DEEP IN-MEMORY COMPUTING

BY

MINGU KANG

DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical and Computer Engineering in the Graduate College of the University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Doctoral Committee:
Professor Naresh R. Shanbhag, Chair
Professor Rob A. Rutenbar
Associate Professor Pavan Kumar Hanumolu
Associate Professor Naveen Verma, Princeton University
ABSTRACT

There is much interest in embedding data analytics into sensor-rich platforms such as wearables, biomedical devices, autonomous vehicles, robots, and Internet-of-Things to provide these with decision-making capabilities. Such platforms often need to implement machine learning (ML) algorithms under stringent energy constraints with battery-powered electronics. Especially, energy consumption in memory subsystems dominates such a system’s energy efficiency. In addition, the memory access latency is a major bottleneck for overall system throughput. To address these issues in memory-intensive inference applications, this dissertation proposes deep in-memory accelerator (DIMA), which deeply embeds computation into the memory array, employing two key principles: (1) accessing and processing multiple rows of memory array at a time, and (2) embedding pitch-matched low-swing analog processing at the periphery of bitcell array. The signal-to-noise ratio (SNR) is budgeted by employing low-swing operations in both memory read and processing to exploit the application level’s error immunity for aggressive energy efficiency.

This dissertation first describes the system rationale underlying the DIMA’s processing stages by identifying the common functional flow across a diverse set of inference algorithms. Based on the analysis, this dissertation presents a multi-functional DIMA to support four algorithms: support vector machine (SVM), template matching (TM), $k$-nearest neighbor ($k$-NN), and matched filter. The circuit and architectural level design techniques and guidelines are provided to address the challenges in achieving multi-functionality. A prototype integrated circuit (IC) of a multi-functional DIMA was fabricated with a 16 KB SRAM array in a 65 nm CMOS process. Measurement results show up to 5.6× and 5.8× energy and delay reductions leading to 31× energy delay product (EDP) reduction with negligible (≤1%) accuracy degradation as compared to the conventional 8-b fixed-point digital implementation.
optimally designed for each algorithm.

Then, DIMA also has been applied to more complex algorithms: (1) convolutional neural network (CNN), (2) sparse distributed memory (SDM), and (3) random forest (RF). System-level simulations of CNN using circuit behavioral models in a 45 nm SOI CMOS demonstrate that high probability (>0.99) of handwritten digit recognition can be achieved using the MNIST database, along with a 24.5× reduced EDP, a 5.0× reduced energy, and a 4.9× higher throughput as compared to the conventional system. The DIMA-based SDM architecture also achieves up to 25× and 12× delay and energy reductions, respectively, over conventional SDM with negligible accuracy degradation (within 0.4%) for 16×16 binary-pixel image classification. A DIMA-based RF was realized as a prototype IC with a 16 KB SRAM array in a 65 nm process. To the best of our knowledge, this is the first IC realization of an RF algorithm. The measurement results show that the prototype achieves a 6.8× lower EDP compared to a conventional design at the same accuracy (94%) for an eight-class traffic sign recognition problem.

The multi-functional DIMA and extension to other algorithms naturally motivated us to consider a programmable DIMA instruction set architecture (ISA), namely MATI. This dissertation explores a synergistic combination of the instruction set, architecture and circuit design to achieve the programmability without losing DIMA’s energy and throughput benefits. Employing silicon-validated energy, delay and behavioral models of deep in-memory components, we demonstrate that MATI is able to realize nine ML benchmarks while incurring negligible overhead in energy (< 0.1%), and area (4.5%), and in throughput, over a fixed four-function DIMA. In this process, MATI is able to simultaneously achieve enhancements in both energy (2.5× to 5.5×) and throughput (1.4× to 3.4×) for an overall EDP improvement of up to 12.6× over fixed-function digital architectures.
To my wife, children, and my parents for their patience and support.
My deepest thanks go to my wife and children, Steven and Brandon, who have gone through the most hectic period in our lives together. My parents are always proud of my academic career, which was great encouragement to me. I am exceptionally grateful to my adviser, Professor Naresh Shanbhag for his extensive amount of time and effort to train me as an independent researcher. He demonstrated doing his best every single moment, uncompromising high standards, and ceaseless passion for teaching. Especially, I made a mistake in my first prototype IC and wasted significant time and research funds. But he encouraged rather than blamed me and allowed one more opportunity to make up for my mistake. Thanks to his trust in me, I was able to complete this dissertation and build the mixed-signal IC tape-out flow for machine learning in our group. I appreciate Professor Pavan Hanumolu, Professor Naveen Verma, and Professor Rob Rutenbar for their insightful suggestions and for agreeing to be on my committee. Their suggestions during my prelim have greatly affected my research direction and helped to improve my Ph.D. research. I would like to thank Ameya Patil and Yongjune Kim for theoretical analysis of deep in-memory computation in Chapter 2, and Sujan Godugondla for providing simulation models for the convolutional neural network in Chapter 4. It was fantastic to have the opportunity to work with Professor Vikram Adve and his student, Prakalp Srivastava for studying the programmable deep in-memory architecture in Chapter 5. I would also like to sincerely thank Min-sun Keel and Wooseok Choi for their generous help and patient guidance for analog and digital IC tape-out flows. I am also grateful to my research group members, Sungmin Lim, Yingyan Lin, Charbel Sakr, Sujan Godugondla, Ameya Patil, and Dr. Yongjune Kim; my research group alumni, Dr. Sai Zhang and Dr. Eric Kim for their valuable advice. I gratefully acknowledge Systems on Nanoscale Information Fabrics (SONIC), one of the six SRC STARnet Centers, sponsored
by SRC and DARPA. I would also like to acknowledge constructive discussions with Sean Eilert, Ken Curewitz, Professor Naveen Verma, Professor Boris Murmann, and Professor Pavan Hanumolu. Finally, this journey would not have been possible without the members of Korean ECE marathon team "Before sunrise", Minji Kim, Hojeong Yu, Key-whan Chung, and Sungmin Lim. I was able to maintain my physical and spiritual stamina via training with them every Saturday.
# CONTENTS

Chapter 1  INTRODUCTION .............................................................. 1
  1.1 Related Work ................................................................. 4
  1.2 Dissertation Contributions and Organization ...................... 6

Chapter 2  DEEP IN-MEMORY ARCHITECTURE (DIMA) ...................... 9
  2.1 DIMA Overview ............................................................... 9
  2.2 DIMA Design Principles .................................................... 12
  2.3 Functional Read (FR) ....................................................... 19
  2.4 Bitline Processing (BLP) and Cross BLP (CBLP) ..................... 26
  2.5 Models for Circuit Non-Ideal Behavior, Energy, and Delay ........ 32
  2.6 Conclusion .................................................................. 46

Chapter 3  DIMA PROTOTYPE INTEGRATED CIRCUITS ...................... 48
  3.1 Multi-Functional DIMA Architecture .................................. 48
  3.2 Multi-Functional DIMA Operations ..................................... 50
  3.3 Measured Results of Multi-Functional DIMA IC ...................... 56
  3.4 Random Forest (RF) DIMA IC ............................................ 62
  3.5 Conclusion .................................................................. 70

Chapter 4  MAPPING INFEERENCE ALGORITHMS TO DIMA .............. 72
  4.1 Convolutional Neural Network (CNN) .................................. 72
  4.2 Sparse Distributed Memory (SDM) ....................................... 80
  4.3 Conclusion .................................................................. 102

