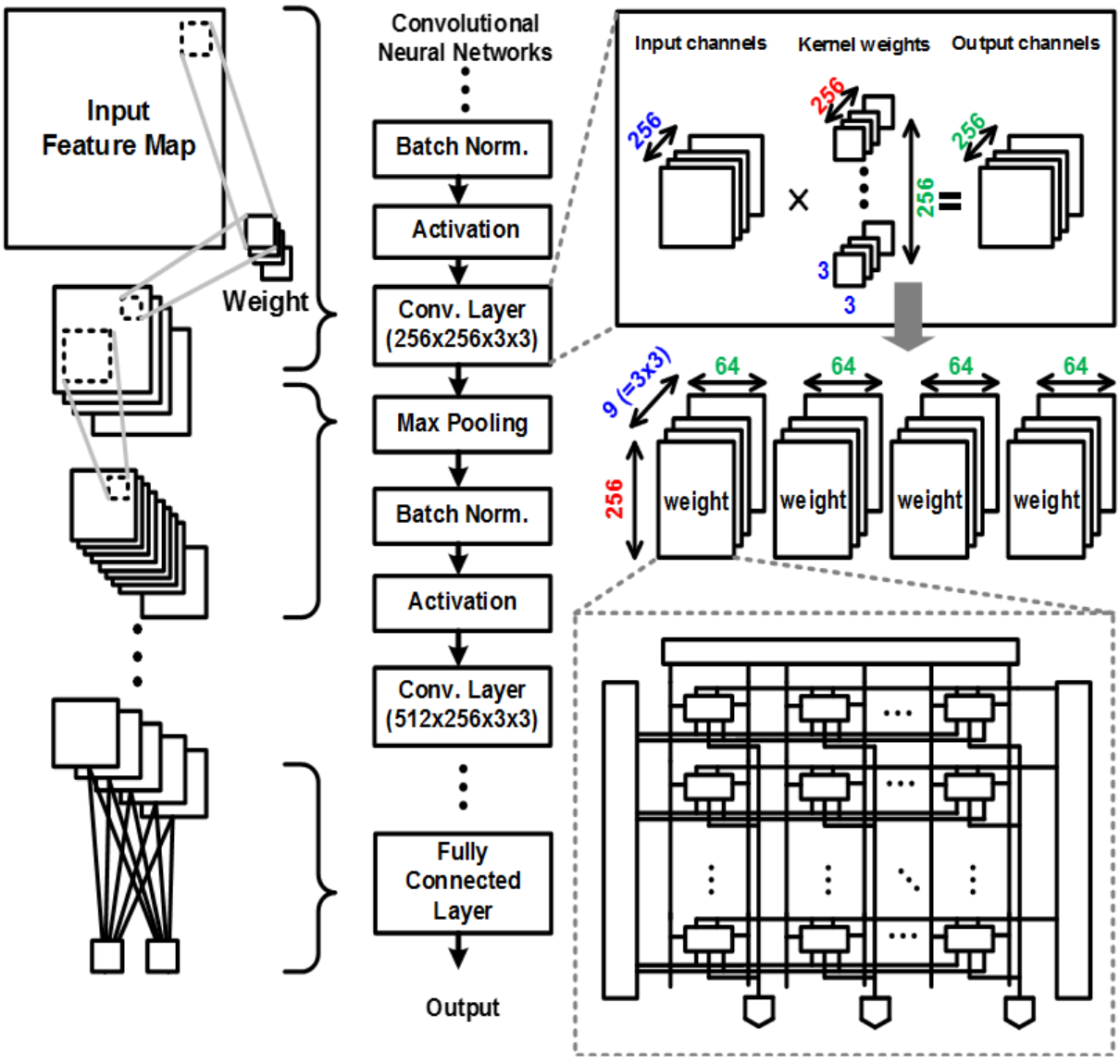
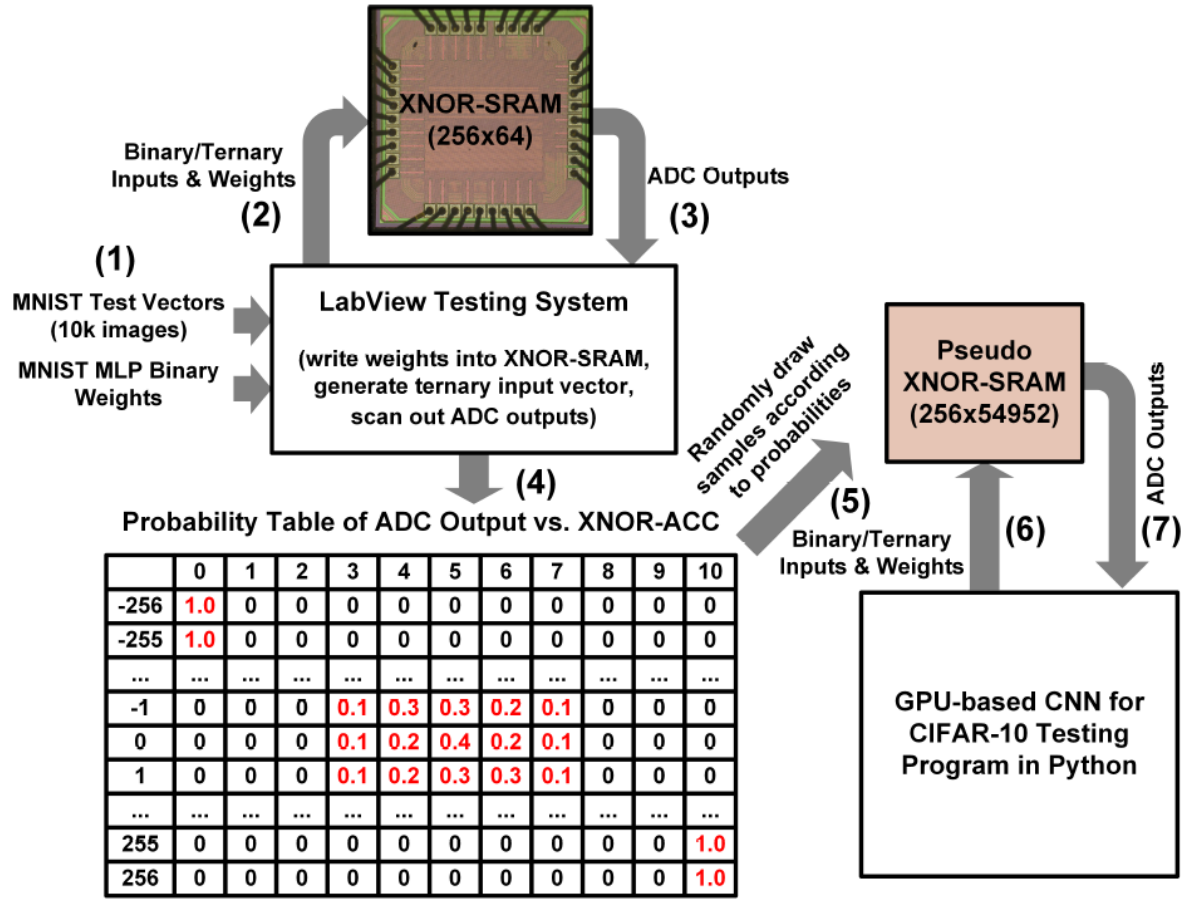


Figure 4.15: Mapping convolutional neural networks to XNOR-SRAM-based in memory computing.



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

14 CNNs for CIFAR-10 datasets with binary/ternary activations at 0.6V chip #2. 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 (Sec. 4.3.2).

4.5 Summary

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. We present an IMC SRAM macro titled XNOR-SRAM that computes ternary-XNOR-

Table 4.2: Measured MLP (for MNIST) and CNN (for CIFAR-10) accuracy summary using XNOR-SRAM 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 Baselien Accuracy	98.77%	99.07%	88.60%	90.70%	89.61%	90.80%
HW Measured Accuracy	98.65%	98.84%	87.23%	88.78%	85.72%	87.05%

and-accumulate operations in binary/ternary MLP and CNNs with high energy-efficiency and high accuracy. Our 256x64 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. The prototype achieves energy-efficiency of 403 TOPS/W for XAC operations and 88.8% test accuracy for CIFAR-10 dataset.

Chapter 5: C3SRAM: An In-Memory-Computing SRAM Macro Based on Robust Capacitive Coupling Computing Mechanism

This chapter presents C3SRAM, another in-memory-computing SRAM macro. The macro is of an SRAM module with the circuits embedded in bitcells and peripherals to perform hardware acceleration for neural networks with binarized weights and activations. The macro utilizes analog-mixed-signal capacitive-coupling computing to evaluate the main computations of binary neural networks, binary-multiply-and-accumulate operations. Without needing to access the stored weights by individual row, the macro asserts all its rows simultaneously and forms an analog voltage at the read bitline node through capacitive voltage division. With one ADC per column, the macro realizes fully-parallel vector-matrix multiplication in a single cycle. The network type the macro supports and the computing mechanism it utilizes are determined by the robustness and error tolerance necessary in analog-mixed-signal computing. The C3SRAM macro is prototyped in a 65-nm CMOS. The prototyped macro is 256x64 in size and computes 64 256-input bMAC in parallel. The macros can be used in a modular fashion to support networks of arbitrary size. It demonstrates an energy efficiency of 672 TOPS/W and a speed of 1,638 GOPS (20.2 TOPS/mm^2), achieving 3,975X better energy-delay product than the conventional digital baseline performing the same operation. The macro achieves 98.3% accuracy for MNIST and 85.5% for CIFAR-10, which is among the best in-memory-computing works in terms of energy efficiency and inference accuracy trade-off.

