A HIERARCHICAL ADAPTIVELY BOOSTED IN-MEMORY CLASSIFIER IN 6T SRAM

BY

SUNGMIN LIM

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Adviser:

Professor Naresh R. Shanbhag

ABSTRACT

Recent emerging machine learning applications such as Internet-of-Things and medical devices require to be operated in a battery-powered platform. As the machine learning algorithms involve heavy data-intensive computations, interest in energy-efficient and low-delay machine learning accelerators is growing. Because there is a trade-off between energy and accuracy in machine learning applications, it is a reasonable direction to provide scalable architecture which has diverse operating points.

This thesis presents a high-accuracy in-memory realization of the AdaBoost machine learning classifier. The proposed classifier employs a deep in-memory architecture (DIMA), and employs foreground calibration to compensate for PVT variations and improve task-level accuracy. The proposed architecture switches between a high accuracy/high power (HA) mode and a low power/low accuracy (LP) mode via soft decision thresholding to provide an elegant energy-accuracy trade-off. The proposed realization achieves an EDP reduction of 43X over a digital architecture at an iso-accuracy of 95% for the MNIST dataset, which is an improvement of 5% over a previous in-memory implementation of AdaBoost.

To my parents, for their love and support.

ACKNOWLEDGMENTS

I sincerely feel gratitude toward my adviser, Prof. Naresh R. Shanbhag, for his support and guidance. I would also like to extend my gratitude to Sujan Gonugondla, Ameya Patil, and Charbel Sakr for helpful discussions. I would like to give special thanks to Mingu Kang and Yongjune Kim for strong support in various aspects. I was lucky to join a research group in which I could experience active discussions in group meetings and have healthy interactions with group members for my individual growth. This work was supported by Systems On Nanoscale Information Fabrics (SONIC), one of the six SRC STARnet Centers, sponsored by SRC and DARPA.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AdaBoost	Adaptive Boosting
BCA	Bit-cell array
BL	Bit-line
BLB	Bit-line bar
BLP	Bit-line processing
CBLP	Cross bit-line processing
DIMA	Deep in-memory architecture
ML	Machine learning
SNR	Signal-to-noise ratio
SRAM	Static random-access memory
WL	Word-line

CHAPTER 1

INTRODUCTION

1.1 Motivation

The high energy and delay costs of current day machine learning (ML) algorithms inhibit their deployment for real-time always-on inference on sensor-rich platforms such as wearables, UAVs, personal biomedical devices, Internet of Things (IoT), and many others. In such systems, the data movement dominates the high energy and latency cost [1]. As a result, energy-efficient and low-latency machine-learning hardware is required to sustain the always-on functionality (e.g. face detection) amid resource constraints such as limited form factor for mobility, processing time for real-time streamed-in input data, and energy for battery-powered platform (Fig. 1.1). However, the current ML platforms (e.g. CPU, GPU, and FPGA) based on von Neumann architecture [2] are not suitable in the battery-powered hardware due to their high data movement cost [3].

A number of machine learning accelerators [4–10] to reduce the data movement cost using data reuse methods have been proposed, but these target the server platform and are limited by the memory-processor interface. Therefore, overcoming von Neumann structure should be the first step to build a data-intensive computing architecture for the emerging applications. Additionally, there is a fundamental trade-off between energy and accuracy in most ML applications. Therefore, it is a reasonable direction to implement scalable architecture with a wide spectrum of operating points because all the applications do not necessarily require highly accurate networks with large energy consumption.

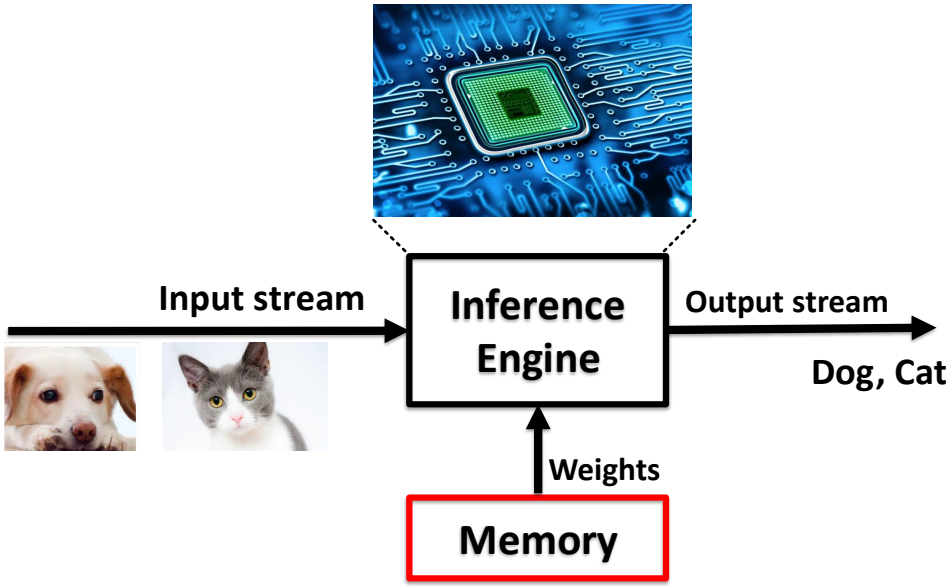


Figure 1.1: Simplified von Neumann machine-based inference.

1.2 Related Work

Recently, in-memory architectures [1, 11–14] were proposed to address the data movement cost. Such architectures embed low-swing analog computations in the periphery of the SRAM bit-cell array (BCA) to minimize memory access rates, substantially reducing the energy-delay product (EDP) of inference, but can lead to degradation in task-level accuracy due to circuit non-idealities such as V_t variations in the bitcell array (BCA). Nevertheless, IC prototypes have realized up to 100× reduction in the energy-delay product (EDP) [12] at iso-accuracy with digital architectures.

The multi-functional in-memory inference processor [1] and the random forest accelerator [11] achieve significant gains in the energy efficiency (10×) and throughput (5.3×) over a conventional digital architecture by exploiting the inherent error tolerance of machine learning algorithms. In contrast, [12] shows that training with chip-in-the-loop can improve DIMA's accuracy but at the expense of a large retraining overhead, e.g., requiring 6400 images for binary classification. On the other hand, [11] implemented the random forest algorithm on DIMA using an embedded crossbar for feature extraction to achieve high accuracy in an 8-class traffic sign recognition task. However, the crossbar can be too complex to be employed in always-on applications.

For the always-on IoT applications, the AdaBoost [15] algorithm is attractive due to its low computational complexity and good accuracy. In spite of its simplicity, not much work has been done on realizing the AdaBoost algorithm using in-memory architectures except for [13]. In [13], a 10-class in-memory AdaBoost classifier [13] achieved an energy efficiency of 630 pJ per decision. However, the achievable accuracy was limited to 90% for the MNIST dataset, which was improved to 91% via the use of four in-memory ICs [16] indicating the challenge of improving the task-level accuracy of AdaBoost using in-memory architectures. Also, 25% of SRAM bit-cells had to be allocated to compensate for comparator offsets, compromising the memory density [13, 16].

As another approach for an efficient architecture, there are a number of attempts to build the scalable hardware which have a trade-off between energy, delay and accuracy. Dynamic Voltage-Accuracy Scaling (DVAS) [17] exploits shorter critical paths combined with lower precision for scaled voltage. As a next version, Dynamic Voltage-Accuracy-Frequency Scaling (DVAFS) [7] made additional frequency scaling possible. Precision scaling [6, 7, 17] is widely used with masking input or weight bits. Lastly, hierarchical recognition by [7] provides increasing numbers of classes as it goes to last recognition stage with growing complexity. As an orthogonal way of scaling, this thesis proposes hybrid mode using soft decision value to determine an operating point.

1.3 Thesis Contributions and Organization

This thesis proposes a hierarchical architecture to realize in-memory AdaBoost with improved accuracy for always-on applications using decision tree-based weak classifiers. The proposed architecture computes a *soft decision margin* in its low-power/low-accuracy model (*LP mode*), and switches to a high-accuracy/high-power mode (*HA mode*) only when the confidence level indicated by the soft decision margin in the LP mode is low. The proposed architecture is experimentally validated via test configurations of a previously reported IC [11] using 8-b precision for both weights and input pixels. The proposed in-memory AdaBoost realization achieves a 43× reduction in EDP at an iso-accuracy of 95% over a digital neural network [18].

This is a 5% improvement in accuracy over the previous in-memory AdaBoost implementation [13] though at a higher EDP.