Chapter 5  MATI: DIMA INSTRUCTION SET ARCHITECTURE ............ 103
  5.1 Background .................................................................. 105
  5.2 MATI: A Programmable DIMA ......................................... 109
  5.3 Validation Methodology ..................................................... 119
  5.4 Evaluation Results .......................................................... 124
  5.5 Conclusion .................................................................. 128

Chapter 6  CONCLUSIONS AND FUTURE WORK ............................. 129
  6.1 Dissertation Contributions ............................................... 129
  6.2 Future Work .................................................................. 131

REFERENCES ..................................................................... 134
Current and emerging applications increasingly rely on the ability to extract patterns from large data sets in order to support inference and decision making. These applications rely heavily on machine learning (ML). Though ML algorithms have begun to exceed human performance in cognitive and decision-making tasks [1, 2], they tend to be computationally complex and require processing large data volumes. These tasks have been processed in the cloud platform due to the heavy processing complexity as shown in Fig. 1.1(a) [3]. However, this paradigm requires a large volume of data transfer to data centers and also the extracted information from the cloud back to the electronics causing 9× more energy for the transfer than the processing itself [3]. Therefore, there is increasing interest in embedding data analytics into sensor-rich platforms to provide these decision-making capabilities locally as shown in Fig. 1.1(b). Such platforms often need to implement ML algorithms under severe resource constraints. Primary metrics for the design of such intelligent systems are: (1) energy efficiency, (2) decision latency and throughput, and (3) decision(-making) accuracy. Energy efficiency is critical for embedded battery-powered and autonomous platforms. As a result, a number of integrated circuit (IC) implementations of ML kernels and algorithms have appeared recently [4–13] to address the problems of designing energy efficient ML systems in silicon.

ML algorithms are computationally intensive and require processing of large data volumes. Therefore, the energy consumption of ML hardware comprises the energy costs of memory accesses and arithmetic operations. Of these, memory accesses tend to dominate as each access is expensive, e.g., 20–100 pJ per access of 16-b word from 32 kB to 1 MB SRAM versus 1 pJ per multiplication in a 45 nm process [19]. This observation was confirmed with two simple inference tasks: pattern matching using (1) Manhattan distance and (2) cross
Figure 1.1: Machine learning (ML) platforms: (a) in the cloud, and (b) in silicon (figure courtesy [14–18]).
A cross correlation to find the closest image from an input query image out of 64 candidate images. A SRAM with a 512×256 bitcell array and synthesized digital logic with 8-b in/output precision was employed for post-layout simulations in a 65 nm process. The energy breakdown (Fig. 1.2) demonstrates that memory energy dominates taking up to 90% of the total system energy. In addition, recent implementations of deep neural networks (DNN) [4, 20] also report that memory accesses account for the largest portion (between 35% to 45%) of the total energy cost.

The data access costs are reported in the context of von Neumann architecture, which separates memory from processor (Fig. 1.3(a)). In memory-intensive applications, this separation severely increases memory access energy and limits the throughput, referred to as the von Neumann’s bottleneck [21]. Hence, there is an imperative need to re-think the processor and memory designs for the memory-intensive inference applications.
Figure 1.3: Architectures for inference: (a) conventional, and (b) deep in-memory architecture (DIMA), where \( P \) is an input pattern and \( D \) is stored database.

### 1.1 Related Work

Several research efforts have tried to minimize the data access cost through architectural optimization. Processor-in-memory (PIM) architectures such as Smart Memory [22–24] and Intelligent RAM [25] locate frequently used logic (e.g., pointer logic [24] or MAC [25]) close to memory using a wide crossbar. However, physical proximity does not reduce memory read and processing costs themselves.

An effective approach to reduce memory access energy and enhance throughput for ML algorithms is to reuse the data once it is read from memory. DianNao [20] first identified and exploited an opportunity of massive data reuse across the fetched tile of input and output feature maps in a convolutional neural network (CNN) achieving \( 21 \times \) energy savings. Eyeriss [4, 26] extended the data reuse opportunities at multiple levels (convolutional, filter and input feature map reuse) to achieve up to \( 2.5 \times \) energy savings for AlexNet.

Low-power circuit techniques have also been explored to achieve energy efficient processing for ML algorithms. ENVISION [12] implemented the CNN with a dynamic voltage-accuracy-frequency scaling technique given a bit-precision requirement. A speech recognizer with a deep neural network (DNN) [10] is also introduced with a voice-activated power gating
technique for energy efficiency during stand-by mode. A sparse DNN engine [11] applied the RAZOR technique [27] to allow minimum supply voltage by tolerating timing errors. These approaches achieve significant energy efficiency, but without exploiting the opportunities afforded by analog processing.

Low-voltage SRAM techniques have been proposed [28,29] to reduce the energy of memory read accesses. These techniques involve operating the bitcell array (BCA) at voltages in the range of a few hundred mVs, which reduces the throughput significantly into the kHz regime. The low-voltage operations degrade SRAM’s read and write static margins causing catastrophic failure of inference applications when MSB errors occur. Therefore, SRAM was tailored for inference algorithms to address this issue [30,31]. Here, selective bitline (BL) negative boosting is employed to improve write-ability and protect MSBs during the read operation by selective error correcting code (ECC). However, these techniques suffer from large BL toggling energy and dropping out of LSBs to accommodate ECC check bits, respectively. In [13], a filter approximation technique was employed and accelerated by 7T SRAM to fetch convolution filter coefficients efficiently by enabling two read modes: row-access and column-access, but at the cost of degraded storage density by employing an additional transistor in the bitcell.

To sum up, previous approaches have addressed the energy cost by co-locating the processor and memory, minimizing data accesses (via data reuse) or employing low-power digital techniques for processing and low-voltage memories. In contrast, associative memories [32,33] embed simple logic operations into the BCA to determine a data vector with the minimum Hamming or Euclidean distances from a reference data vector. This is done at the expense of storage density due to the logic circuits added to a bitcell. Kerneltron [34] also embeds computation (bit-wise multiplication) into the BCA to process and read simultaneously in the charge domain. However, this requires the use of charge injection devices in the BCA and a massive array of ADCs to interface analog and digital processing. Moreover, the need for special devices makes it incompatible with mainstream memory topologies such as SRAM or DRAM.
1.2 Dissertation Contributions and Organization

This dissertation proposes deep in-memory architecture (DIMA) [35], which eliminates the separation between memory and processor to minimize the cost of not only memory read but also processing as shown in Fig. 1.3(b). This is achieved by unconventional ways of accessing data from memory and deeply embedding mixed-signal circuitry into the memory. DIMA is characterized by the following:

- multi-row low-swing (e.g., < 30 mV/LSB) memory read: multiple rows of BCA are accessed simultaneously via pulse width or amplitude modulated (PWAM) WL, referred to as multi-row functional read (FR)

- low-swing data processing: pitch-matched mixed-signal circuitry is embedded at the periphery of the BCA for further processing of BL information

- preservation of the standard BCA structure: thereby storage density and conventional read/write functionality are maintained without incurring delay and energy penalty

This dissertation proves the versatility of DIMA by achieving up to $24.5 \times$ reduction of energy-delay product (EDP) in various algorithms such as a template matching (TM) [35], CNN [36], and sparse distributed memory (SDM) [37, 38] with simulations. The DIMA’s concept has also been verified with two prototype ICs demonstrating up to $31 \times$ reduction of EDP. This dissertation also introduces energy, delay, and mixed-signal circuit behavioral models, which are employed to predict the system level’s performance as well as energy and delay trends as a function of major design parameters. Finally, the DIMA platform is extended to programmable instruction set architecture (ISA) to support various ML algorithms and user-friendly programming interface.