5.1 Motivation

As detailed in 4.1, IMC designs show significant energy-efficiency and throughput advantages over conventional architectures. However, 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 [76].

Analog MAC can be broadly placed in two categories. 1) Current domain computing techniques include resistive voltage divider, discharging rate; 2) charge domain computing techniques include charge sharing, capacitive voltage divider.

XNOR-SRAM bitcells would each turn on pull-up/pull-down transistors according to activation input and stored weights. These paths are applied to the same MAC bitline, creating a resistive voltage divider. The non-linearity of transistor resistance creates high gain in the desired transfer function region. However, it is at the cost of high crowbar current and device variability. [57] implements its MAC operation by (dis)charging the wordlines like PWM-based DAC. However, the current source is a simple transistor and thus non-linear. To compensate, the design uses additional transistors to calibrate current in various conditions.

[62] uses charge-sharing to performs bMAC. Each bitcell has an individual capacitor which is charged/discharged based on a bit-multiplication result. The capacitors are then tied together to share the charges, performing accumulation. Conv-SRAM [58] also uses charge-sharing to perform MAC, where the charges (one row at a time) are placed on the bitline and shared row-wise. However, the multi-bit activation is derived from PWM-based DAC, thus the charges being shared are already the result of current-domain computing.

The proposed C3SRAM macro uses capacitive coupling to compute bMAC in the charge domain. The detailed description of architecture and operational procedures is presented in the following section.

5.2 Architecture and Operation

5.2.1 Memory Array Operation

In this subsection, we present the C3SRAM architecture and the operations in detail. Figure 5.1 presents the architecture of the proposed macro. C3SRAM performs a fully-parallel vector-matrix multiplication of 256 binary inputs and 256x64 binary weights. The macro consists of a 256x64

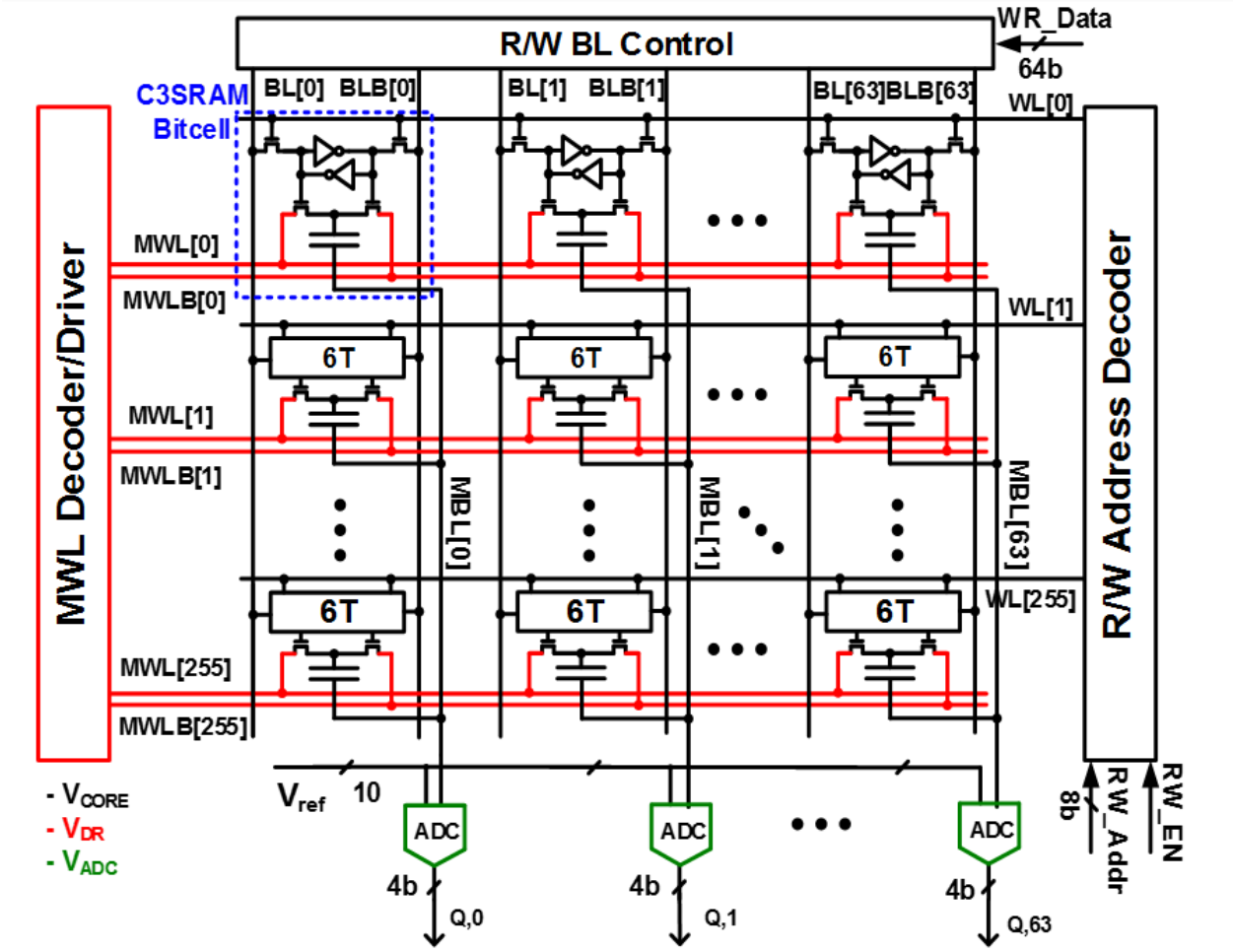


Figure 5.1: Architecture of C3SRAM in-memory computing macro.

memory cell array, SRAM peripherals for read/write operations, input activation decoder/driver, and per-column flash analog-to-digital converters (ADCs).

Figure 5.2 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 C_C (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

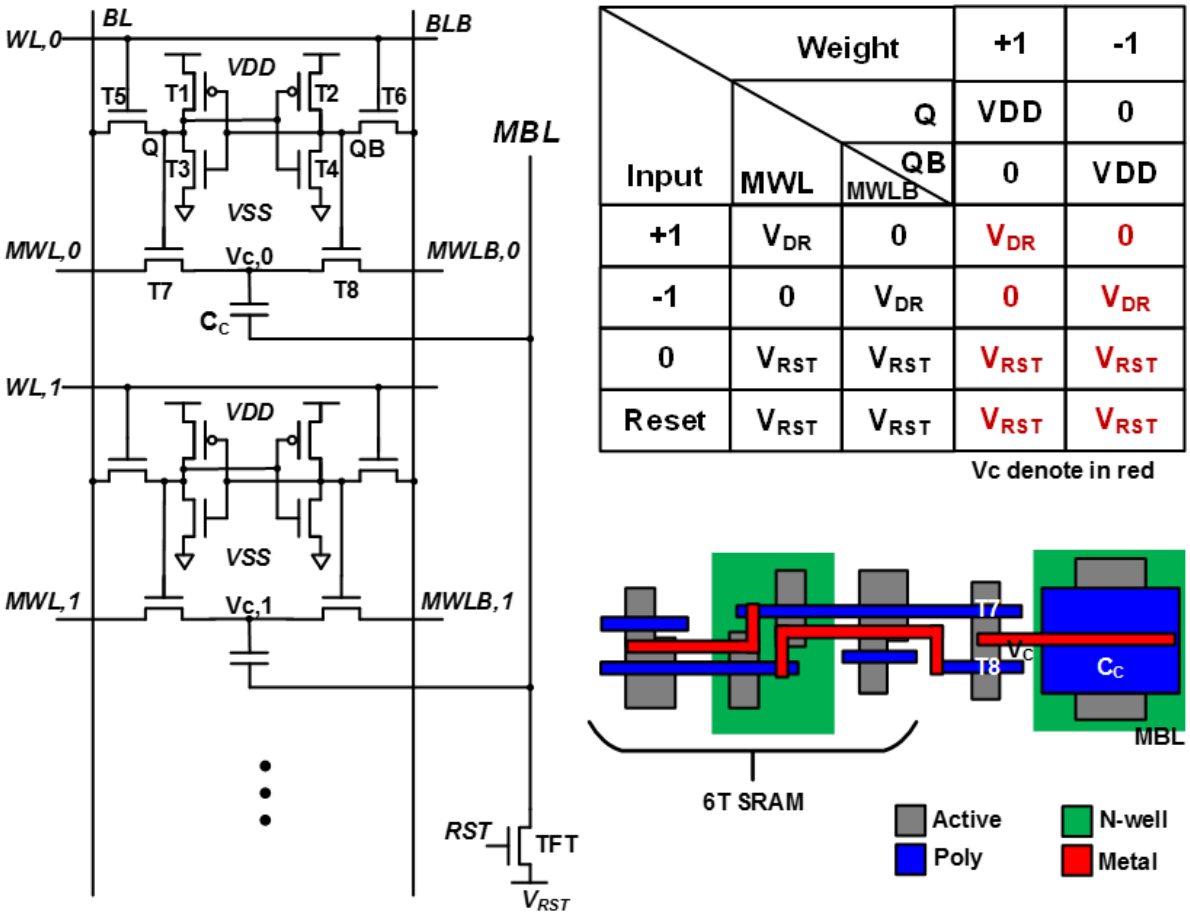


Figure 5.2: C3SRAM bitcell design and in-cell bMAC operand table.