This remainder of thesis is organized as follows. Chapter 2 explains background for DIMA and AdaBoost. Implementing the AdaBoost algorithm on DIMA, the proposed architecture, and foreground calibration techniques are described in Chapter 3. The measurement results including energy, throughput, and accuracy from both component-level and task-level are described in Chapter 4. Finally, Chapter 5 concludes the thesis.

CHAPTER 2

BACKGROUND

2.1 Deep In-memory Architecture (DIMA)

2.1.1 DIMA Overview

This thesis employs the DIMA platform to implement an energy-efficient and low-latency AdaBoost algorithm accelerator. DIMA reads multiple rows of a standard 6T SRAM bitcell array (BCA) per precharge via pulse width modulated (PWM) wordline (WL) pulses to read word-level information. This stage is called multi-row functional read [1]. It processes the consequent bitline (BL) voltage drops ΔV_{BL} via column pith-matched bitline processor (BLP) in the periphery of the BCA (Fig. 2.1).

The BLP computes scalar distances such as multiplication and scalar comparison. Following BLP output is aggregated in the cross bitline processing stage for dimension reduction operation (e.g. sum). This work uses BLP as scalar comparison for thresholding.

While the conventional SRAM architecture requires a $L:1$ column mux ratio (typically $L = 4$ to 32) due to large area of sense amplifiers (SA) as shown in Fig. 2.2, DIMA does not require SAs to read memory because it directly uses ΔV_{BL} without massive data transfer [1, 11]. Column mux limits the number of bits per access to N_{COL}/L in standard SRAM compared to $N_{COL} \times 4$ in DIMA with multi-row read as shown in Fig. 2.1. As a result, DIMA reduces dominant memory access cost in computations [3] compared to the conventional digital architecture. Comparing data-flow conventional digital architecture (Fig. 2.2) and DIMA (Fig. 2.1), it is clear that DIMA has fewer intermediate steps than the digital architecture. This implies that DIMA has strong benefits in terms of energy and delay.

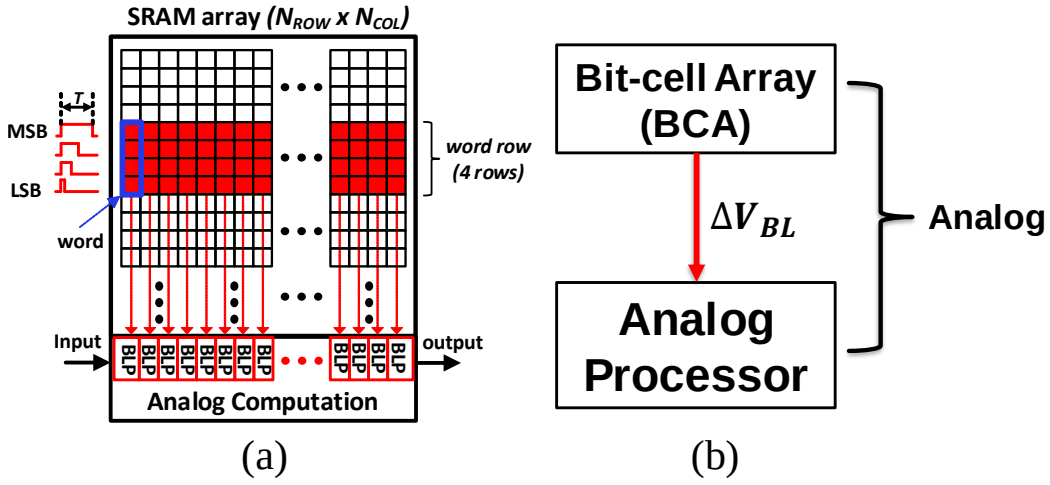


Figure 2.1: Proposed in-memory computing hardware: (a) DIMA [1, 11], and (b) data-flow.

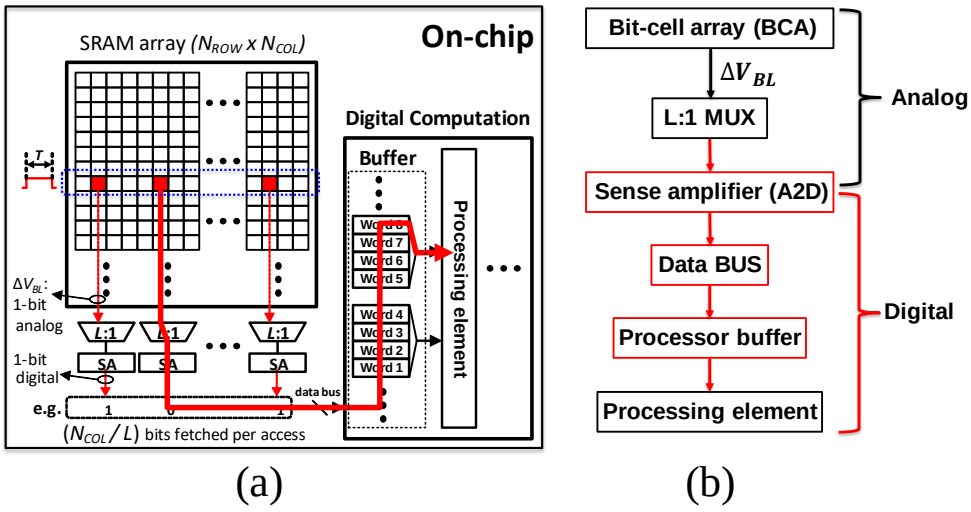


Figure 2.2: Conventional digital hardware: (a) Conventional digital architecture, and (b) data-flow.

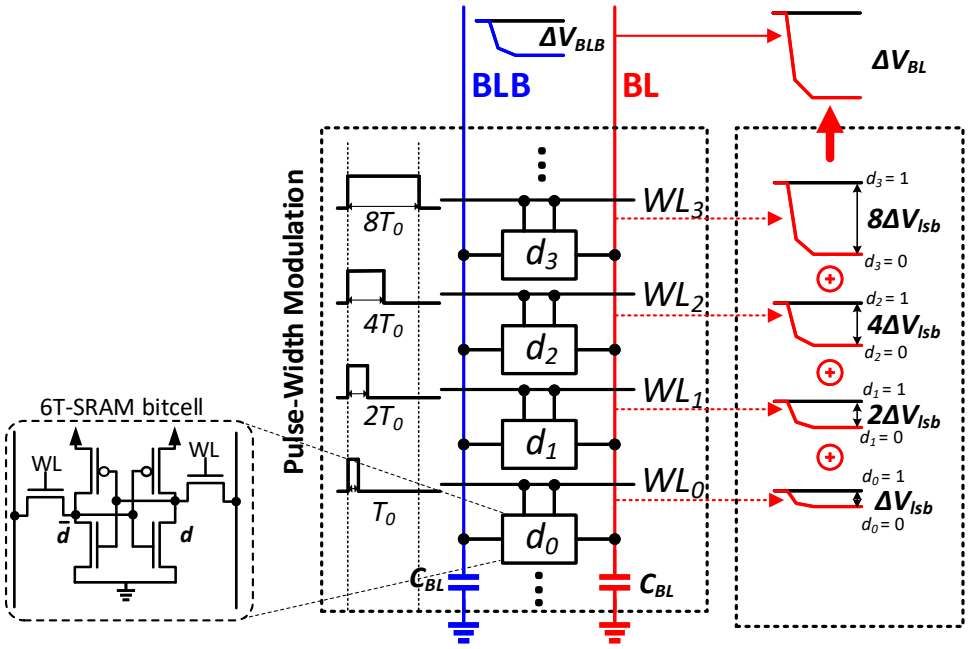


Figure 2.3: DIMA multi-row functional read ($B = 4$).

2.1.2 Multi-row Functional Read

The multi-row functional read (Fig. 2.3) creates voltage drop ΔV_{BL} which is proportional to B-bit word data $D(= \sum_{i=0}^{B-1} 2^i d_i)$ stored in column-major. The $\Delta V_{BL}(D)$ can be fomulated as:

$$\Delta V_{BL}(D) = \frac{V_{PRE}}{R_{BL}C_{BL}} T_0 \sum_{i=0}^{B-1} 2^{i-1}\overline{d_i} = \Delta V_{lsb}\overline{D},$$

where V_{PRE} is BL precharge voltage, R_{BL} is resistance of the BL discharge path via the access and pull-down NMOS transistors, C_{BL} is BL parasitic capacitance, and T_0 is minimum pulse width enabled on WL. Here, $\overline{D}$ is the integer number of one's complement of D. The unit BL voltage drop $\Delta V_{lsb} = \frac{V_{PRE}}{R_{BL}C_{BL}}$ is a function of WL voltage V_{WL} as R_{BL} depends on V_{WL}. As ΔV_{BL} is closely related to energy consumption, V_{WL} can be used to implement scalable architecture.