While DIMA has strong potential for energy and throughput benefits, the following design challenges need to be addressed without compromising the accuracy:

- Algorithm: the common functional flow across a diverse set of ML algorithms needs to be identified and then mapped on DIMA’s sequential processing stages to cover a
wide variety of ML algorithms.

- **Architecture**: the BCA has to be kept intact, thereby preserving the storage density of standard SRAM. The read/write functionality also needs to be preserved without incurring delay and energy penalty.

- **Circuit**: multiple functions need to be enabled with re-configurable mixed-signal circuitry complying with the stringent row and column pitch-matching requirements imposed by the BCA.

- **Modeling**: accurate statistical modeling of the non-ideal analog circuit behaviors is required to study the impact of non-idealities to the application level’s accuracy. The accuracy needs to be maintained even with diverse noise sources from low-SNR processing.

- **Programmability**: the instruction set needs to be designed considering analog driven DIMA operations. In addition, the throughput and accuracy losses from introducing the programmability should be minimized.

These challenges are addressed in the following chapters:

**Chapter 2** introduces DIMA, its unique features and processing stages. The system-level rationale is also provided to demonstrate DIMA’s robustness in the presence of noise. Furthermore, design techniques and guidelines are provided to address the circuit and architectural levels’ implementation challenges. In addition, this chapter provides energy, delay and behavioral models with the key design parameters. Then, the models are employed to predict the application level’s accuracy.

**Chapter 3** presents two DIMA prototype ICs in a 65 nm CMOS process: (1) multi-functional DIMA that supports four algorithms: support vector machine (SVM), template matching (TM), $k$-nearest neighbor ($k$-NN), and matched filter (MF), and (2) random forest (RF). This chapter also describes design techniques to achieve the re-configurability with analog circuitry. Measurement results of those prototype ICs show up to $31 \times$ and $6.8 \times$ EDP reductions with negligible ($\leq 1\%$) accuracy degradation, respectively.
Chapter 4 applies DIMA to more complex algorithms: (1) CNN, and (2) SDM to show the versatility of DIMA. Especially, algorithm is optimized to maximize the DIMA’s benefit by following two techniques: error-aware retraining for CNN and hierarchical decision in SDM.

Chapter 5 extends the DIMA to programmable instruction set architecture (ISA), namely MATI. The instruction set is designed to be aligned well with the common functional flow of nine ML benchmarks employed in MATI. Simulation results show that the MATI achieves EDP improvement of up to 12.6× over fixed-function digital architectures. In addition, the MATI’s programming overhead is negligible in energy (< 0.1%), and area (4.5%), and throughput, over multi-functional DIMA even with programmability.

Chapter 6 concludes this dissertation and provides future research direction.
Chapter 2

DEEP IN-MEMORY ARCHITECTURE (DIMA)

This chapter provides an overview of DIMA, where a common functional flow across a diverse set of inference algorithms is identified and mapped to the sequential flow of four processing stages on DIMA. DIMA’s underlying design principles are also explained to demonstrate DIMA’s robustness despite its low-swing operations. Furthermore, this chapter provides practical design guidelines and techniques for the circuit and architectural implementations. Finally, energy, delay, and behavioral models of analog circuitry are provided in the following chapters.

2.1 DIMA Overview

Figure 2.1(a) describes typical functional flow of ML algorithms, which compute the vector distance (VD) between N-dimensional vectors $D$ (stored data) and $P$ (input pattern) followed by a thresholding function $f(\cdot)$ to generate the decision $y$. The VD is obtained by computing the element-wise scalar distance (SD) and then aggregating these. Figure 2.1(b) lists required operations for each algorithm (e.g., VD: dot product, and $f(\cdot)$: sign in SVM). This common algorithmic flow is mapped to four sub-blocks of DIMA architecture corresponding to following processing stages: (1) multi-row functional read (FR) for fetching data $D$, (2) BL processing (BLP) for SD computations, (3) cross BL processing (CBLP) for the aggregation of SD results, and (4) ADC and residual digital logic (RDL) for realizing thresholding decision function $f(\cdot)$.

The conventional digital architecture (Fig. 2.2(a)) and DIMA [35–37, 39] (Fig. 2.2(b)) both employ identical BCAs to store $D$ and an input buffer to store streamed $P$. A key difference being, in DIMA, $N_{col}$ analog SD computations are embedded next to the BLs via BLPs, while the digital architecture needs to fetch out the data from the memory before processing.
Figure 2.1: Functional flow of inference algorithms: (a) functional diagram, and (b) operations for each algorithm.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Vector distance</th>
<th>Scalar distance</th>
<th>( f() )</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>Dot product</td>
<td>Multiplication</td>
<td>sign</td>
</tr>
<tr>
<td>TM</td>
<td>Manhattan distance</td>
<td>Absolute difference</td>
<td>min</td>
</tr>
<tr>
<td>( k )-NN</td>
<td>Manhattan distance</td>
<td>Absolute difference</td>
<td>majority vote</td>
</tr>
<tr>
<td>MF</td>
<td>Dot product</td>
<td>Multiplication</td>
<td>max</td>
</tr>
</tbody>
</table>

Figure 2.2: Inference architecture (red marked drivers: turned-on wordline (WL) drivers at a time): (a) conventional system, and (b) deep in-memory accelerator (DIMA).
Table 2.1: DIMA vs. conventional architecture (with $N_{col} \times N_{row}$ bitcell array).

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Conventional</th>
<th>DIMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>data storage pattern</td>
<td>row major</td>
<td>column major</td>
</tr>
<tr>
<td>column mux ratio</td>
<td>$L:1$</td>
<td>$1:1$</td>
</tr>
<tr>
<td>fetched words per access</td>
<td>$N_{col}/(LB)$</td>
<td>$N_{col}$</td>
</tr>
<tr>
<td>$BL$ swing/LSB ($\Delta V_{BL}$)</td>
<td>250 – 300 mV</td>
<td>5 – 30 mV</td>
</tr>
<tr>
<td># of rows per access</td>
<td>1</td>
<td>$B$</td>
</tr>
<tr>
<td>$WL$ driver</td>
<td>fixed pulse width</td>
<td>pulse width/amp modulated</td>
</tr>
</tbody>
</table>

In this way, DIMA can bypass the column muxing requirements imposed on conventional SRAMs. Furthermore, DIMA can directly aggregate the outputs of BLPs by CBLP to generate a VD. Thus, the final output of the analog section in DIMA is a VD instead of data bits as in the case of a digital architecture. These differences are summarized in Table 2.1 and described as follows:

- **Storage pattern:** DIMA stores $B$ bits of $D$ in a column-major format vs. row-major used in the digital architecture (Fig. 2.3(a)).