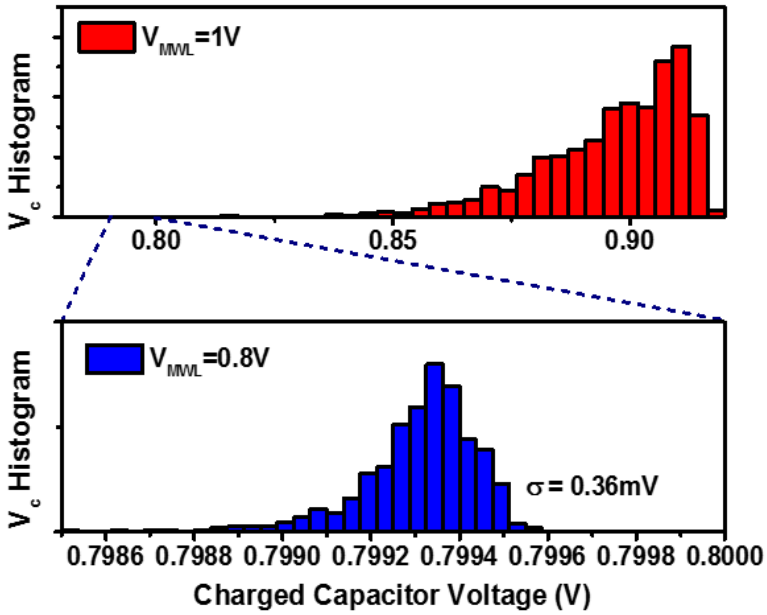


Figure 5.3: Threshold voltage variability effects on charged capacitor voltage.

transistors are NFETs, there would be highly variable threshold voltage (V_t)) drop across T7/T8 if the MWLs and the memory core have the same voltage source. To avoid the variability problem, we implement T7/T8 with LVT devices, and set a separate source V_{DR} to drive the MWLs, at 200mV lower than V_{CORE} (e.g., 0.8V V_{DR} for 1V V_{CORE}). We ran Monte Carlo SPICE simulations including only C_C, T7, and T8 to isolate the variation of the pass transistors' effect on capacitor charge. As shown in Figure 5.3 histograms, the x-axis (note the different scales) shows the voltage level at the V_C node. Given the 200mV margin, the variation of V_C has the small sigma of only 0.36mV. T7/T8 decouple memory function and compute function to avoid potential read and write disturb [57, 61].

The bMAC operation is shown in Figure 5.4. 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 V_{RST} =0.5V_{DR}. V_{RST} 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 and MWLB are likewise reset to V_{RST} 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 Figure 5.4 bottom left.

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

$$I_C = C_C \frac{dV_{MWL(B)}}{dt} \tag{5.1}$$

The charge transferred from the bitcell to MBL is:

$$Q_{Ci} = \int_0^{t_1} I_C dt = \frac{1}{2} C_C V_{DR} \tag{5.2}$$

where t_1 is the time it takes VMWL to reach V_{DR}. The shared MBL voltage is set to:

$$V_{MBL} = C_C V_{DR} \sum_1^{256} \frac{XNOR_i}{256C_C + C_p} \tag{5.3}$$

where $XNOR_i$ is the XNOR output of the i-th bitcell and C_p 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 parallel-connected capacitors which are in turn connected in series between V_{DR} and VSS, forming a capacitive voltage divider as illustrated in Figure 5.4 bottom right.

MOSCAP's high capacitive density gives bMAC transfer curve a wider full-scale range (FSR) than using the less capacitively-dense MOMCAP [62]. 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. Figure 5.5(a) shows C_C change over temperature and gate voltage. Figure 5.5(b) shows the simulated transfer function of a capacitive voltage divider composed of C_C. 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

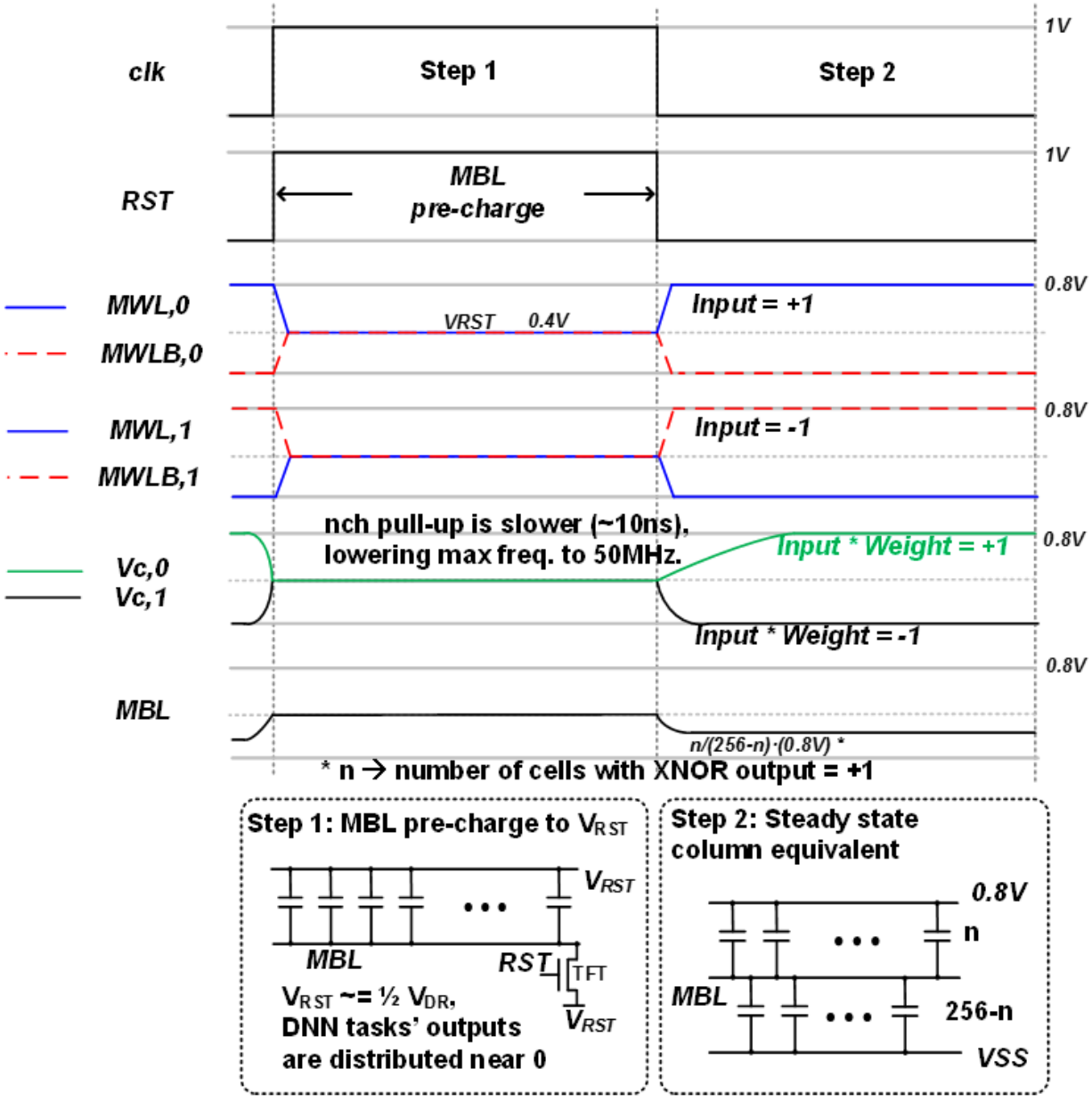


Figure 5.4: Capacitive coupling based in-memory computation of bMAC.

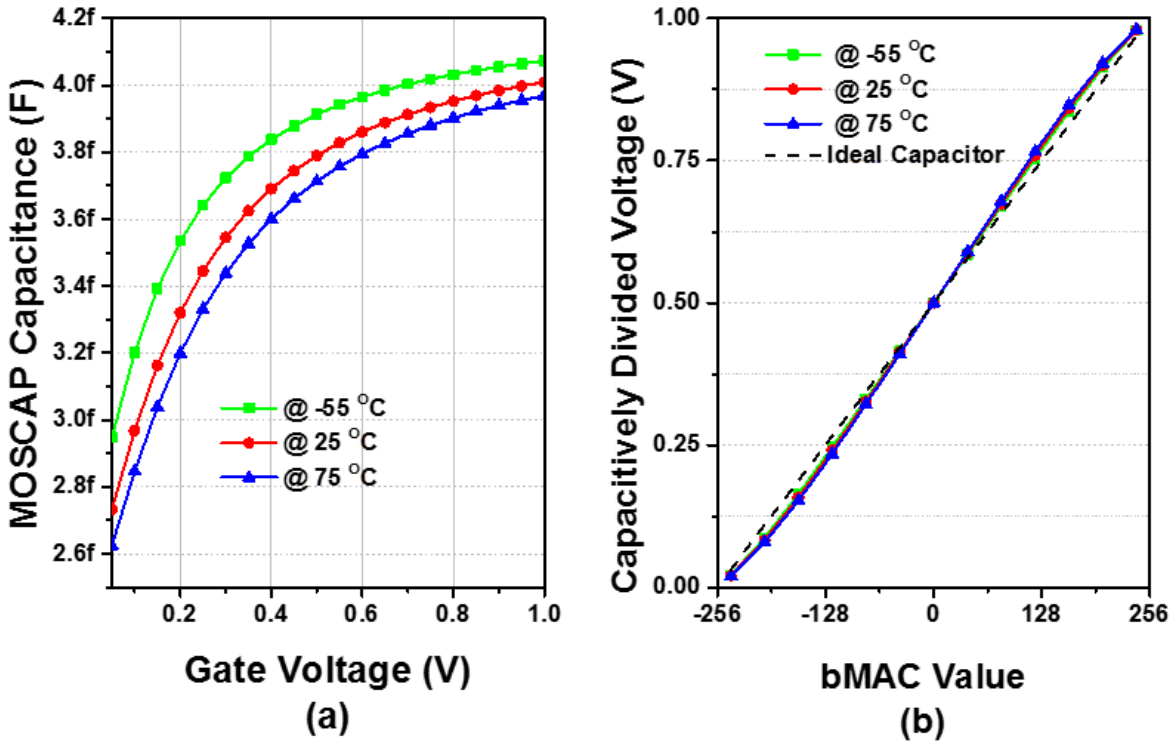


Figure 5.5: (a) MOSCAP capacitance at TT corner simulation shows variation across temperature as well as the gate voltage; (b) MOSCAP capacitive voltage divider transfer function at various temperatures.