2.2 Adaptive Boosting (AdaBoost)

The ensemble classifier is one of the machine learning classification algorithms. The ensemble methods create a strong classifier by combining weak classifiers which are inaccurate with around 50% detection failure rate. An analog circuit can exploit this property because the analog computations are usually faster and lower energy, but less accurate, than digital implementations. Therefore, designing a weak classifier in analog domain can take advantage of ensemble classification.

AdaBoost, as an ensemble method, adds the weak learners iteratively to build a highly accurate network by training the weak learners on different distributions over the example dataset. Let us say training examples $\langle (I_1, y_1),\ (I_2, y_2), ...,\ (I_m, y_m) \rangle$ are given, where $I_i \in I$ is an image sample in the training set I and $y_i \in \{-1, +1\}$ is a label. In the first step, a distribution D_1 on the training set is initialized as:

$$D_1(i) = 1/m, \quad \forall\, i = 1, ..., m.$$

The distribution D is used to give different weighting distributions on examples at each training sequence. In the t-th iteration, a current weak learner focuses on misclassified input images in the previous iteration in order to reduce the same errors at a subsequent learner by weighting the failed images, namely adaptively boosting. Given an input set, the t-th weak classifier q_t is trained to minimize the weighted error ϵ_t, which is a sum of the product of the i-th weak hypothesis $q_t(I_i) \in \{-1,\ +1\}$ error and the i-th distribution $D_t(i)$ in the t-th iteration:

$$\epsilon_t = \sum_{i=1}^{m} D_t(i) \cdot \mathbf{1}[q_t(I_i) \neq y_i].$$

Once the ϵ_t is determined, AdaBoost chooses a parameter α_t as follows:

$$\alpha_t = \frac{1}{2} \ln(\frac{1 - \epsilon_t}{\epsilon_t}).$$

The higher value of α_t means that the t-th weak classifier is more reliable, having high impact on a final (strong) classification. In the next iteration, AdaBoost updates the distribution D_t to D_{t+1} for $i = 1, ..., m$ as follows:

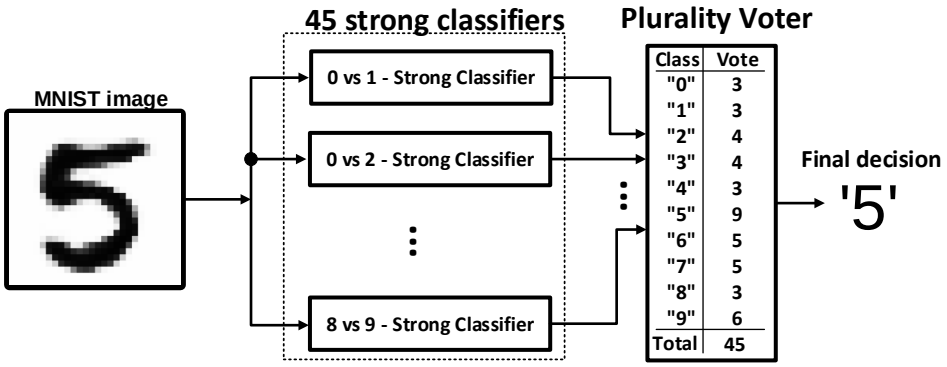


Figure 2.4: 10-class classifier with 45 strong classifiers.

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i q_t(I_i))}{Z_t},$$

where Z_t is a normalization factor:

$$Z_t = \sum_{i}^{m} D_t(i)\exp(-\alpha_t y_i q_t(I_i)).$$

Finally, during inference, a test image I_j is classified as:

$$\hat{y}(I_j) = sign(\sum_{t=1}^{M} \alpha_t q_t(I_j)),$$

where M is a total number of weak classifiers.

This thesis employs a simple comparison between a pixel value and a trained threshold as a single weak classifier in AdaBoost inference, resulting in binary output q_t. In order to build a 10-class classifier, the $C_2^{10} = 45$ strong binary classification results are computed using the one-vs.-one strategy. Note that the one-vs.-one strategy only distinguishes two classes (binary decision). As shown in Fig. 2.4, the strong binary decisions from the 45 one-vs.-one classifiers are fed into a plurality voter and the final decision (e.g., 5) is made.

CHAPTER 3

DIMA-BASED ADABOOST ARCHITECTURE

3.1 Implementation Challenges

Implementing a multi-class inference system in-memory using AdaBoost with decision trees presents the following challenges: (1) *Crossbar cost*: Each strong classifier (Fig. 3.1) needs to use a different subset of features necessitating a crossbar whose complexity increases with feature dimension. (2) *Circuit non-idealities*: DIMA is vulnerable to various circuit non-idealities such as PVT variations and non-linearity due to low-voltage analog operations. (3) *Retraining complexity*: Retraining to overcome circuit non-idealities incurs significant complexity overhead, e.g., 25% of memory capacity [13] devoted to offset calibration, and massive retraining dataset [12]. By implementing AdaBoost on the in-memory architecture IC [11], this work demonstrates the benefits of exploiting the intrinsic error-tolerance of an ensemble classifier and the energy-efficiency of a mixed-signal implementation.

3.2 Proposed In-memory Architecture

The proposed architecture in Fig. 3.1 realizes a 10-class classifier output $\hat{y}$ by *plurality voting* the outputs $\hat{y}_n$ ($n = 1, \ldots, N = 45$) of $N = C_2^{10} = 45$ boosted (strong) binary classifiers. The n-th strong classifier's output obtained is

$$\hat{y}_n = sgn(|\tilde{y}_n - \widehat{T}_n|), \tag{3.1}$$

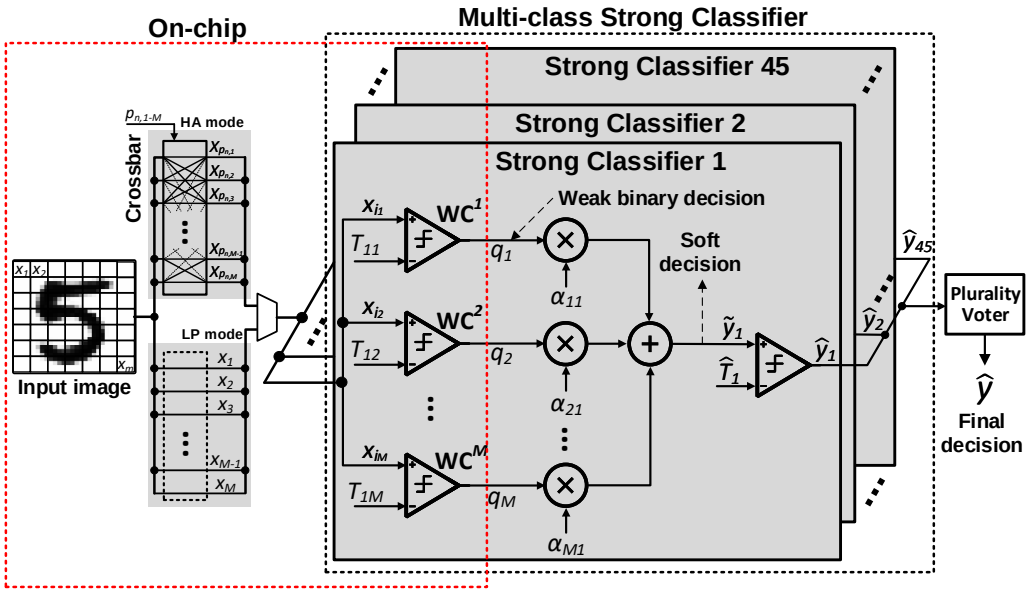


Figure 3.1: Multi-class AdaBoost classifier: top-level architecture with $M = 256$ weak classifiers per strong classifier and $N = 45$ strong classifiers.