- **Read access:** per $BL$ precharge (read cycle), DIMA reads a function of $B$ rows or a word-row vs. a single row in the conventional architecture. This process, referred to as multi-row functional read (FR), generates a $BL$ voltage drop $\Delta V_{BL}$ proportional to a weighted sum of the $B$ bits per column [35] by using pulse-width modulated (PWM) (Fig. 2.3(b)) or pulse amplitude modulated (PAM) $WL$ signals [40, 41]. Thus, DIMA needs many fewer precharge cycles to read the same number of bits and this leads to both energy and throughput gains. However, in exchange, DIMA relaxes the fidelity of its reads as long as these fall within the error tolerance of the ML algorithm.

- **Column muxing:** unlike standard SRAMs which require an $L : 1$ column mux ratio (typical $L = 4$ to 32) to accommodate a large-area sense amplifier (SA) as shown in Fig. 2.2(a), DIMA bypasses it via SD computations in BLPs whose horizontal dimen-
sion is matched to the column pitch of the BCA. Column muxing limits the number of bits per access to $N_{col}/L$ in standard SRAM compared to $N_{col}B$ in FR.

- Data reduction and decision: while the conventional architecture computes in digital processor, DIMA implements SD and VD via BLP and CBLP right next to the BCA using charge-based analog circuits. An ADC is used to digitize the analog CBLP output and pass it on to the RDL to compute $f(\cdot)$ in Fig. 2.2(b). This ADC operates once per 128-256 SD computations.

- Accuracy vs. energy: DIMA computations necessarily have a lower signal-to-noise ratio (SNR) than the computations in the digital architecture. This loss in SNR arises from the spatial transistor threshold voltage variations in the BCA which affect the FR process, and due to severe area-constraints on the BLP and CBLP. However, the SNR can be tuned to a level required by the ML algorithm by adjusting the $BL$ swing $\Delta V_{BL}$.

In summary, the key to DIMA’s speed-up and energy advantages over a digital architecture arises from its ability to read $N_{col}B$ bits per access by FR, bypassing the column mux, and via low-swing analog processing. The detailed operations of circuit and architecture in FR, BLP and CBLP stages are described in the following sections.

### 2.2 DIMA Design Principles

This section explains DIMA design principles based on SNR budgeting. The ML algorithms have inherent error resiliency due to following reasons: (1) the thresholding operation into fixed number of classes provides an algorithmic noise margin as small errors do not change the classification results, and (2) the aggregation of a large number of elements, widely used for the dimensionality reduction in ML algorithms, makes the system insensitive to component noise.

The inherent noise immunity can be exploited to achieve aggressive energy and throughput benefits for hardware implementations. Traditionally, supply voltage scaling has been widely
Figure 2.3: Comparing conventional read and DIMA multi-row functional read (FR) operations: (a) fetched data, where $B = 4$ and $L = 4$ assumed, and red marked bitcells read simultaneously, (b) bitline swing ($\Delta V_{BL}$).
Figure 2.4: Functional flow of inference system: (a) conventional system, and (b) DIMA system.

...employed to achieve energy efficiency, but at the cost of significantly degraded throughput (e.g., quadratically degraded to voltage scaling [42]). Moreover, the error tends to happen at MSB in digital processors due to the carry ripple behavior in arithmetic operations [43]. Another potential approach to exploit the ML’s error immunity is reducing the $\Delta V_{BL}$ to save memory read energy. However, the conventional architecture with separated memory and processor does not allow enough $\Delta V_{BL}$ scaling. This is because the $\Delta V_{BL}$ needs to be converted to full swing to be processed in digital logic through SA, which acts as bitwise early decisions (Fig. 2.4(a)). If early decision errors happen at MSB due to $\eta_{1,2,...,N}$ by $\Delta V_{BL}$ scaling, the error magnitude can increase beyond application’s error immunity, leading to a catastrophic failure of the inference system (e.g., peak signal-to-noise ratio (PSNR)
Figure 2.5: Delayed vs. early decision scenarios: (a) simplified functional flow with $x$ and $\hat{x} \in \{1, -1\}$, and (b) bit error probability $p_e$ vs. SNR.

< 10dB with MSB error [30, 31]). Being aware of this issue, several low-power SRAM techniques [30, 31] were proposed for inference applications with selective MSB protection achieving limited (35%) energy savings in the memory, but not in the processor.

To resolve these issues, DIMA eliminates the separation between the low-swing memory and high-swing processor, but budgets SNR by employing aggressively low-swing operations in both memory and process jointly (Fig. 2.4(b)). This is enabled by employing the FR, which implicitly performs D/A conversion, and subsequent mixed-signal BLP and CBLP processings. Despite degraded SNR, the system level robustness of DIMA is maintained.
by following three principles: (1) delayed decision, (2) non-uniform bit protection, and (3) aggregation.

2.2.1 Delayed Decision

Due to the absence of SAs between memory and processor, DIMA does not require early decision right after BL discharge where noise $\eta_a$ is added, but hard decision (thresholding) happens only at the end of classification process (Fig. 2.4(b)), namely delayed decision. In this section, early and delayed decisions are compared with a simple example of binary bit transmission ($x \in \{1, -1\}$) in Fig. 2.5(a), where noise distribution $\eta_{1,2} \sim N(0, \sigma_n^2)$, threshold level 0, and equal prior $P(X = 1) = P(X = -1) = 0.5$ are assumed. The bit error probabilities $p_e$, assuming identical independent noise sources, can be derived as follows:

$$p_e = \begin{cases} 
2Q(\frac{1}{\sigma_n})[1 - Q(\frac{1}{\sigma_n})] & \text{for early decision} \\
Q(\frac{1}{\sqrt{2}\sigma_n}) & \text{for delayed decision}
\end{cases} \quad (2.1)$$

Figure 2.5(b) shows $p_e$ behavior with respect to the SNR based on (2.1). It is clearly shown that the delayed decision is more robust in a low-SNR regime as the early decision tends to amplify noise contribution in the regime. In this sense, the delayed decision improves the robustness of DIMA, where the SNR is tightly budgeted.

2.2.2 Non-Uniform Bit Protection

The FR with PWAM (Fig. 2.2(b)) allows us to assign voltage swing unequally based on the significance of information (bit positions), namely non-uniform bit protection. When the total swing $\Delta V_{BL}$ is budgeted for multi-row reading of $B$ bits, the swing assigned for the $n$-th bit position $\Delta V_n$ is as follows:
\[
\Delta V_n = \begin{cases} 
\Delta V_{BL}/B & \text{for uniform bit protection} \\
\Delta V_{BL}2^{n-1}/(2^B - 1) & \text{for non-uniform bit protection}
\end{cases}
\] (2.2)

For example, the non-uniform bit protection gives roughly $2.1 \times$ more swing for MSB and $3.8 \times$ less swing for LSB compared to uniform bit protection when $B = 4$. In this way, the limited resource (voltage swing) is more efficiently budgeted in the DIMA to reduce the errors in more significant bit positions and thus improve application level robustness.

2.2.3 Aggregation

The charge-sharing process in the CBLP efficiently averages out noise contributions by reducing the standard deviation of output by $\sqrt{N}$ times after aggregating $N$ independent mean-zero random noise sources in Fig. 2.4(b). Specifically, the DIMA’s CBLP is effective in averaging out the noise in the SD computations as the error statistics of noise in the analog circuitry closely follows independent mean-zero behavior (e.g., spatial threshold variation). Moreover, DIMA provides the average-out opportunity for both $\eta_{a1}$ and $\eta_{a2}$ before the thresholding due to the delayed decision whereas the conventional digital architecture goes through the early decision before having the aggregation opportunity.