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.

5.2.2 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 each consists of 10 double-sampling-based self-calibrating single-ended comparators (Figure 5.6 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 (Figure 5.6): during bMAC computation (step 2), MBL connects to the comparator input capacitor. The input and output of the first inverter are closed, placing

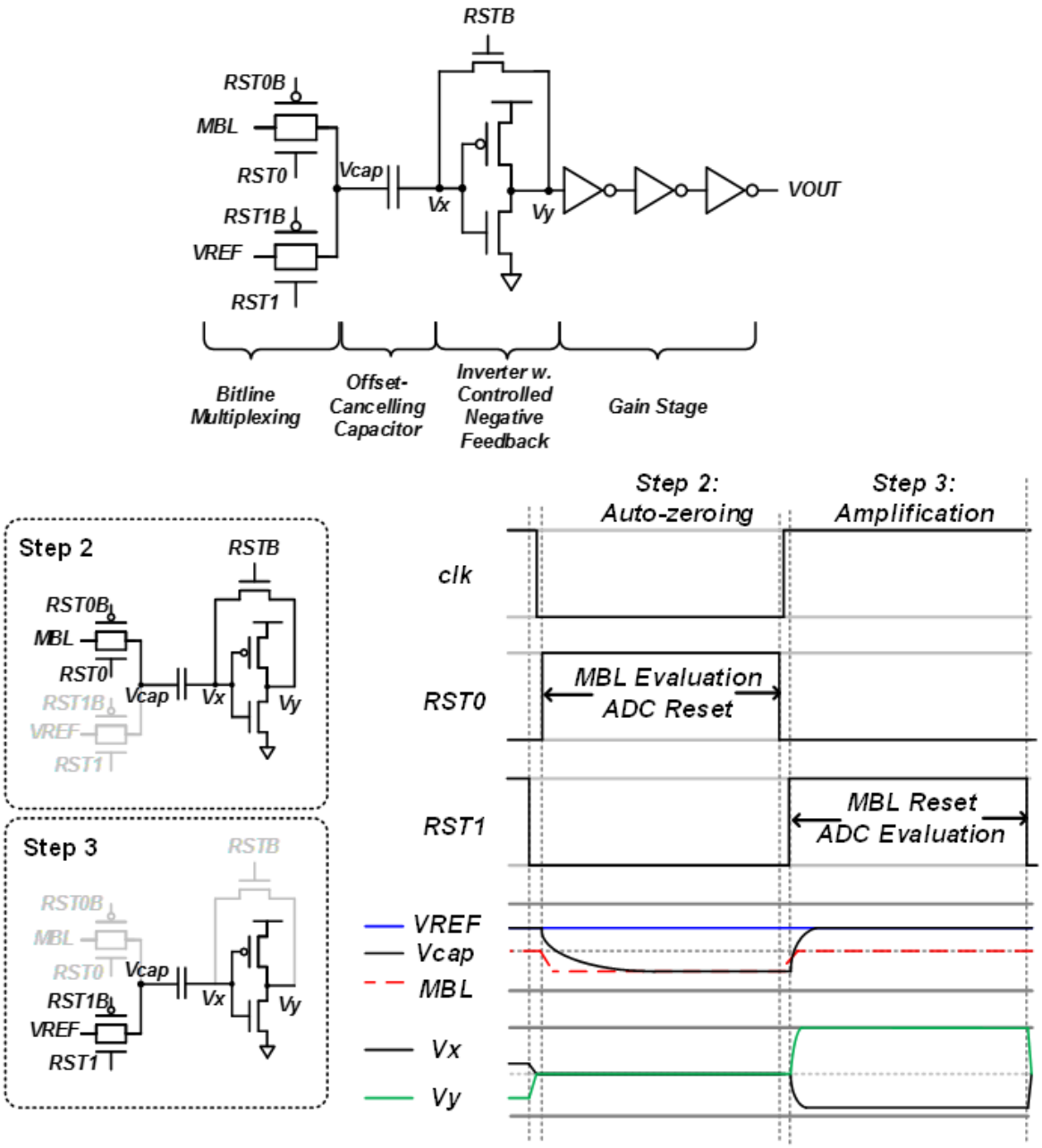


Figure 5.6: Operation of the double-sampling self-calibrating single-ended comparator.

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 V_{MBL} and V_{ref} 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.

5.2.3 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 pipeline of a half-cycle where step 1 (Figure 5.3) and step 3 (Figure 5.6) 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, shown in Figure 5.7. We implemented timing control circuitry with minimal delay elements to guarantee the correctness of the signals' order. The relevant transitions from steps 1/3 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, V_{MBL} 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 1/3 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

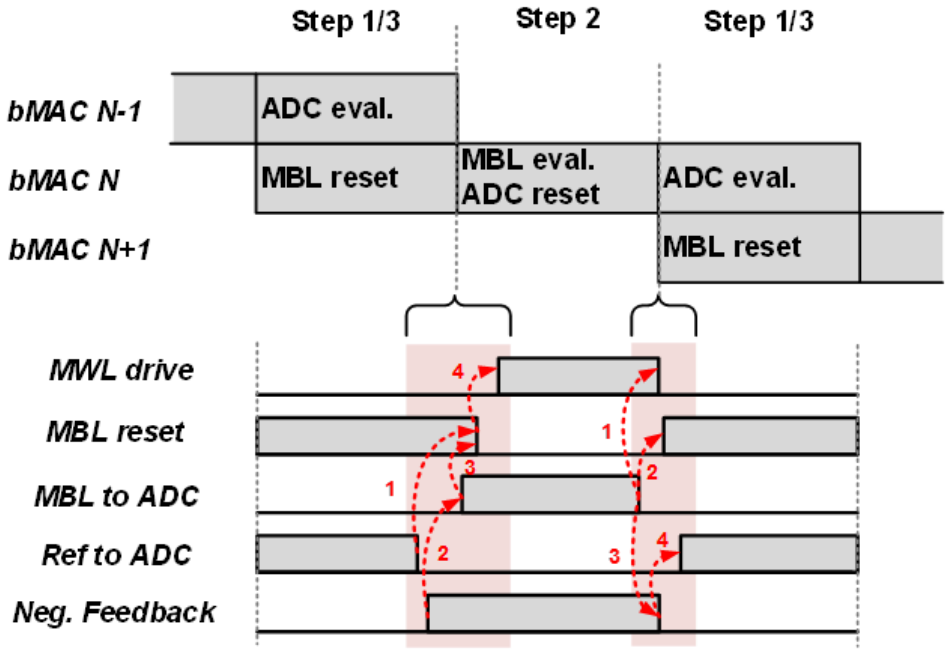


Figure 5.7: Signal transition order for reducing analog non-idealities.

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.

5.3 Algorithm Hardware Specification

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

5.3.1 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 [69]. One reason is that the analog representation of multi-bit activation requires additional domain conversion through DAC [67]. Another reason is that network models trained at higher precision are more sensitive to AMS error. As the study in

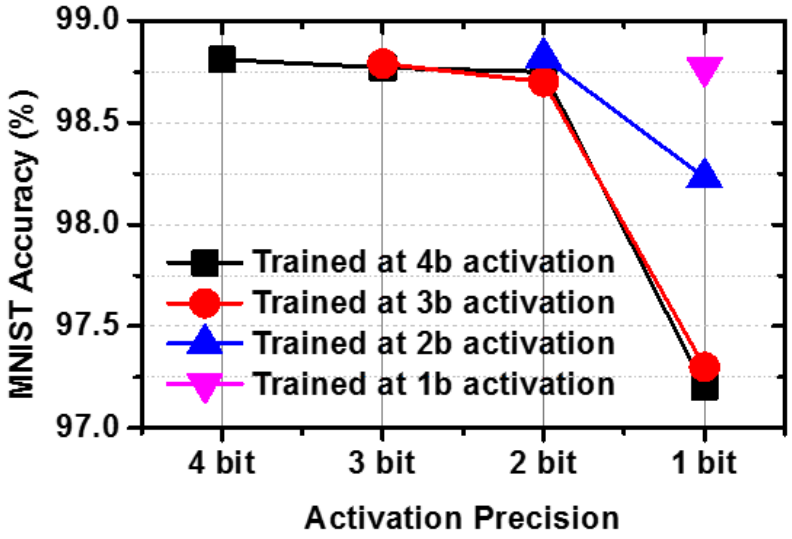


Figure 5.8: MLP on MNIST dataset inference accuracy losses at various levels of activation precisions.