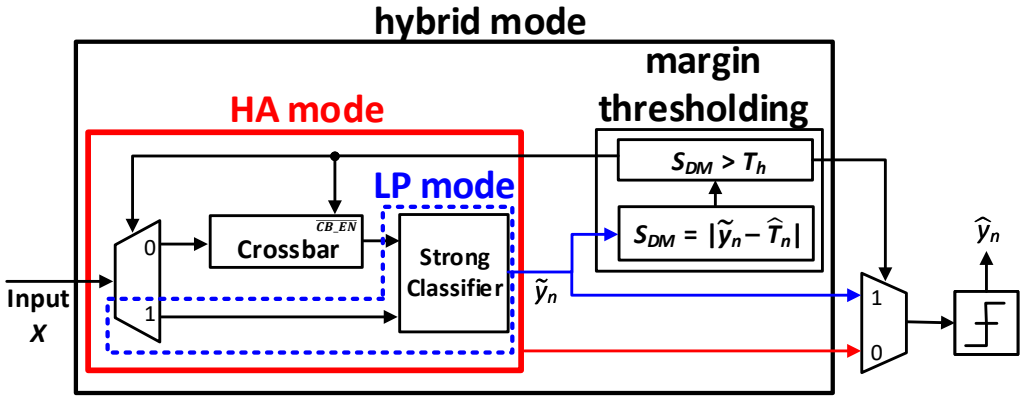


Figure 3.2: Proposed AdaBoost modes: the low-power (LP), high accuracy (HA), and hybrid modes in a strong classifier.

where $\widehat{T}_n$ is the *strong classifier threshold*, and the *soft decision* $\tilde{y}_n$ is computed as

$$\tilde{y}_n = \sum_{m=1}^{256} \alpha_{nm} q_{nm}. \tag{3.2}$$

Here, α_{nm}'s are the trained strong classifier parameters, and q_{nm} ($n = 1, \ldots, 45; m = 1, ..., 256$) are the weak classifier decisions obtained as:

$$q_{nm} = \begin{cases} 1 & \text{if } T_{nm} > X_i \\ 0 & \text{otherwise,} \end{cases} \tag{3.3}$$

where T_{nm} is the trained *pixel threshold* of the m-th weak-classifier within the n-th strong classifier, and X_i is a pixel with index $i \in [1, 256]$. Though each weak classifier has low accuracy, e.g., slightly greater than 50%, each strong classifier generates a $> 90\%$ accuracy. Plurality voting of 45 such strong binary classifiers generates the final 10-class prediction. Each weak classifier is a scalar comparator that compares pixel X_i with a pixel threshold T_{nm} via in-memory computations in [11], thereby addressing challenges (2) and (3) in section 3.1. Using scalar comparators as weak classifiers enables simple foreground calibration, where the threshold and all possible pixel values are compared in-memory to retrain the thresholds. In this manner, the comparator offset and bit-cell variation are compensated for without requiring complex gradient descent based approaches [12, 13] with massive training datasets.

Challenge (1) is addressed by introducing three strong classifier modes (Fig. 3.2): (a) a *high-accuracy (HA) mode* by realizing a many-to-one mapping of pixels to a threshold via a crossbar, (b) a *low-power (LP) mode* by realizing a one-to-one mapping of pixels to a threshold bypassing the crossbar to achieve energy and delay efficiency at the cost of accuracy, and (c) *a hybrid mode* to obtain the energy-efficiency of the LP mode and accuracy of the HA mode by selectively enabling crossbar using *soft decision margin* ($S_{DM} = |\tilde{y}_n - \widehat{T}_n|$) of the LP mode (Fig. 3.2). Here, the LP mode works as an *always-on* early detector to filter binary decisions of the strong classifiers which have $S_{DM} < T_h$ (low-confidence decisions), where T_h is the margin threshold. The costly HA mode is enabled to improve accuracy only if the

soft decision margin in the LP mode is low. Challenge (1) in the HA mode is further addressed by deterministic sub-sampling (DSS) [11], where four groups of weak classifiers are constrained to use a dedicated one-of-four 4:1 sub-sampled input images. Therefore, a single 256:1 crossbar can be replaced by four 64:1 crossbars achieving significant complexity reduction with less than 0.2% accuracy degradation.

3.3 Circuit Design

The proposed 10-class classifier (Fig. 3.3) includes a SRAM BCA to store pretrained 8-b thresholds T_{nm}^{HA}, T_{nm}^{LP} and 6-b pixel index $p_{n,m}$, multi-row wordline (WL) drivers, 64-b I/O with a 4:1 column mux, DSS input buffer to store streamed-in 256 8-b pixels X_i, four 64:1 crossbars, and peripherals for standard read/write operations. The LP and HA modes use T_{nm}^{HA} and T_{nm}^{LP}, respectively, to classify an image. The crossbar is enabled only in the HA mode and routes pixels to replica bit-cell array via the pixel index $p_{n,m}$ which is stored in the BCA at the start of in-memory comparison [11].

In-memory comparison (Fig. 3.4) [11] begins by storing the 128 pixels of X into the replica BCA, which is designed to write the 8-b pixels X_i efficiently by having additional write BL with access transistors. Storing the X in the replica BCA (Fig. 3.5) allows fast writing through a separate write BL (WBL) and wordline (WWL) by eliminating the overheads of slow write operation into normal BCA. The multi-row WL driver applies binary pulse-width modulated pulses simultaneously to WL_{3-0} and RWL_{3-0} to discharge BL (BLB) creating voltage swing $\Delta V_{BL}(\Delta V_{BLB})$ proportional to $X - T(T - X)$. Here, linearity of the multi-row read is improved by reading 4-b MSBs and LSBs separately from adjacent columns followed by a capacitively weighted charge sharing that assigns 16× greater weight to the MSBs. The WL voltage is reduced (e.g. 0.65 V) to prevent destructive read and further improve the linearity.

Finally, in-memory comparison phase generates 128 binary weak decisions $q_{n1\sim128}$, requiring two such cycles to compute one strong classifier decision $\hat{y}_n$. After 90 such in-memory cycles, the final multi-class decision $\hat{y}$ is generated via plurality voting the 45 binary strong decisions $\hat{y}_n$. The computation of $\tilde{y}_n$, $\hat{y}_n$, and plurality voting to obtain $\hat{y}$ is done off-chip.

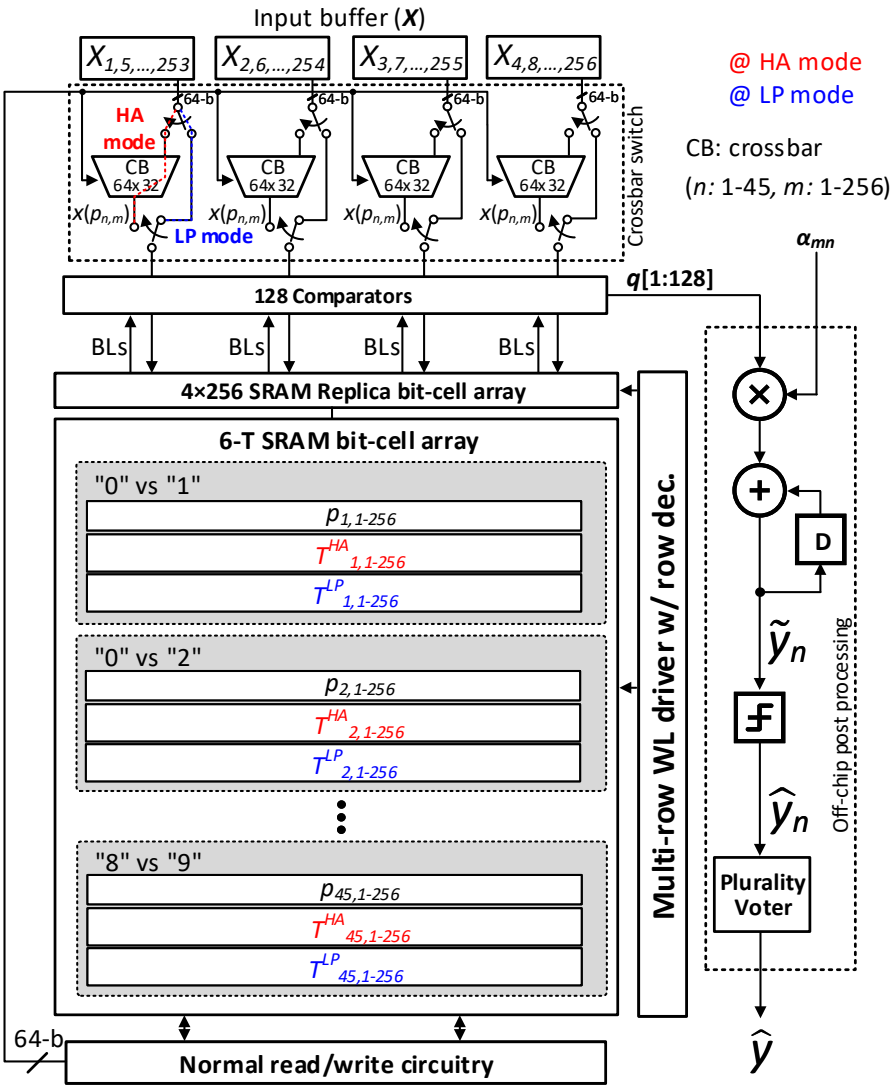


Figure 3.3: Proposed in-memory AdaBoost hardware architecture with HA and LP modes.