2.2.4 Measured Result

This section proves the effectiveness of the above mentioned principles based on measurement results of a prototype IC in a 65 nm CMOS process, which is described in more detail in Chapter 3. Figure 2.6(a) shows bit error rate (BER) measured by fetching 16 KB via the conventional SRAM read and the energy trend by scaling $\Delta V_{BL}$. The measured BER is injected to system simulations for face detection using SVM to evaluate the impact on the
Figure 2.6: Bit error rate (BER) vs. $\Delta V_{BL}$ vs. application accuracy (face detection with SVM) in conventional system: (a) BER of normal SRAM read operation measured from prototype IC, and (b) application accuracy obtained by simulating a conventional system with measured BER with measured DIMA accuracy.
application level accuracy in Fig. 2.6(b), where $\Delta V_{BL,avg}$ is the average $BL$ voltage swing assigned per bit for a fair comparison of two systems. The DIMA’s FR fetches $B$-bits with single $BL$ swing $\Delta V_{BL}$ whereas conventional memory reads a single bit, leading to the following definition of $\Delta V_{BL,avg}$:

$$\Delta V_{BL,avg} = \begin{cases} 
\Delta V_{BL} & \text{for conventional memory} \\
\Delta V_{BL}/B & \text{for DIMA}
\end{cases}$$

The setup for SVM including data set and image size is described in Chapter 3. It is shown that 0.7% BER at $\Delta V_{BL} = 230\,mV$ can cause 4% degradation in detection accuracy of the conventional system. On the other hand, DIMA’s measured accuracy shows much more robustness with even lower $\Delta V_{BL,avg}$ due to the above mentioned three principles. Specifically, the effective $\Delta V_{BL}$ of conventional architecture is $L$ times larger than those shown in Fig. 2.6(b) as $L\,BLs$ needs to be discharged to access only one of those through column muxing. Thus, it can be concluded that the DIMA allows more aggressive $\Delta V_{BL}$ scaling compared to the conventional system, leading to significant energy savings.

The following section focuses the realization of the above described system rationale and implementation challenges and solutions.

2.3 Functional Read (FR)

This section explains the FR stage in detail, which performs data access and simple SD computations. Two design techniques: (1) sub-ranged read, and (2) replica bitcell are introduced to enhance the linearity of FR and achieve efficient data writing, respectively. In addition, design principles for key parameters such as pulse width $T_0$ and amplitude $V_{WL}$ are presented to minimize the impact of non-ideal behavior from low-swing analog processing.
2.3.1 FR Operation

The FR stage generates the bitline voltage drop $\Delta V_{BL}(D)$ proportional to the weighted sum $D = \Sigma_{i=0}^{B-1}2^id_i$ of column-major stored data $\{d_0, d_1, ..., d_{B-1}\}$ (see Fig. 2.7(a)). The voltage drop $\Delta V_{BL}(D)$ can be generated via a simultaneous application of PWAM access pulses to multiple rows per precharge cycle. This is in contrast to the use of single-row fixed width and amplitude pulses per precharge cycle in conventional SRAM read.

Consider FR of $B$ rows using PWM [35] with binary-weighted pulse widths $T_i \propto 2^i$ ($i \in [0, B - 1]$) of $V_{WL(i)}$ as shown in Fig. 2.7(b) [35]. The charge $\Delta Q_i(d_i)$ drawn from the $BL$ capacitance $C_{BL}$ is given by:

$$\Delta Q_i(d_i) = d_i T_i I(T_i) \tag{2.3}$$

where $I(t)$ is the current drawn by the $i^{th}$ bitcell. This current can be Taylor series approximated as:

$$I(t) = \frac{V_{PRE}}{R_i} e^{-\frac{t}{RC_{BL}}} \approx \frac{V_{PRE}}{R_i} \left(1 - \frac{t}{R_i C_{BL}}\right) \approx \frac{V_{PRE}}{R_i} \tag{2.4}$$

provided $t \ll R_i C_{BL}$. Substituting $t = T_i$ into (2.4) and the resulting expression for $I(T_i)$
into (2.3), we obtain:

$$
\Delta Q(d_i) = \bar{d}_i T_i \frac{V_{PRE}}{R_i}
$$

where $T_i \ll R_i C_{BL}$. Therefore, the expression for the total $BL$ voltage drop $\Delta V_{BL}(D)$ can be obtained as follows:

$$
\Delta V_{BL}(D) = \sum_{i=0}^{B-1} \frac{\Delta Q_i}{C_{BL}} = \frac{V_{PRE}}{C_{BL}} \sum_{i=0}^{B-1} \bar{d}_i T_i
$$

As the pulse widths are binary weighted $T_i = 2^i T_0$ where $T_0$ is the LSB pulse width, and if $R_i = R_{BL}$, i.e., the discharge paths of all the $B$ bitcells in a column have identical resistances, then

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

where $\Delta V_{lsb} = \frac{V_{PRE} T_0}{R_{BL} C_{BL}}$, and $\bar{D}$ is the decimal value of the one’s complement of $D$. The expression in (2.7) is idealized as it assumes the following four conditions:

1. $T_i \ll R_i C_{BL}$
2. $T_i = 2^i T_0$
3. $R_i = R_{BL}$ (no variation across rows)
4. $R_{BL}$ is a constant over $V_{BL}$.

In practice, these conditions will not be fully met leading to a deviation, i.e., non-linearity, from (2.7), and spatial variations from one group of $B$ bits to another across the BCA. These non-idealities and techniques to alleviate them will be described in Section 2.3.3.

A similar expression as (2.7) for $\Delta V_{BLB}(D)$ can be obtained by replacing $\bar{d}_i$ with $d_i$ in (2.7). Thus, the FR stage converts the stored digital data $D$ into bitline voltage drops $\Delta V_{BL}(D)$ and $\Delta V_{BLB}(D)$, i.e., the FR stage is a digital-to-analog converter. Additionally, the FR stage can also realize simple SD functions such as the addition and subtraction of.
two $B$-bit words ($D$ and $P$) stored different rows but in the same column. For example, from (2.7), $D + P$ is obtained by applying FR to rows containing $D$ and $P$ to obtain:

$$\Delta V_{BL}(D + P) = \Delta V_{lsb}(D + P) = \sum_{i=0}^{B-1} 2^i (d_i + p_i)$$ (2.8)

Similarly, subtraction $D - P$ can be realized by storing $\bar{P}$ (one’s complement of $P$) in the same column as $D$. Subtraction will be discussed in Section 2.4 in more detail.

### 2.3.2 Design Guidelines

The $BL$ swing $\Delta V_{BL}$ generated by the FR stage is subject to the impact of spatial transistor threshold voltage variations caused by random dopant fluctuations [44], voltage-dependence of the discharge path (access and pull-down transistor in the bitcell) resistance $R_{BL}$ (see (2.7)), and the finite transition (rise and fall) times of the PWM $WL$ access pulses. These non-idealities can be incorporated into (2.7) as follows:

$$\Delta V_{BL}(D) = \Delta V_{lsb} \sum_{i=0}^{B-1} \frac{2^i d_i (1 + \gamma_i)}{(1 + \rho_i(V_{BL}) + \delta_i)}$$ (2.9)

where $\delta_i$ is a random variable describing the impact of spatial transistor threshold voltage variations on the discharge path resistance $R_{BL}$ affecting Condition 3 ($R_i = R_{BL}$) in Section 2.3.1, $\rho_i(V_{BL})$ is a variable that captures the impact of the $BL$ voltage-dependence of $R_{BL}$ which affects Condition 4 ($R_{BL}$ should be a constant), and $\gamma_i$ is a deterministic variable that captures the impact of finite transition times on the pulse widths $T_i$ affecting Condition 2 ($T_i = 2^i T_0$), with the affect on the LSB pulse width $T_0$ being most severe.