[78] suggest, 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 [69], i.e., activations are fed in a bit-serial fashion and outputs are accumulated using digital adders. As shown in Figure 5.8, 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.

5.3.2 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.

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 [45], with six convolutional layers and three FC layers. For partial convolution results in these network models, we apply several quantization levels, successively reducing ΔV_{ref}. Figure 5.9 shows the effect of quantization on the task accuracy. We find that in both cases the accuracies reach saturation at 30mV ΔV_{ref} which corresponds to 5-bit resolution given the 640mV FSR. This is consistent with results in [58] 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 BWN/BNN activations functions ReLU/binary step. Figure 5.10 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. 10 bottom shows an illustration of an ideal transfer function and linear quantization levels in a confined range. The x-axis is the bMAC outputs; the y-axis is V_{MBL} corresponding to the bMAC output. In the voltage region corresponding to bMAC value from -120 to +120, only 11 reference levels (ΔV_{ref} = 30 mV) are needed to match 5-bit ADC resolution.

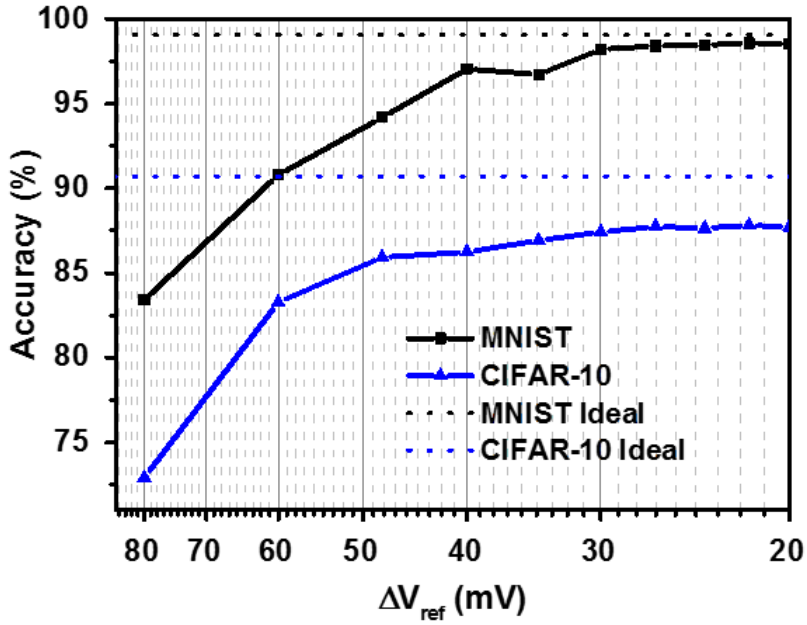


Figure 5.9: MNIST and CIFAR-10 inference accuracies increase as quantization resolution of pre-activation partial sum (256 input) increases.

5.4 Measurement and Analysis

The C3SRAM macro is implemented in a 65-nm CMOS. Figure 5.11 shows the micrograph of the test chip. The macro has a capacity of 16 kb at the footprint of 0.081 mm^2.

5.4.1 Energy and Throughput

The memory macro has 2kB capacity. For bMAC computations, the macro operates at a max 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 (V_{CORE}), 0.8V driver (V_{DR}), and 0.6V ADC (V_{ADC}), it consume 49 pJ excluding in/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 [69], 14X over XNOR-SRAM (Figure 5.12). Figure 5.13 left shows the power breakdown measurements: 38.7% of the total power is consumed by driving the MWL and bitcell capacitors, 22.0% by ADCs, and 39.3%

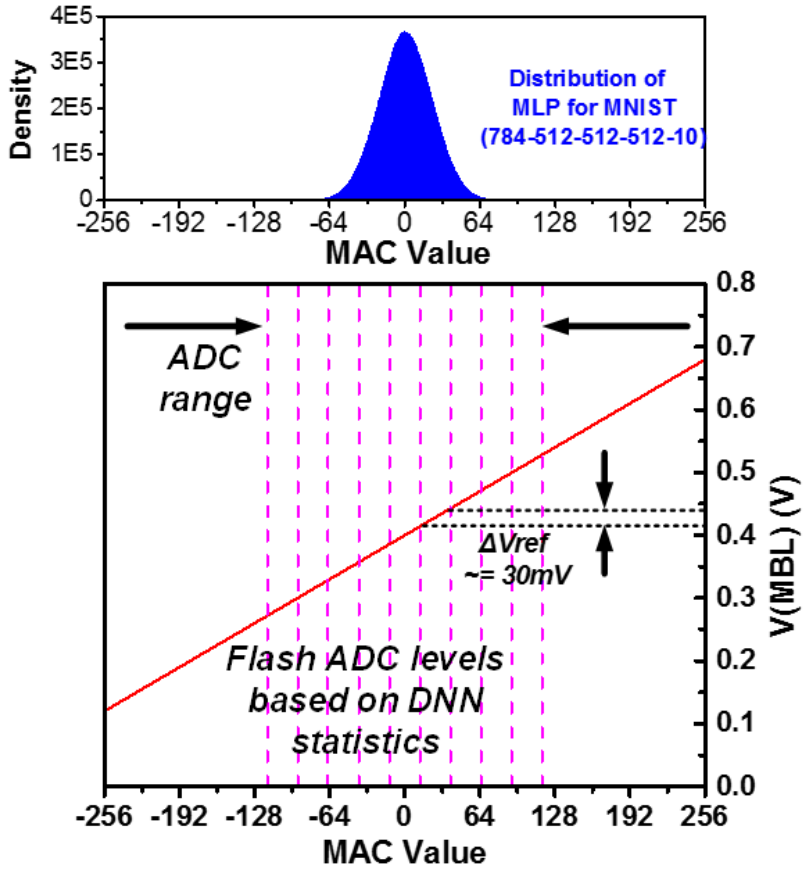


Figure 5.10: The bMAC distribution of MLP for MNIST and quantization in limited ADC range.

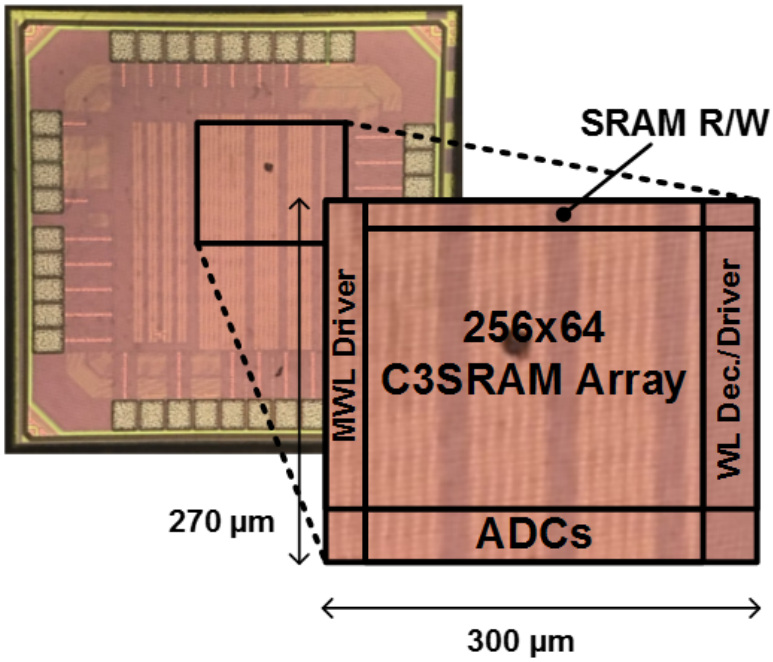


Figure 5.11: C3SRAM prototype chip micrograph.

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 Vesti [69] are also needed to prevent the memory macros from idling for input. 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 5.1 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.

5.4.2 Transfer Function

In this subsection, we measure and analyze the transfer function characteristics of C3SRAM under the same operating condition as Section 5.4.1. The transfer function is measured according these following steps. First we set all weights to zero, then apply known input pattern correspond-

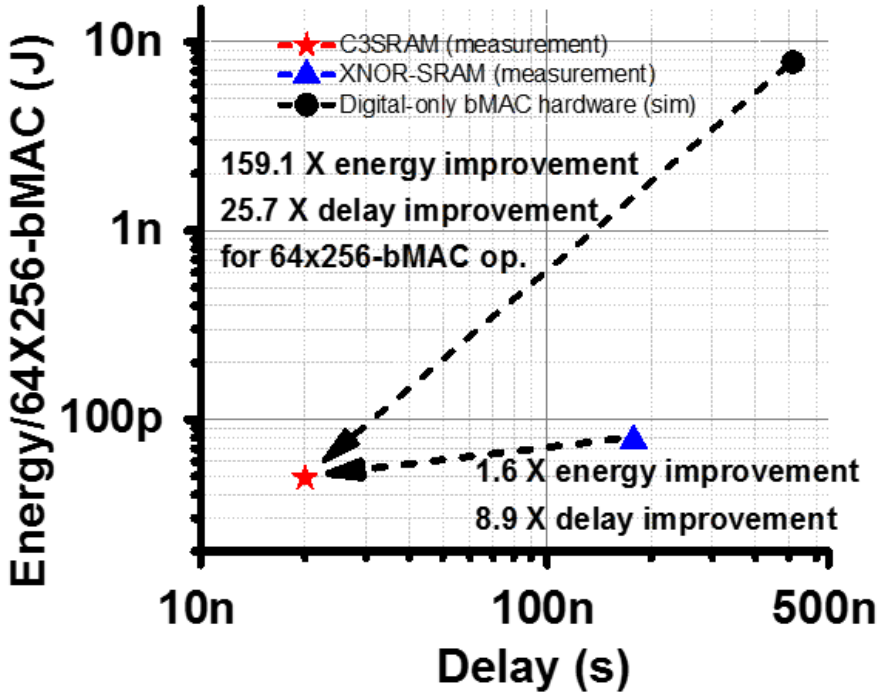


Figure 5.12: C3SRAM energy and delay comparison with XNOR-SRAM and digital ASIC with traditional SRAM and ALU.