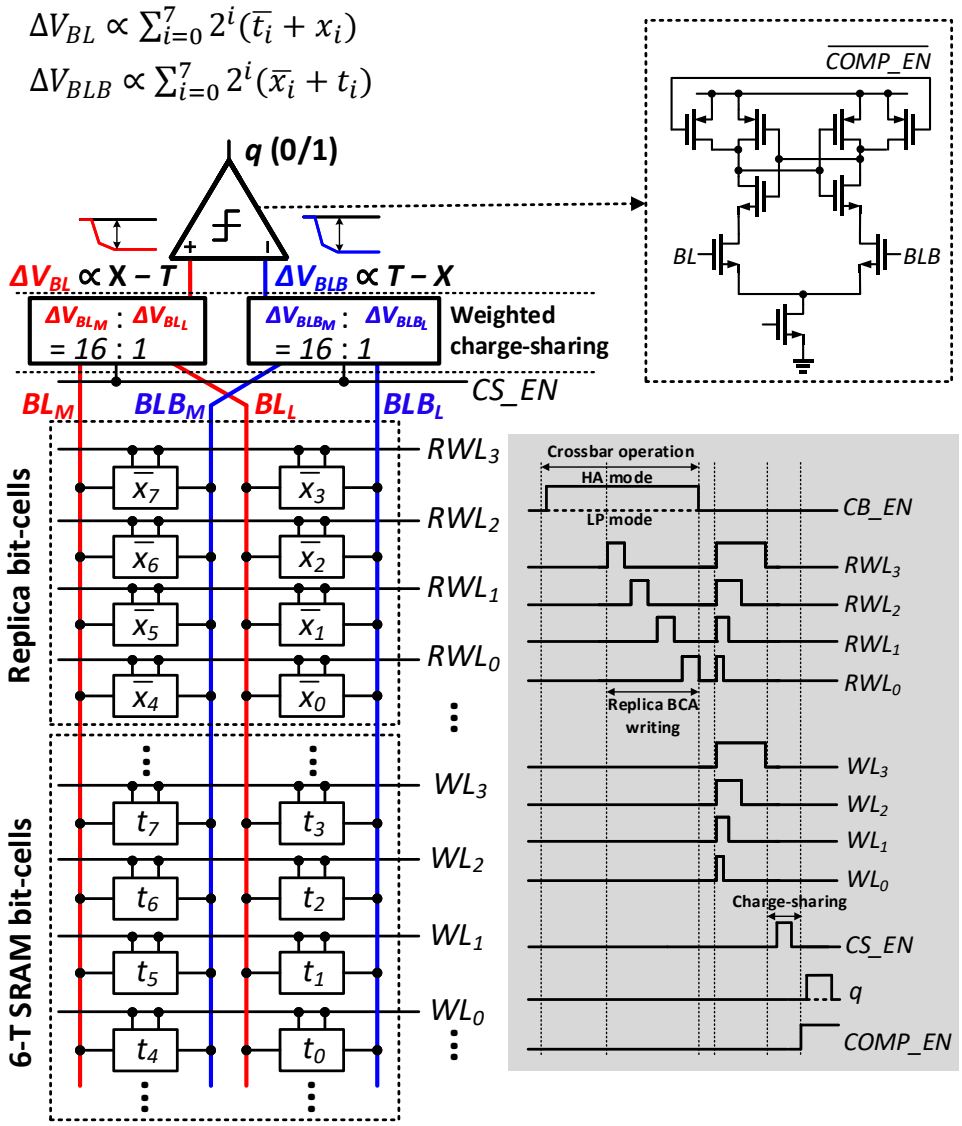


Figure 3.4: In-memory comparison with analog comparator and timing diagram.

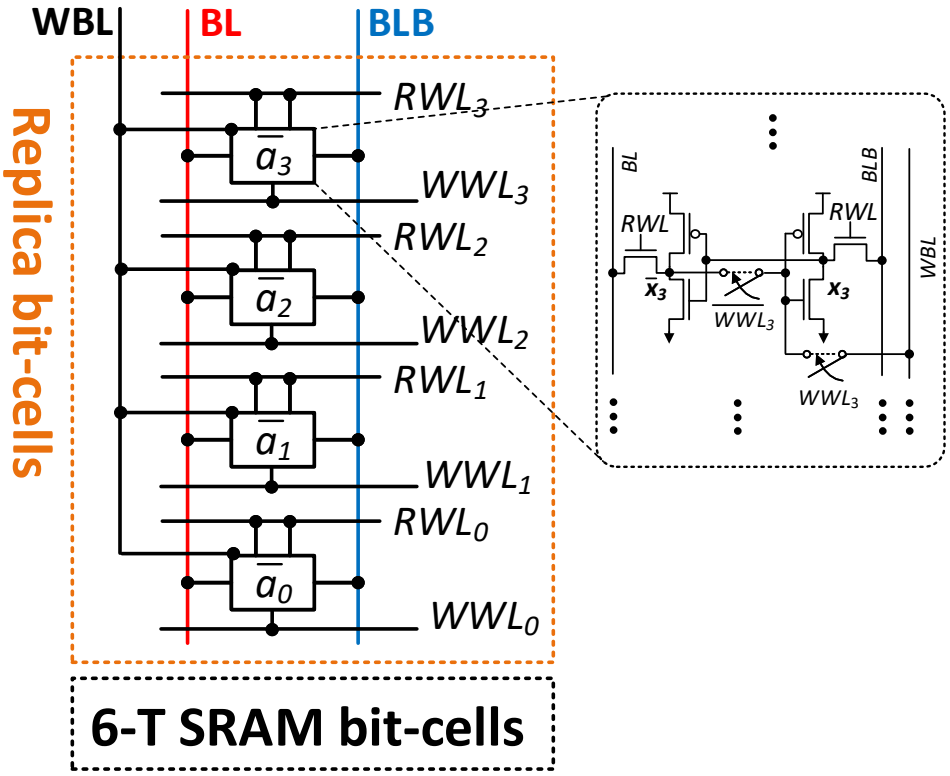


Figure 3.5: Replica bit-cell circuit configuration.

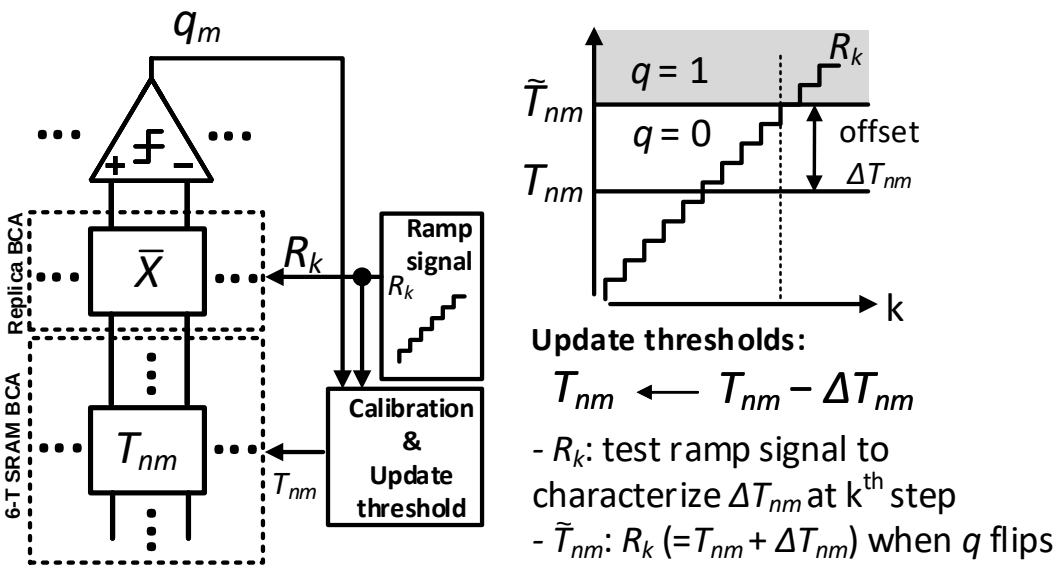


Figure 3.6: The foreground calibration process to compensate for bit-cell variations.