The presence of $\delta_i$, $\rho_i$, and $\gamma_i$ imposes certain design constraints in order to alleviate their impact so that (2.9) approaches the ideal expression in (2.7). For example, $\rho_i$ can be reduced by ensuring that the access transistor in the discharge path does not transit from saturation into the triode region to satisfy Condition 4. This can be achieved by lowering the $WL$ access pulse amplitude $V_{WL}$, which has the additional benefit that $R_{BL}$ is increased thereby making it easier to satisfy the overarching Condition 1 ($T_i \ll R_{BL} C_{BL}$).
Similarly, the impact of $\gamma_i$ can be alleviated by ensuring that the design parameter $T_0$ is lower bounded as $T_0 > T_{\text{min}}$ so that the rise ($T_r$) and fall ($T_f$) times of $V_{WL}$ are a small fraction, e.g., $T_r + T_f < 0.5T_{\text{min}}$, of $T_0$ and hence Condition 2 ($T_i = 2^iT_0$) can be met. That and Condition 1 implies that $T_{\text{min}} < T_0 < T_{\text{max}}$. Lastly, $\delta_i$ can be alleviated by ensuring that $V_{WL}$ is sufficiently large so that variations in $R_i$ are reduced, i.e., Condition 3 ($R_i = R_{BL}$) is approximated well. This lower bound on $V_{WL}$ can be relaxed as DIMA’s aggregation process in the CBLP compensates for the impact of $\delta_i$. However, $V_{WL}$ does have an upper bound to avoid destructive read operation, e.g., $V_{WL} < 0.8V_{PRE}$. Hence, for the prototype IC, we chose $V_{WL} = 0.65V_{PRE}$. Note that it is possible to pre-distort the data stored in the BCA in order to alleviate the impact of deterministic errors $\rho_i$ and $\gamma_i$.

The worst-case values of $\rho_i$ and $\gamma_i$ are estimated to be less than 41% and 37%, respectively, as estimated from measured results of the multi-functional 65 nm CMOS prototype IC. Monte Carlo post-layout simulations of the BCA shows that the impact of $\delta_i$ leads to a 12% variation ($\sigma/\mu$) in $\Delta V_{BL}(D)$ for typical values of $V_{WL} = 0.65$ V, $V_{PRE} = 1$ V, $T_0 = 250$ ps, $N = 128$, $B = 8$, and $N_{\text{row}} = 512$. Section 3.3.3 indicates that these non-idealities have a negligible impact on inference accuracy for the data sets being considered in this work.

2.3.3 Design Techniques

We present two design techniques to overcome the design constraints described in Section 2.3.2.

- **Sub-ranged Read**: Realizing a highly linear FR stage when $B > 4$ bits is challenging because the constraint $T_{\text{min}} = 2(T_r + T_f) < T_0 < T_{\text{max}} \ll 2^{1-B}R_{BL}C_{BL}$ is hard to meet. For example, $T_0 < 125$ ps when $B = 5$ and $T_4 = 2$ ns. This value of $T_0$ is hard to achieve when driving high WL capacitance (e.g., 200 fF) with a row pitch-matched wordline driver. The sub-ranged read technique solves this problem as described next.

In sub-ranged read [39], the $B/2$ MSBs representing data $D_M$ and $B/2$ LSBs representing data $D_L$ are stored in adjacent columns of the BCA as shown in Fig. 2.8(a). For example, when $B = 8$, $D_M = 8d_7 + 4d_6 + 2d_5 + d_4$ and $D_L = 8d_3 + 4d_2 + 2d_4 + d_0$. Three switches $\phi_{1,2,3}$ are used in specific sequence to charge share the BL voltages. Explicit tuning capacitor
Figure 2.8: Sub-ranged read with \( B = 8 \): (a) BL pair structure (two neighboring bitcell columns), and (b) equivalent capacitance model \([39]\), where \( D_M = 8d_7 + 4d_6 + 2d_5 + d_4 \) and \( D_L = 8d_3 + 4d_2 + 2d_1 + d_0 \).

\( C_{\text{tune}} \) enables the realization of a predefined capacitance ratio \((16 : 1 \text{ for } B = 8)\) between the MSB and LSB BLs’ capacitances \( C_M \) and \( C_L \), respectively. This ratio is needed to weigh the voltage drop on the MSB BL (\( \Delta V_{BLM} \)) by a factor of \( 2^B \) more as compared to the voltage drop on the LSB BL (\( \Delta V_{BLL} \)). The desired capacitance ratio is obtained by setting \( C_M = C_{BL} + C_{BLP} + C_{\text{tune}} \) and \( C_L = C_{BLP} \) as shown in Fig. 2.8(b), and varying \( C_{\text{tune}} \) modifies \( C_M \) so as to realize \( C_M : C_L = 2^B : 1 \).

The sub-ranged read proceeds as follows:

1. the FR process is simultaneously applied to the MSB and LSB columns with \( \phi_{1,2,3} = 0 \) (all open) thereby generating voltage drops \( \Delta V_{BLM}(D_M) \) and \( \Delta V_{BLL}(D_L) \), respectively.

2. the switch \( \phi_1 \) is closed so that the voltage \( \Delta V_{BLM}(D_M) \) is developed across \( C_M \). Switch \( \phi_2 \) is pulsed to generate the voltage \( \Delta V_{BLL}(D_L) \) on \( C_L \).

3. the switch \( \phi_3 \) is closed to generate the voltage drop at the final output

\[
\Delta V_{BL}(D) = \frac{1}{C_M + C_L} \left[ C_M \Delta V_{BLM}(D_M) + C_L \Delta V_{BLL}(D_L) \right] \tag{2.10}
\]

\[
= \frac{1}{2^B + 1} \left[ 2^B \Delta V_{BLM}(D_M) + \Delta V_{BLL}(D_L) \right] \tag{2.11}
\]
Thus, for $B = 8$, the voltage drop $\Delta V_{BLM}(D_M)$ is weighted $16 \times$ more than $\Delta V_{BLL}(D_L)$ to obtain the voltage drop $\Delta V_{BL}(D)$ which is proportional to $16D_M + D_L$.

- **FR Replica BCA:** As described in Section 2.3.1, two operands $D$ and $P$ are required to implement various SD computations. For example, in order to realize the difference $D - P$ (see (2.8)), $P$ is stored in the same column as $D$ but in a different row. Typically, $P$ is a streamed in data, e.g., a template in template matching or image pixels. Storing $P$ in the same BCA as $D$ will require repeated SRAM write operations which incurs large energy and delay costs as these require full $BL$ swing. This problem can be solved via the use of a replica BCA as described next.