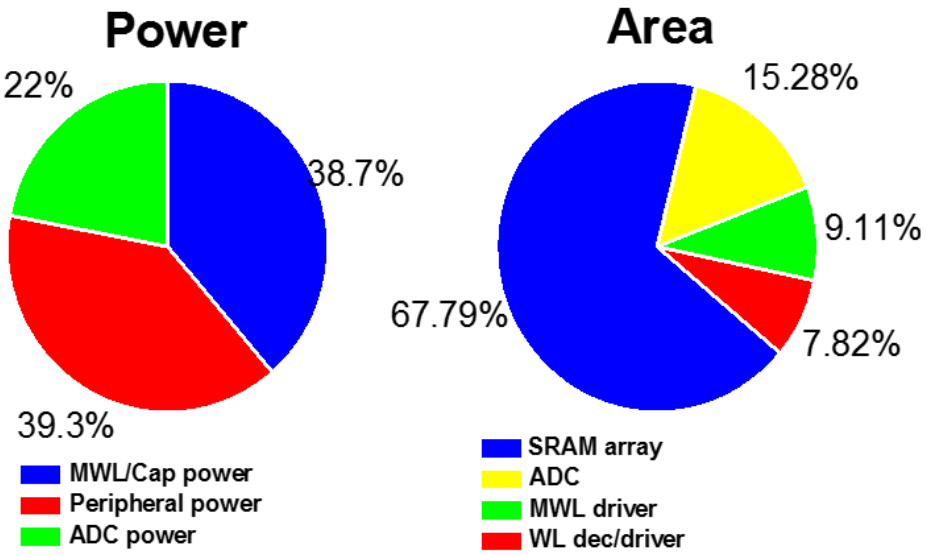


Figure 5.13: Left: measured power consumption breakdown between the three supplies powering bMAC compute (blue), partial sum accumulation (red), ADC (green). Right: area breakdown of the C3SRAM module.

Table 5.1: Comparison to prior IMC works.

	[58]	[61]	[62]	[66]	[67]	[79]	This work
Technology	65nm	65nm	65nm	65nm	55nm	65nm	65nm
Cell Type	10T	Split-6T	10T1C	12T	Twin-8T	6T	8T1C
Operating Voltage	1.2V (DAC) 0.8V (Array) 1V (rest)	1V	1V	0.6-1V	1V	1V	1V (Array) 0.8V (Driver) 0.6V (ADC)
Memory Capacity	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 256x64 as unit capacity.

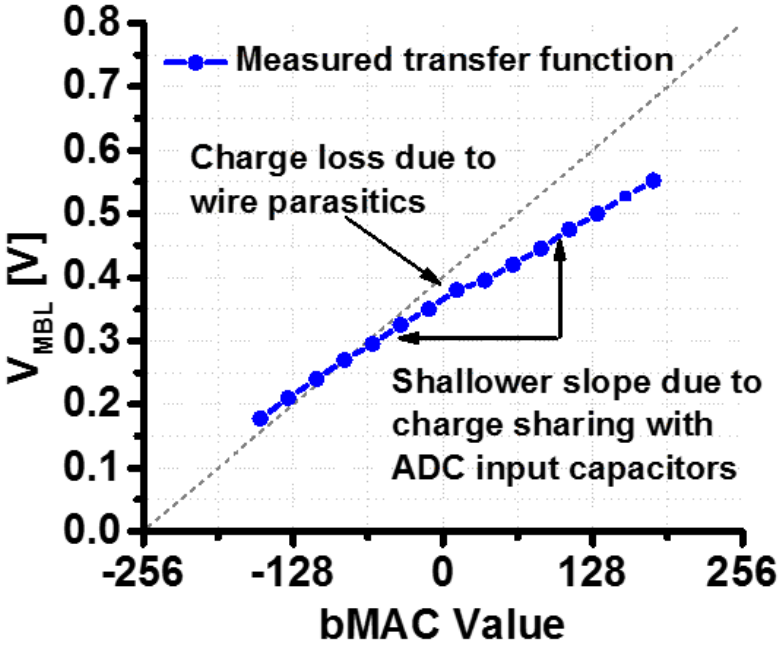


Figure 5.14: Measured V_{MBL} transfer curve shows lower FSR from ideal curve due to charge sharing with ADC input capacitors.

ing 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 V_{DR} 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 Figure 5.14, the transfer function measurement shows good linearity and stability, note the FSR is reduced due to ADC input capacitors and MBL wire parasitics.

5.4.3 Variability Measurement

In this subsection, we characterize C3SRAM variabilities. The box chart in Figure 5.15(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 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 5.2.3, 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. Figure 5.16 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 Figure 5.16.

Figure 5.15(b) shows the measured variation of bMAC operations. The main sources of variation are C_C mismatch. The mismatch variation of a single C_C has σ_C/C_C of 4.2% according to MC simulation. We determine the deviation on the transfer function using propagation of uncertainty rule:

$$\sigma_{MBL} = \frac{n}{256}\sqrt{\frac{n\sigma_C^2}{n^2C_C^2} + \frac{256\sigma_C^2}{256^2C_C^2}} = \frac{n\sigma_C}{256C_C}\sqrt{\frac{1}{n} + \frac{1}{256}} \tag{5.4}$$

The variation of V_{MBL} 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 Figure 5.14, consistent with Figure 5.15(b) (which also includes intra-chip ADC offset variation).

Figure 5.15(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.

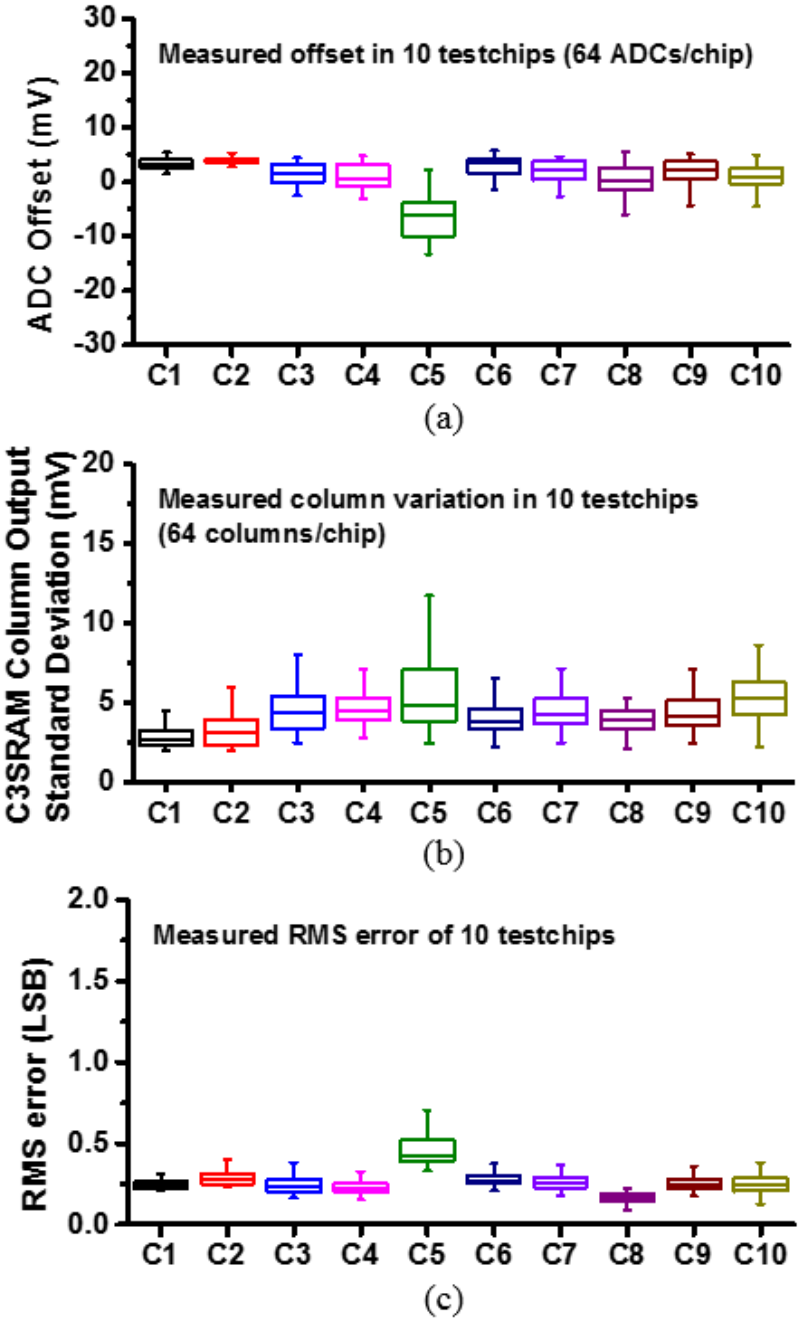


Figure 5.15: (a) ADC output offset due to comparator gain stage mismatch, (b) V_{MBL} error variation measurement, (c) RMS error of the macro.