3.4 Retraining with Foreground Calibration

Figure 3.6 shows the foreground calibration process to reduce weak classifier errors due to process variations. The foreground calibration uses in-memory comparisons to estimate the offset ΔT_{nm} between ideal T_{nm} and realized $\tilde{T}_{nm}$ caused by PVT variations in bit-cells and comparators. By comparing ramp signal R_k and off-chip trained thresholds (T_{nm}), the update $T_{nm} \leftarrow T_{nm} + \Delta T_{nm}$ is performed and stored in the BCA to compensate for the variations. By doing so, foreground calibration achieves 1.3× better memory density than [13] without requiring dedicated offset cancellation bit-cells and 25× lower retraining dataset complexity than [12].

CHAPTER 4

MEASUREMENT RESULTS

This chapter provides the measured results from the prototype IC [11] including energy, delay, and accuracy. By comparing component- and task-level accuracy, DIMA-based AdaBoost architecture demonstrates its robustness to circuit non-idealities and its trade-off between energy and accuracy. The prototype IC from [11] with measurement setup is shown in Fig. 4.1 and its summary is presented in Table 4.1.

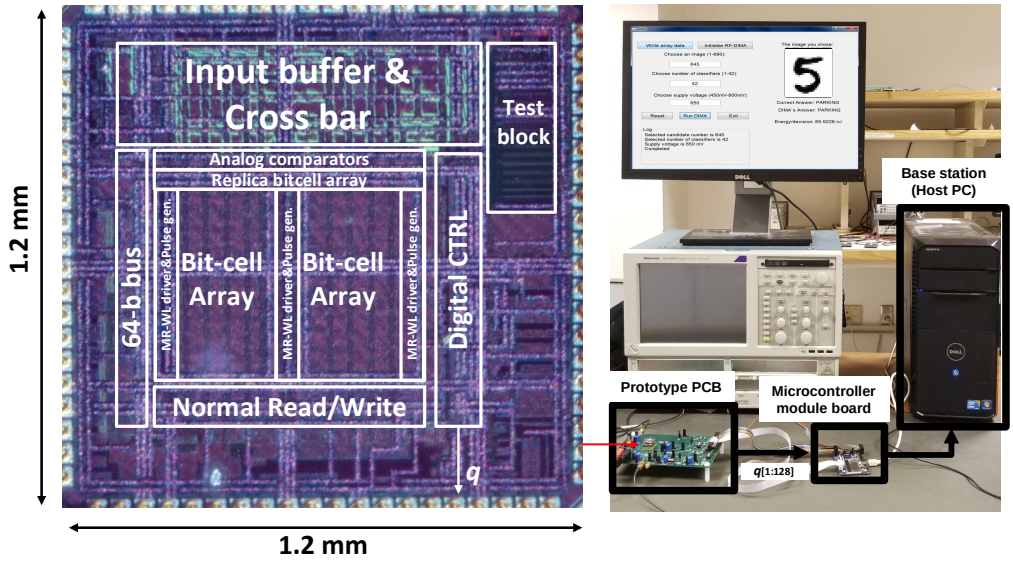
4.1 Component-level Accuracy

Figure 4.2 shows the distribution of measured classification results using first and second weak classifiers in 3 vs. 5 strong classifier with different bit-line voltage swing ΔV_{BL} per-LSB. Blue and red dots represent the measured classification results that are positive and negative, respectively. The black vertical and horizontal lines are used to mark individual ideal thresholds for the first and second weak classifiers. In Fig. 4.2, the classification distribution with lower $\Delta V_{lsb} = 15\,\text{mV}$ shows more imprecise behaviors than the classification with higher $\Delta V_{lsb} = 25\,\text{mV}$ because of higher SNR.

Figure 4.3 shows the measured comparator error rate induced by circuit non-idealities at different BL voltage swings per-LSB for two configurations:

Table 4.1: Measurement summary ($\Delta V_{lsb} = 30$ mV)

	LP mode	**HA mode**	**Hybrid mode**
Energy/decision (nJ/decision)	19	120	67.3
Throughput (decisions/s)	331k	21k	117.7k
MNIST Accuracy	92%	95%	95%



Technology	65nm CMOS	**Die size**	1.2mm × 1.2mm
SRAM capacity	16kB (512 × 256)	**Bit-cell dimension**	2.11μm × 2.11μm
CTRL Frequency	1GHz	**Supply voltage**	1V

Figure 4.1: Chip micrograph [11] and measurement setup.

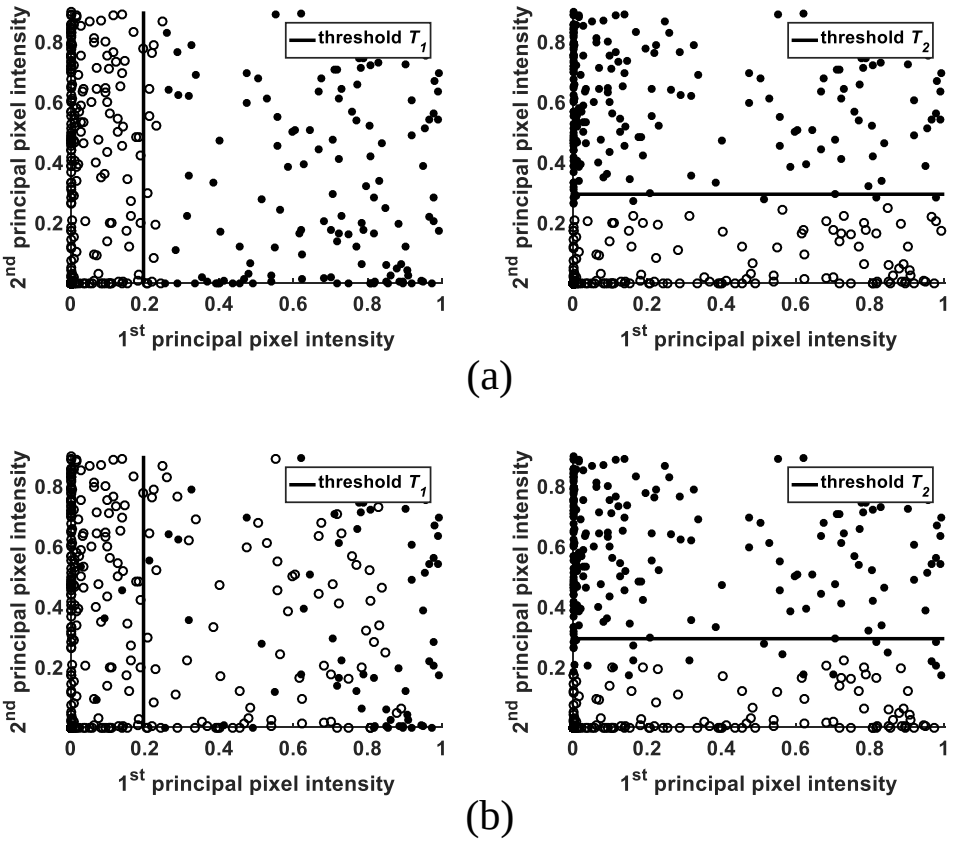


Figure 4.2: Measured results (○: q = 0, ●: q = 1) of the first and second weak classifiers at: (a) $\Delta V_{lsb} = 25\,\mathrm{mV}$, and (b) $\Delta V_{lsb} = 15\,\mathrm{mV}$, where each dot corresponds to one of the MNIST test images for the number 3 and 5.