---

Figure 2.9: Replica BCA: (a) bitcell column ($B = 4$), and (b) timing diagram for replica BCA writing [39].
Figure 2.10: Pitch-matched layouts of BLP blocks relative to a SRAM bitcell: (a) analog comparator, and (b) a part (1/5) of charge redistribution-based multiplier.

The replica BCA (Fig. 2.9(a)) enables fast writes of $\overline{P}$ via separate write BL (WBL) and WL (WWL) [39]. Thus, $\overline{P}$ can be written into the replica BCA column by providing data in a bit-serial manner via the WBL (Fig. 2.9(b)) while disabling the cross-coupled inverter feedback loop in the replica bitcell via WWL. During a subsequent FR, the replica BCA behaves as an extension of the regular BCA. The layout of replica bitcell needs to be similar to normal bitcell to have the same discharge strength except the needs of WBL and WWL circuitry.

2.4 Bitline Processing (BLP) and Cross BLP (CBLP)

This section describes BLP and CBLP stages, which perform various SD computations and aggregation, respectively. The BLP and CBLP operations rely on tightly pitch-matched analog processing (see Fig. 2.10) resulting in many implementation challenges. Thus, design
principles for key parameters are presented based on the analysis of various noise sources such as charge injection, thermal noise, and coupling noise.

2.4.1 BLP Operation

The $N_{col}$ BLP block in Fig. 2.2(a) accepts two operands: (1) its corresponding bitline voltage drop $\Delta V_{BL}(D)$ generated via the FR stage, and (2) a word $P$ to generate an output voltage $V_B(D, P)$. The BLP block needs to be re-configurable in order to support multiple SD computations required by various ML algorithms. Furthermore, the BLP block layout needs to be column-pitch matched to the BCA. Thus, the BLP stage is a massively parallel analog SIMD processor.

Next, we describe how SD functions such as absolute difference $|D - P|$ [35] and multiplication $DP$ [36] can be implemented in the BLP. These SD functions are required to compute commonly used VDs: Manhattan distance (MD) ($\sum_{i=1}^{N} |D_i - P_i|$) and dot product (DP) ($\sum_{i=1}^{N} D_i P_i$).

2.4.1.1 Absolute Difference

The absolute difference $|D - P|$ can be written as [35]:

$$|D - P| = \max(D - P, P - D) \quad (2.12)$$

From (2.8), the bitline voltage drop corresponding to the difference $D - P$ is obtained as:

$$\Delta V_{BL}(D - P) = \Delta V_{lsb}(\overline{D} + P) \quad (2.13)$$

The intrinsically differential structure of the SRAM bitcell enables one to evaluate $\max(V_{BL}, V_{BLB})$, i.e., the voltages on the $BL$ and $BLB$, quite easily via the use of a local $BL$ compare-select as shown in Fig. 2.11(a). Thus, from (2.12) and (2.13), we get
Figure 2.11: Bitline processing (BLP): (a) absolute difference, where $D$ and $P$ are stored in the same column [35], and (b) charge redistribution-based multiplication with $B = 4$ [36].
\[
\max(V_{BL}, V_{BLB}) = \max(V_{PRE} - \Delta V_{lsb}(P + D), V_{PRE} - \Delta V_{lsb}(D + \overline{P})) \\
= \max(V_{PRE} - \Delta V_{lsb}(P + 2^B - 1 - D), V_{PRE} - \Delta V_{lsb}(D + 2^B - 1 - P)) \\
= V_{PRE} - (2^B - 1)\Delta V_{lsb} + \Delta V_{lsb,\text{max}}(P - D, D - P)
\] (2.14)

Thus, applying FR to \(D\) and \(\overline{P}\) simultaneously results in \(V_{BL}\) and \(V_{BLB}\) being proportional to \(P - D\) and \(D - P\), respectively. The local BL compare-select block (Fig. 2.11(a)) provides the maximum of \(V_{BL}\) and \(V_{BLB}\), and hence the absolute difference \(|D - P|\).

### 2.4.1.2 Multiplication

Figure 2.11(b) shows a charge redistribution-based mixed-signal multiplier with inputs \(\Delta V_{BLB}(D)\) (FR stage output) and an externally provided \(B\)-bit digital word \(P\), whose bits \(p_i\) control the \(\phi_{2,i}\) switches. The multiplier output voltage \(V_B\) is given by:

\[
V_B(D, P) = V_B(DP) = V_{PRE} - (0.5)^BP\Delta V_{BLB}(D) = V_{PRE} - (0.5)^B\Delta V_{lsb}DP
\] (2.15)

Thus, voltage drop \(\Delta V_B(DP) = V_{PRE} - V_B(DP) \propto DP\) represents the product of \(D\) and \(P\).

Note that, the multiplier employs unit size (25 fF) capacitors rather than binary-weighted ones as in [45], due to stringent column pitch-match constraints on the BLP.

The timing diagram in Fig. 2.11(b) describes the operation of the multiplier, which is also summarized below:

- first, the unit capacitors are charged to \(\Delta V_{BLB}(D)\) by pulsing the \(\phi_{\text{damp}}\) switches.

- then, the switch \(\phi_{2,i}\) is pulsed only if \(p_i = 0\) thereby charging the capacitor corresponding to \(i\)-th bit \(V_{PRE}\). The capacitors corresponding to \(p_i = 1\) retain the voltage equal to \(\Delta V_{BLB}(D)\).

- finally, the switches \(\phi_{3,i}\) are pulsed sequentially starting from \(\phi_{3,0}, \phi_{3,1} \ldots \phi_{3,B-1}\). This leads to charge sharing between adjacent capacitors.
Note that when $\phi_{3,k}$ ($k = 0, \ldots, B - 1$) is pulsed, the two charge sharing capacitors settle to a voltage of:

$$V_{PRE} - (0.5)^{k+1}(2^k p_k + \cdots + 2p_1 + p_0)\Delta V_{BLB}(D)$$

thereby realizing (2.15) when $k = B - 1$.

2.4.2 Cross Bitline Processing (CBLP), ADC, and Residual Digital Logic (RDL)

The cross BL processor (CBLP) (Fig. 2.12) samples the output voltage ($V_B(D, P)$) of the BLP on the BL-wise sampling capacitors $C_S$ at each column by pulsing the $\phi_1$ switches. Next the $\phi_2$ switches are pulsed to generate the CBLP output $V_C$ in one step. In this way, CBLP implements dimensionality reduction, which is a widely used function in inference algorithms. Finally, the CBLP output $V_C$ is converted to the digital domain by the ADC to be stored or further processed by the RDL. The RDL implements slicing/thresholding functions such as min, max, sign, sigmoid, and majority vote. Note that the ADC and RDL need to process one scalar value ($V_C$) generated from a massively parallel ($> 128$) SD processing step in the BLP. Thus, the energy overhead of ADC and RDL is negligible.
2.4.3 Design Guidelines

The BLP can be configured to compute the absolute difference $|D - P|$ (MD mode) or the scalar product $DP$ (DP mode).