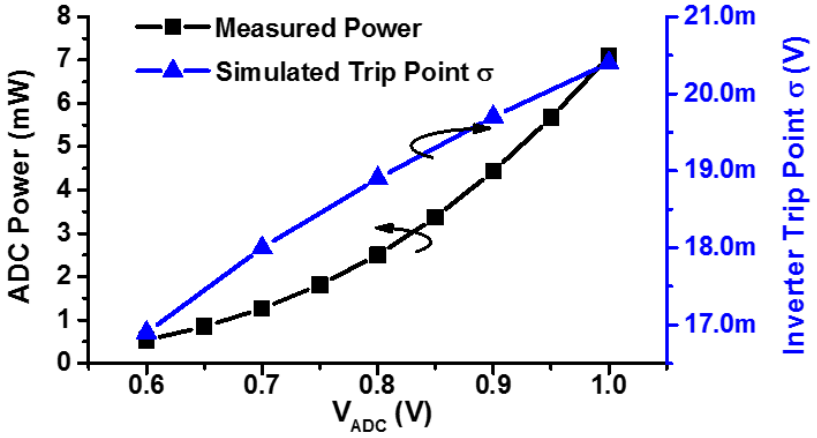


Figure 5.16: ADC power increases exponentially as ADC power supply increases; the trip point variation increases linearly.

5.4.4 Evaluation on Neural Network 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. The mapping scheme remain the same as in section 4.4.4.

As detailed in Table II, 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 results were 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 [48, 78] have found BNN conversion with wider topology improve both robustness and accuracy. As [48] demonstrates the algorithmic advances of BNNs,

Table 5.2: Measured MLP (for MNIST) and CNN (for CIFAR-10) accuracy summary using C3SRAM.

	MNIST	CIFAR-10
Neural Network	MLP	VGG-like CNN
Network Topology	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%

nCk – kxk kernel convolutional layer with n filters.
mFC – m-neuron FC layer.
MPp – max-pooling layer with pxp pooling size.

the scalable mapping of C3SRAM could be used as computational primitive for larger BNNs for more complex machine learning tasks.

5.5 Summary

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 networks in a modular fashion. Using robust capacitive coupling mechanism, the architecture can reach comparable accuracy with algorithmic baseline. The 16 kb prototype in 65 nm achieves the energy-efficiency of 671.5 TOPS/W and throughput of 1,638 GOPS for bMAC operations.

Conclusion

This work has demonstrated several algorithm/hardware co-designs for the low power local and edge computing environments. The designs presented here have disparate optimization goals and task partition considerations, thus leading to vastly different optimization focuses. It presents low power techniques across the design stacks from algorithm to micro-architecture to circuit. The designs are simulated and realized in deep sub-micron silicon technology, demonstrating state-of-the-art energy-efficiencies in their respective area.

Key enablements for local and edge computing are power and energy efficiency, which in turn effect many other aspects of these devices, like form factor, battery cycle, and device life time. To achieve high energy efficiency, many IC design techniques have been proposed over the decades, ranging from circuit level techniques like voltage scaling, power gating, clock gating, sub-threshold computing, to micro-architectural techniques like event-driven design, time-multiplexing, etc. Through early algorithm/hardware iteration, these techniques can be used to improve the co-design specification and constraint allocation and partition. This thesis demonstrates a range of low power techniques on the hardware stacks and how these techniques can be combined with algorithmic specification and implementation to improve the energy and power efficiency of the hardware systems. The designs are designed, fabricated, and tested to show significant improvements over the prior state-of-art in either hardware energy and power or algorithmic accuracy.

In this thesis, Chapter 1 presents a spike sorting hardware supporting an augmented leader algorithm. The added feature screening stage drastically reduces centroid convergence time, both

improving the spike sorting task accuracy and reducing the energy consumption of the training stage. The project discussed in Chapter 2 improves the design presented in Chapter 1 through another major modification to algorithm to allow more aggressive sleep-to-wake time to further improve the energy-efficiency. By adapting boundary based sorting criteria instead of distance based criteria, the dimensionality of the feature, and hence the computation run-time, is greatly reduced. This also allows other techniques such as utilizing content addressable memory to unroll iterations at low cost to achieve even more energy savings. Chapter 3 includes a modified off-line trained variant of the Chapter 2 design as well as partial decoding computation. The major modification of the classic Kalman Filter for neural decoding produces low data rate early in the computing pipeline to optimize task partition for implant-side power. The biological time scale of the algorithm is also taken advantage of, allowing event driven computing to compress multiplication to mere additions. Chapter 4 presents an IMC macro using a resistive voltage divider mechanism. The IMC computing paradigm is a tailor made approach for machine learning inference at the edge. The high throughput and mixed signal computing are the primary drivers of the energy and power efficiency. Chapter 5 presents an IMC macro with improved computing mechanism of capacitive coupling. By using capacitive elements as computing units, the design variability and power consumption are drastically improved.

This thesis presented algorithmic, micro-architectural, and circuit level techniques for improving the power and energy efficiency of digital and mixed signal integrated circuits for local and edge computing. The presented BCI works showcase co-design decisions bringing together existing low power hardware techniques and algorithmic modifications; the IMC works demonstrate designs in the entirely new area specifically developed for machine learning application. As algorithm/hardware co-design methodology continue to see growing practice, more computing paradigms, co-synthesis methodologies, and partition guideline will be arise in its steps, paving way for future advances in low power computing systems.

References

[1] A. Orsborn, H. Moorman, S. Overduin, M. Shanechi, D. Dimitrov, and J. Carmena, "Closed-loop decoder adaptation shapes neural plasticity for skillful neuroprosthetic control," *Neuron*, vol. 82, pp. 1380–1393, 6 2014.

[2] S. H. Lee, K. Choi, S. Jeong, J. S. Kin, and C. K. Chung, "Classifying ECoG signals prior to voluntary movement onset," in *Brain-Computer Interface (BCI)*, 2013 International Winter Workshop on, Gangwo, 2013.

[3] T. M. Vaughan, D. J. McFarland, G. Schalk, W. A. Samacki, D. J. Krusienski, E. W. Sellers, and J. R. Wolpaw, "The Wadsworth BCI research and development program: At home with BCI," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 14, pp. 229–233, 2 2006.

[4] M. S. Chae, Z. Yang, M. Yuce, L. Hoang, and W. Liu, "A 128-Channel 6 mW wireless neural recording IC with spike feature extraction and UWB transmitter," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 17, pp. 312–321, 4 2009.

[5] R. R. Harrison, R. J. Kier, C. A. Chestek, V. Gilija, P. Nuyujukian, S. Ryu, B. Greer, F. Solzbacher, and K. V. Shenoy, "Wireless neural recording with single low-power integrated circuit," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 17, pp. 322–329, 4 2009.

[6] A. M. Kamboh and A. J. Mason, "Computationally efficient neural feature extraction for spike sorting in implantable high-density recording systems," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 21, pp. 1–9, 1 2013.

[7] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering," *Neural Comp.*, pp. 1661–1687, 16 2004.

[8] V. Karkare, S. Gibson, and D. Markovic, "A 75-W, 16-channel neural spike-sorting processor with unsupervised clustering," *IEEE J. Solid-State Circuits*, vol. 48, pp. 2230–2238, 9 2013.

[9] Q. Wang, R. M. Webber, and G. B. Stanley, "Thalamic synchrony and the adaptive gating of information flow to cortex," *Nature Neuroscience*, vol. 13, pp. 1534–1541, 2010.

[10] Q. Wang, D. C. Millard, H. J. Zheng, and G. B. Stanley, "Voltage-sensitive dye imaging reveals improved topographic activation of cortex in response to manipulation of thalamic microstimulation parameters," *Journal of Neural Engineering*, vol. 9, 2 2012.

[11] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proceedings of the 29th international conference on Very large data bases*, vol. 29, 2003, pp. 81–92.

[12] M. Mahahan, P. Nimbhorkar, and K. Varadarajan, "The planar k-Means problem is NP-hard," *Theoretical Computer Science*, vol. 442, pp. 13–21, 13 2012.

[13] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *Data Mining and Knowledge Discovery*, vol. 2, pp. 169–194, 2 1998.

[14] V. Karkare, S. Gibson, and D. Markovic, "A 130 W, 64-channel, spike-sorting DSP chip," *IEEE Asian Solid-State Circuits Conference*, pp. 289–292, 2011.

[15] R. Olsson and K. Wise, "A three dimensional neural recording microsystem with implantable data compression circuitry," *IEEE J. Solid-State Circuits*, vol. 40, pp. 2796–2804, 12 2005.

[16] T. Chen, K. Chen, Z. Yang, K. Cockerham, and W. Liu, "A biomedical multiprocessor SoC for closed-loop neuroprosthetic applications," *IEEE ISSCC Digest of Technical Papers*, pp. 434–435, 2009.

[17] S. Mitra, J. Putzeys, F. Battaglia, C. Lopez, M. Welkenhuysen, C. Pennartz, C. Van Hoof, and R. Yazicioglu, "24-channel dual-band wireless neural recorder with activity-dependent power consumption," *IEEE ISSCC Digest of Technical Papers*, pp. 292–293, 2013.

[18] P. Domingos and M. Pazzani, "On the optimality of the simple Bayesian classifier under zero-one loss," *Machine learning*, vol. 29, pp. 103–130, 2-3 1997.

[19] M. Serruya, A. Shaikhouni, and J. Donoghue, "Neural decoding of cursor motion using a Kalman filter," *Neural Information Processing Systems*, vol. 15, p. 133, 2003.