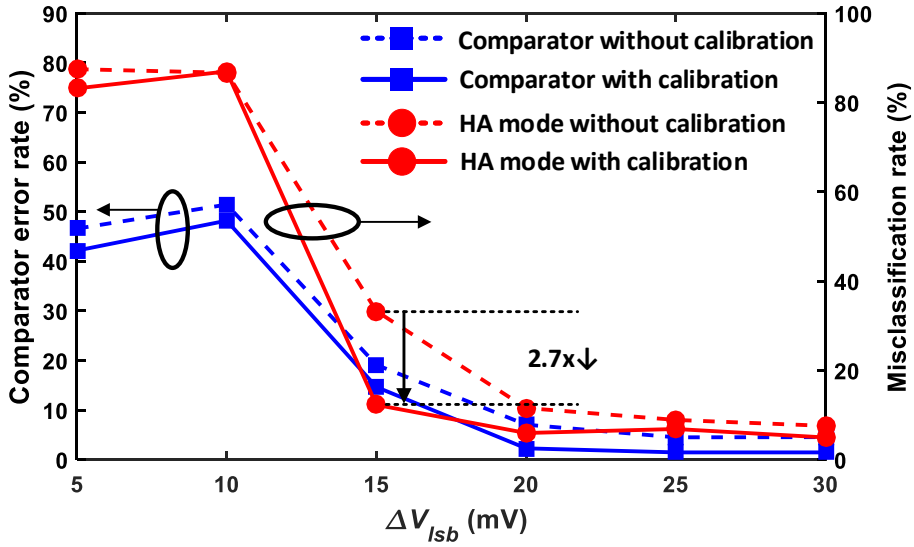


Figure 4.3: Measured results with (without) foreground calibration: comparator error rate vs. misclassification rate w.r.t. ΔV_{lsb} for MNIST dataset.

with and without foreground calibration. The comparison-level errors are measured at each ΔV_{lsb} [1] by counting the errors during the classification with MNIST dataset. As the ΔV_{lsb} increases, the comparator errors reduce from 50% (53%) to 0.7% (2.5%) with (without) foreground calibration. As shown in Fig. 4.3, the foreground calibration dramatically improves task-level misclassification rate by 21% at ΔV_{lsb}= 15 mV. In other words, the optimal operating point can be pushed to the limit where energy efficiency increases at the same level of misclassification rate.

4.2 Task-level Accuracy and Energy

Figure 4.4 shows the misclassification rate for the 10-class MNIST handwritten digit recognition task [19]. As benchmark, we compare with a digital architecture with an identically sized SRAM array and a synthesized digital processor. The energy of the conventional architecture is obtained by measuring the SRAM read energy from the prototype IC [11] and the energy

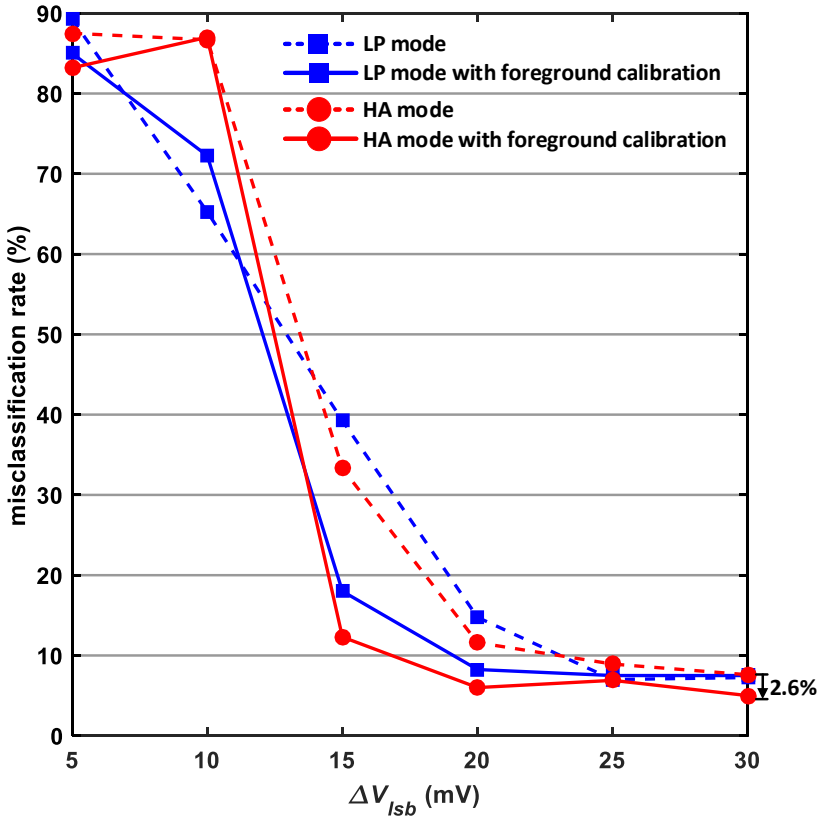


Figure 4.4: Measurement results: error rate vs. ΔV_{lsb} for HA mode and LP mode.

of the digital processor from post-layout simulations. The energy and delay costs of off-chip processing in our architecture are estimated from post-layout simulations. The robustness of classification accuracy to circuit nonidealities is observed as the BL swing ΔV_{lsb} is reduced. Measurements (Fig. 4.5) show 92% (95%) accuracy in the LP (HA) mode at throughput of 331k (21k) decisions/s and energy-efficiency of 19 (120) nJ/decision. This corresponds to 14.7× (7.3×) lower EDP in the LP (HA) mode compared to a conventional digital implementation. In the hybrid mode (Fig. 4.6), 9.70× EDP reduction at accuracy of 95%, energy consumption of 67.3 nJ/decision, and throughput of 117.7K are achieved at $T_h = 2$. In addition, the hybrid mode enables roughly 26× EDP scalability. Figure 4.7 provides a comparison with recent works that use the MNIST dataset. The proposed in-memory AdaBoost re-

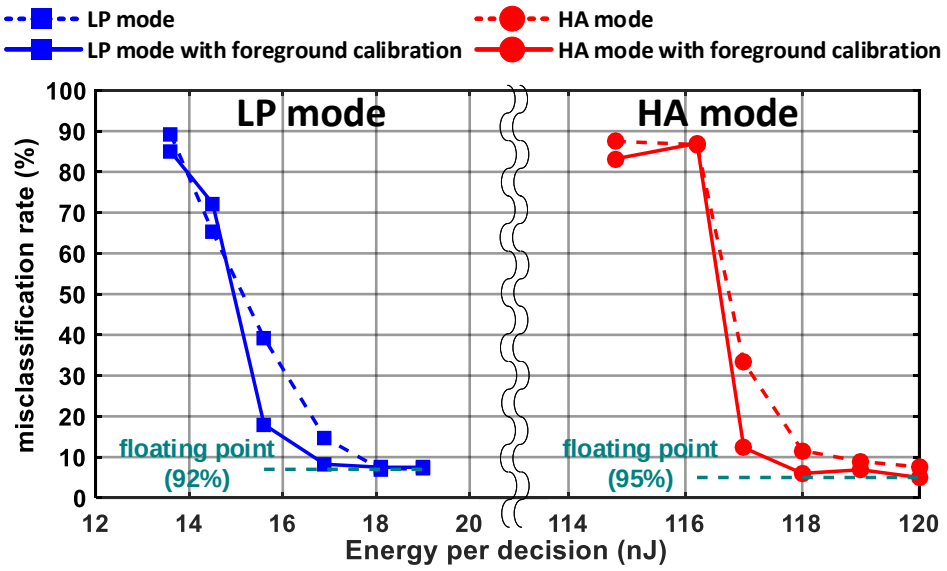


Figure 4.5: Misclassification rate vs. energy in HA mode and LP mode.

alization achieves a 43× reduction in EDP at an iso-accuracy of 95% over a digital neural network [18]. This is a 5% improvement in accuracy over the previous in-memory AdaBoost implementation [13] though at a higher EDP. The higher EDP of our implementation is primarily due to the lower row-parallelism in [11] which reduces the throughput. Furthermore, the proposed architecture can provide energy vs. accuracy scalability by simply adjusting the margin threshold T_h as shown in Fig. 4.7. Note that one of the measurement points in [18] is used for iso-accuracy comparison. The comparison graph (Fig. 4.7) also indicates that hybrid mode provides a better EDP point ($T_h = 2.0$) than HA mode at the same accuracy.

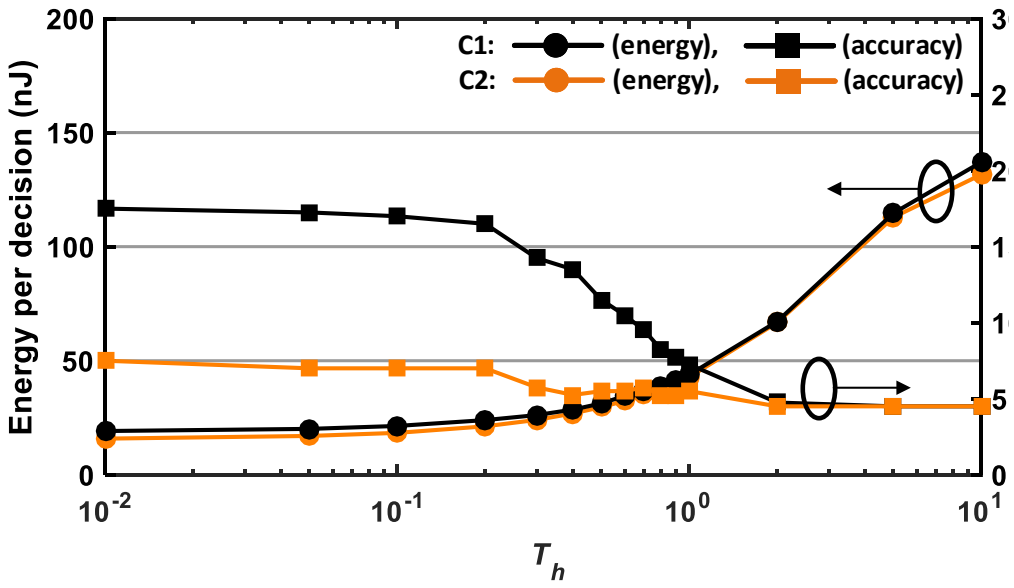


Figure 4.6: Error rate vs. average energy in hybrid mode in two configurations: C1=[ΔV_{lsb}(LP) = 15 mV; ΔV_{lsb}(HP) = 30 mV] and C2=[ΔV_{lsb}(LP) = 30 mV; ΔV_{lsb}(HP) = 30 mV].

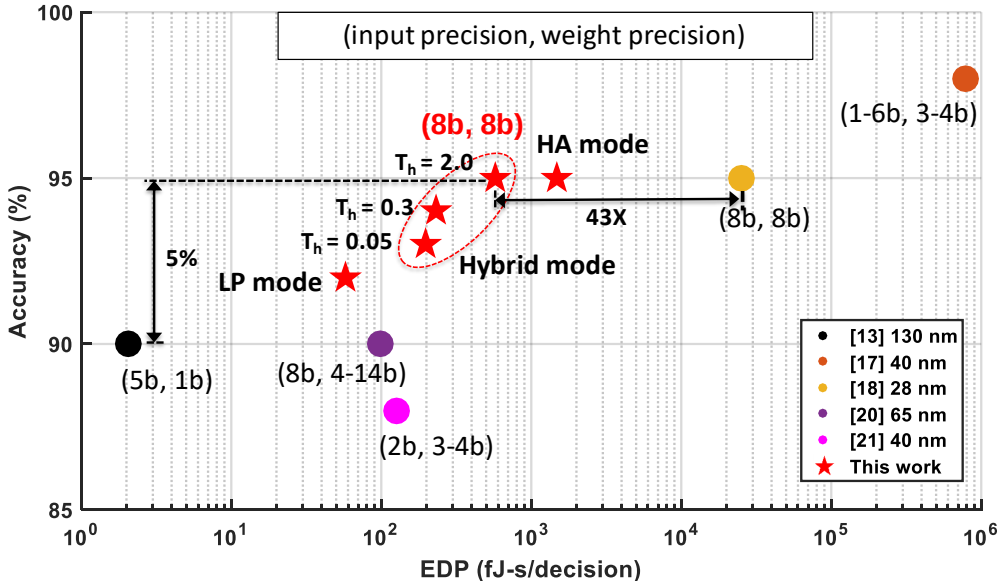


Figure 4.7: Accuracy vs. EDP for 10-class MNIST dataset. Throughput and energy scaled to a 65 nm process [13, 17, 18, 20, 21].

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

This work presents a hierarchical AdaBoost in-memory classifier to provide scalable architecture in the trade-off between energy and accuracy. As a result, the prototype IC demonstrates that the hybrid mode shows EDP reduction of 9.70× (95% accuracy) at energy consumption of 67.3 nJ/decision, compared to a conventional digital architecture for MNIST dataset. Also, this work shows 43× reduction of EDP in the same accuracy and 5% improvement of accuracy at a similar level of EDP. Foreground calibration compensates for the circuit non-idealities of DIMA, achieving task-level accuracy improvement of 21%.

To summarize, there are two crucial points based on the prototype IC measurements. First, a scalable architecture by cascading two different modes achieves high accuracy of the HA mode (95%) and low energy consumption (67.3 nJ/decision) that lies between that of the HA and LP modes. Second, foreground calibration helps maximally utilize benefits of mixed-signal computations in terms of delay and energy without accuracy loss at low retraining cost, showing EDP reduction of 43× compared to the state-of-the-art digital architecture.

5.2 Future Work

Alternative high-density memory technologies such as NAND flash and MRAM can replace SRAM in this work. As the proposed AdaBoost architecture has a simple structure, resource-constrained applications like IoT devices can employ the proposed design to sustain the always-on functionality. Another

extension to on-chip foreground calibration IC can also be considered to achieve higher robustness under severe resource constraints.

REFERENCES

[1] M. Kang, S. K. Gonugondla, A. Patil, and N. R. Shanbhag, "A multi-functional in-memory inference processor using a standard 6T SRAM array," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, 2018.

[2] J. Von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," *Automata Studies*, vol. 34, pp. 43–98, 1956.

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

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

[5] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," in *ACM Sigplan Notices*, vol. 49, no. 4. ACM, 2014, pp. 269–284.

[6] D. Shin, J. Lee, J. Lee, and H.-J. Yoo, "DNPU: An 8.1 TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," in *Solid-State Circuits Conference (ISSCC), 2017 IEEE International*. IEEE, 2017, pp. 240–241.

[7] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "ENVISION: A 0.26-to-10 TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm FDSOI," in *2017 IEEE International Solid-State Circuits Conference-(ISSCC) Digest of Technical Papers*, 2017, pp. 246–247.

[8] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam et al., "Truenorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, 2015.

[9] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers et al., "In-datacenter performance analysis of a tensor processing unit," in *Computer Architecture (ISCA), 2017 ACM/IEEE 44th Annual International Symposium on*. IEEE, 2017, pp. 1–12.

[10] H. Kaul, M. A. Anders, S. K. Mathew, G. Chen, S. K. Satpathy, S. K. Hsu, A. Agarwal, and R. K. Krishnamurthy, "A 21.5 M-query-vectors/s 3.37 nJ/vector reconfigurable k-nearest-neighbor accelerator with adaptive precision in 14nm tri-gate CMOS," in *IEEE International Solid-State Circuits Conference-(ISSCC) Digest of Technical Papers*, 2016, pp. 260–261.

[11] M. Kang, S. K. Gonugondla, S. Lim, and N. R. Shanbhag, "A 19.4 nJ/decision, 364K decisions/s, in-memory random forest multi-class inference accelerator," *IEEE Journal of Solid-State Circuits*, no. 99, pp. 1–10, 2018.

[12] 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*, no. 99, pp. 1–11, 2018.

[13] 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.

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

[15] R. E. Schapire and Y. Freund, *Boosting: Foundations and Algorithms*. MIT Press, 2012.

[16] Y. Tang, J. Zhang, and N. Verma, "Scaling up in-memory-computing classifiers via boosted feature subsets in banked architectures," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2018.

[17] B. Moons and M. Verhelst, "A 0.3–2.6 TOPS/W precision-scalable processor for real-time large-scale ConvNets," in *VLSI Circuits (VLSI-Circuits), 2016 IEEE Symposium on*. IEEE, 2016, pp. 1–2.

[18] 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 *Solid-State Circuits Conference (ISSCC), 2017 IEEE International*. IEEE, 2017, pp. 242–243.

[19] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010, AT&T Labs. [Online]. Available: http://yann.lecun.com/exdb/mnist

[20] J. K. Kim, P. Knag, T. Chen, and Z. Zhang, "A 640m pixel/s 3.65 mW sparse event-driven neuromorphic object recognition processor with on-chip learning," in *VLSI Circuits (VLSI Circuits), 2015 Symposium on*. IEEE, 2015, pp. C50–C51.

[21] F. N. Buhler, P. Brown, J. Li, T. Chen, Z. Zhang, and M. P. Flynn, "A 3.43 TOPS/W 48.9 pJ/pixel 50.1 nJ/classification 512 analog neuron sparse coding neural network with on-chip learning and classification in 40nm CMOS," in *VLSI Circuits, 2017 Symposium on*. IEEE, 2017, pp. C30–C31.