[20] S. Kim, J. Simeral, L. Hochberg, J. Donoghue, and M. Black, "Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia," *Journal of neural engineering*, vol. 5, p. 455, 4 2008.

[21] V. Gilja, P. Nuyujukian, C. Chestek, J. Cunningham, M. Byron, J. Fan, M. Churchland, M. Kaufman, J. Kao, S. Ryu, and K. Shenoy, "A high-performance neural prosthesis enabled by control algorithm design," *Nature neuroscience*, vol. 15, pp. 1752–1757, 12 2012.

[22] J. Pineda, B. Allison, and A. Vankov, "The effects of self-movement, observation, and imagination on rhythms and readiness potentials (RP's): toward a brain-computer interface (BCI)," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, pp. 219–222, 2 2000.

[23] J. Mellinger, G. Schalk, C. Braun, H. Preissl, W. Rosenstiel, N. Birbaumer, and A. Kübler, "An MEG-based brain–computer interface (BCI)," *Neuroimage*, vol. 36, pp. 581–593, 3 2007.

[24] Z. Jiang, J. Cerqueira, J. Kim, Q. Wang, and M. Seok, "1.74 μW/ch, 95.3% - accurate Spike sorting hardware based on Bayesian decision," *Symposium on VLSI Circuits*, pp. 34–35, 2016.

[25] Z. Jiang, Q. Wang, and M. Seok, "A low power unsupervised spike sorting accelerator insensitive to clustering initialization in sub-optimal feature space," *Proceedings of the 52nd Annual Design Automation Conference*, p. 174, 2015.

[26] S. Zeinolabedin, A. Do, D. Jeon, D. Sylvester, and T. Kim, "A 128-channel spike sorting processor featuring 0.175 μw and 0.0033 mm^2 per channel in 65-nm CMOS," *Symposium on VLSI Circuits*, pp. 32–33, 2016.

[27] M. Hauschild, G. Mulliken, I. Fineman, G. Loeb, and R. Andersen, "Cognitive signals for brain–machine interfaces in posterior parietal cortex include continuous 3D trajectory commands," *Proceedings of the National Academy of Sciences*, vol. 109, pp. 17 075–17 080, 42 2012.

[28] S. Waldert, T. Pistohl, C. Braun, T. Ball, A. Aertsen, and C. Mehring, "A review on directional information in neural signals for brain-machine interfaces," *Journal of Physiology-Paris*, vol. 103, pp. 244–254, 3-5 2009.

[29] Y. Zhang and S. M. Chase, "A stabilized dual Kalman filter for adaptive tracking of brain-computer interface decoding parameters," *35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 7100–7103, 2013.

[30] I. Stevenson, A. Cherian, B. London, N. Sachs, E. Lindberg, J. Reimer, M. Slutzky, N. Hatsopoulos, L. Miller, and K. Kording, "Statistical assessment of the stability of neural movement representations," *Journal of neurophysiology*, vol. 106, pp. 764–774, 2 2011.

[31] B. Walker and K. Kording, "The database for reaching experiments and models," *PloS one*, vol. 8, e78747, 11 2013.

[32] J. Li, P. Chundi, S. Kim, Z. Jiang, M. Yang, J. Kang, S. Jung, S. Kim, and M. Seok, "A 0.78-μw 96-ch. deep sub-v_t neural spike processor integrated with a nanowatt power management unit," *IEEE 44th European Solid State Circuits Conference*, pp. 154–157, 2018.

[33] Y. Yang, S. Boling, and A. J. Mason, "A hardware-efficient scalable spike sorting neural signal processor module for implantable high-channel-count brain machine interfaces," *IEEE transactions on biomedical circuits and systems*, vol. 11, pp. 743–754, 4 2017.

[34] M. Zamani, D. Jiang, and A. Demosthenous, "An adaptive neural spike processor with embedded active learning for improved unsupervised sorting accuracy," *IEEE transactions on biomedical circuits and systems*, vol. 12, pp. 665–676, 3 2018.

[35] S. Gibson, J. W. Judy, and D. Markovic, "Technology-aware algorithm design for neural spike detection, feature extraction, and dimensionality reduction," *IEEE transactions on neural systems and rehabilitation engineering*, vol. 18, pp. 469–478, 5 2010.

[36] K. Mizuseki, A. Sirota, E. Pastalkova, and G. Buzsáki, "Theta oscillations provide temporal windows for local circuit computation in the entorhinal-hippocampal loop," *Neuron*, vol. 64, pp. 267–280, 2 2009.

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

[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015.

[39] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. 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.

[40] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *The 32nd International Conference on Machine Learning (ICML)*, 2015, pp. 448–456.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[42] 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. arXiv: 1609.07061.

[43] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *The 32nd International Conference on Machine Learning (ICML)*, 2015, pp. 1737–1746.

[44] M. Courbariaux, Y. Bengio, and J. David, "BinaryConnect: Training Deep Neural Networks with Binary Weights During Propagations," in *Advances in Neural Information and Processing Systems (NIPS)*, 2015, pp. 3123–3131.

[45] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized Neural Networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 4107–4115.

[46] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *European Conference on Computer Vision (ECCV)*, 2016.

[47] 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. arXiv: 1606.06160.

[48] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," *Advances in Neural Information Processing Systems (NIPS)*, pp. 345–353, 2017.

[49] T. Guan, X. Zeng, and M. Seok, "Recursive binary neural network learning model with 2-bit/weight storage requirement," *CoRR*, vol. abs/1709.05306, 2017. arXiv: 1709.05306.

[50] 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, 668–673, 2014.

[51] Y.-H. Chen, T. Krishna, J. Emer, and V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 52, no. 1, pp. 127–138, 2017.

[52] 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 *IEEE International Solid-State Circuits Conference (ISSCC)*, 2017, pp. 246–247.

[53] D. Shin, J. Lee, J. Lee, and H.-J. Yoo, "DNPU: An 8.1TOPS/W Reconfigurable CNN-RNN Processor for General-Purpose Deep Neural Networks," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2017, pp. 240–241.

[54] P. N. Whatmough, S. K. Lee, H. Lee, S. Rama, D. Brooks, and G.-Y. 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 *IEEE International Solid-State Circuits Conference (ISSCC)*, 2017, pp. 242–243.

[55] 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 *IEEE Symposium on VLSI Circuits*, 2017, pp. 160–161.

[56] 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 *European Solid-State Circuits Conference (ESSCIRC)*, 2017.

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

[58] 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, 2019.

[59] S. K. Gonugondla, M. Kang, and N. Shanbhag, "A 42pj/decision 3.12tops/w robust in-memory machine learning classifier with on- chip training," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2018.

[60] W.-H. Chen, K.-X. Li, W.-Y. Lin, K.-H. Hsu, P.-Y. Li, C.-H. Yang, C.-X. Xue, E.-Y. Yang, Y.-K. Chen, Y.-S. Chang, T.-H. Hsu, Y.-C. King, C.-J. Lin, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, and M.-F. Chang, "A 65nm 1Mb Nonvolatile Computing-in-Memory ReRAM Macro with Sub-16ns Multiply-and-Accumulate for Binary DNN AI Edge Processors," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2018.

[61] 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," in *IEEE Transactions on Circuits and Systems I*, vol. 66, 2019, pp. 4172–4185.

[62] 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 *IEEE Symposium on VLSI Circuits*, 2018, 141–142.

[63] M. Kang, S. Lim, S. Gonugondla, and N. R. Shanbhag, "An in-memory vlsi architecture for convolutional neural networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 3, pp. 494–505, 2018.

[64] 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)*, 2019, 224–226.

[65] C. Eckert, X. Wang, J. Wang, A. Subramaniyan, R. Iyery, D. Sylvester, D. Blaauw, and R. Das, "Neural Cache: Bit-Serial In-Cache Acceleration of Deep Neural Networks," in *IEEE International Symposium on Computer Architecture (ISCA)*, 2018.

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

[67] 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)*, 2019, 396–398.

[68] H. Valavi, P. Ramadge, E. Nestler, and N. Verma, "A 64-tile 2.4-Mb in-memory-computing CNN accelerator employing charge-domain compute," *IEEE Journal of Solid-State Circuits*, pp. 1789–1799, 2019.

[69] S. Yin, Z. Jiang, M. Kim, T. Gupta, M. Seok, and J. Seo, "Vesti: Energy-efficient in-memory computing accelerator for deep neural networks," in *IEEE Transactions on VLSI Systems*, 2019.

[70] M. Horowitz, *Computing's energy problem (and what we can do about it)*, 2014.

[71] 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, 2015.

[72] 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)*, 2015, pp. 17.1.1–17.1.4.

[73] 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 *IEEE International Symposium on Computer Architecture (ISCA)*, 2016, pp. 27–39.

[74] A. Chen and M.-R. Lin, "Variability of resistive switching memories and its impact on crossbar array performance," in *IEEE International Reliability Physics Symposium (IRPS)*, 2011.

[75] T. Gokmen and Y. Vlasov, "Acceleration of deep neural network training with resistive cross-point devices: design considerations," *Frontiers in Neuroscience*, vol. 10, 2016.

[76] 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, 43–55, 2019.

[77] 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," 2018. arXiv: 1811.04047.

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

[79] S. K. Gonugondla, M. Kang, and N. R. Shanbhag, "A variation-tolerant in-memory machine learning classifier via on-chip training," *IEEE Journal of Solid-State Circuits*, vol. 53, 3163–3173, 11 2018.