The dominant source of non-ideality in computing $|D - P|$ is due to the comparator offset in the compare-select block (see Fig. 2.11(a)). However, this input offset affects the BLP output $V_B$ minimally. This is because the input offset affects the output only when $V_{BL}$ and $V_{BLB}$ are close to each other. Thus, the BLP output $V_B = \max(V_{BL}, V_{BLB})$ in (2.12) is supposed to have an error of only small magnitude $|V_{BL} - V_{BLB}|$. Additionally, the error in $V_B$ being uncorrelated across the columns gets averaged out further by the CBLP. The column pitch-matched comparator layout shown in Fig. 2.10(a) is constrained to be symmetric to minimize the input offset. Monte Carlo post-layout simulations in the 65 nm CMOS process indicates that the input offset follows the distribution $\mathcal{N}(0, (10 \text{mV})^2)$.

Computation of the product $DP$ in the BLP (see Fig. 2.11(b)) and summation in the CBLP (see Fig. 2.12) is done via charge redistribution circuits. These circuits suffer from multiple noise sources: (1) charge-injection noise, (2) coupling noise, and (3) thermal noise. Assuming a junction capacitance of 0.05 fF [46] using minimum sized switches, we find that storage capacitance $C$ needs to be larger than 13 fF in order to ensure 8-b output precision. The 8-b precision in a swing of 300 mV results in a resolution of $V_{\text{res}} = 1 \text{mV}$. Hence, thermal noise considerations ($\sqrt{KT/C} < 0.5V_{\text{res}}$) lead to the requirement of $C > 17 \text{ fF}$ at $T = 300 \text{ K}$. Hence, we chose $C = 25 \text{ fF}$ to provide sufficient design margin.

Due to the tight pitch-matching constraints, digital signals need to be routed over analog nodes in the BLP and CBLP generating significant coupling noise. In order to alleviate coupling noise, low-swing analog nodes were shielded from the digital full-swing lines as shown in Fig. 2.10(b).
2.5 Models for Circuit Non-Ideal Behavior, Energy, and Delay

The analog-intensive DIMA operation is subject to a number of circuit-level non-idealities [35]. This section presents comprehensive behavioral models of dominant non-idealities in each analog signal processing step for the prediction of application accuracy. Energy and delay models are also provided as a function of major design parameters.

The major non-idealities are as follows:
(a) non-linearity of the FR process due to the voltage-dependent resistance of BL discharge path
(b) local transistor threshold voltage $V_{th}$-mismatch across bitcells caused by random dopant fluctuations
(c) non-ideal sub-ranged read due to the inaccuracy of capacitance ratio between the MSB and LSB columns
(d) input offset of the analog comparator

This section focuses on Manhattan distance (MD) to analyze a variety of noise sources. A similar analysis can be applied to the other VD kernels.

2.5.1 Circuit-Aware Behavioral Model

2.5.1.1 FR with Subtraction (summation)

The FR for MD computation is simulated with HSPICE and the non-linearity is modeled by a polynomial equation given by:

$$\Delta V_{BL}'(D, P) = c_2(D + P)^2 + c_1(D + P) + c_0$$

(2.17)

where $\Delta V_{BL}'(D, P)$ is a distorted version of $\Delta V_{BL}(D, P)$, and the fitting parameter $c_0$, $c_1$, and $c_2$ depend upon $R_{BL}$, $C_{BL}$, $T_0$, the process parameters including $V_{th}$, carrier mobility, saturation carrier velocity, and the channel length modulation parameter. The expressions

---

for $\Delta V'_{BLB}(D, P)$ can be obtained by substituting $P$ and $\overline{P}$ in (2.17) with $\overline{D}$ and $D$, respectively.

2.5.1.2 Variation of FR

The impact of $V_{th}$-mismatch is modeled as Gaussian distributed random variables as shown below:

$$\Delta \hat{V}_{BL}(D, P) \sim \mathcal{N}(\Delta V'_{BL}(D, P), \sigma^2_{\Delta V_{BL}(D, P)}) \quad (2.18)$$

where $\sigma^2_{\Delta V_{BL}(D, P)}$ is the variance of $\Delta \hat{V}_{BL}(D, P)$ due to $V_{th}$-mismatch across bitcells.

The variance $\sigma^2_{\Delta V_{BL}(D, P)}$ is expressed as follows by assuming that the BL voltage drop from reading $D$ and $\overline{P}$ are independent:

$$\sigma^2_{\Delta V_{BL}(D, P)} = \sigma^2_{\Delta V_{BL}(D)} + \sigma^2_{\Delta V_{BL}(\overline{P})} \quad (2.19)$$

Furthermore, the voltage drop by FR of $D$ can be modeled as an addition of binary scaled $B$ independent Gaussian random variables when sub-ranged read is employed for every $B$ bits. Thus, the variance $\sigma^2_{\Delta V_{BL}(D)}$ can be expressed by:

$$\sigma^2_{\Delta V_{BL}(D)} = \{(2^{B-1}d_{B-1})^2 + \ldots + (2^2d_2)^2 + (2d_1)^2 + (d_0)^2\} \times \sigma^2_{\Delta V_{BLB}(D=1)} \quad (2.20)$$

The $\sigma^2_{\Delta V_{BL}(\overline{P})}$ can be achieved by replacing the $\overline{d}_i$ in (2.20) into $p_i$.

2.5.1.3 Sub-Ranged Read

The parasitic capacitance ratio between $BLM$ and $BLL$ needs to be $M(= 2^{B/2}) : 1$ for the sub-ranged read as shown in (2.10). Inaccuracy of the ratio $(M : 1)$ causes a non-ideality, which is modeled by:
\[ \hat{V}_{BL}(D, P) = \frac{M}{M+1} [\{V_{PRE} - \Delta \hat{V}_{BLM}(D_M, P_M)\} + \{V_{PRE} - \Delta \hat{V}_{BLL}(D_L, P_L)\}/M] \] (2.21)

### 2.5.1.4 Absolute Operation

The non-ideal BLP output \( \hat{V}_{B,j} \) from the \( j \)-th column in DIMA is expressed with the ideal value \( V_{B,j} \) as follows:

\[ \hat{V}_{B,j} = g_1 \{Q\hat{V}_{BL}(D_j, P_j) + \overline{Q}\hat{V}_{BLB}(D_j, P_j)\} + g_0 \] (2.22)

\[ Q = \begin{cases} 
1 & \text{if } \hat{V}_{BL}(D_j, P_j) > \hat{V}_{BLB}(D_j, P_j) + V_{OS} \\
0 & \text{otherwise}
\end{cases} \]

where \( V_{OS} \sim \mathcal{N}(0, \sigma_{comp}^2) \) is input offset voltage of the comparator which is modeled as a zero mean Gaussian random variable with variance \( \sigma_{comp}^2 \). Coefficients \( g_0 \) and \( g_1 \) model voltage change due to the charge sharing between the BL and sampling capacitor, and leakage current.

### 2.5.1.5 Sampling and Capacitive Addition

The thermal noise in the sampling capacitor is negligibly small as compared to other noise sources by employing capacitors larger than \( 10 \text{ fF} \). The non-ideal CBLP output (\( \hat{V} \)) from charge sharing \( N \) sampling capacitances can be modeled as follows:

\[ \hat{V}_{C} = \frac{1}{N} \sum_{j=1}^{N} \hat{V}_{B,j} \] (2.23)

### 2.5.2 Throughput Model

The throughput of conventional systems is limited by the memory access delay in memory-intensive algorithms. In such applications, throughput improvement factor \( S \) by the DIMA
can be expressed as follows:

\[ S = \frac{1}{2}(LB)\left(\frac{T_{conv}}{T_{DIMA}}\right) \]  

(2.24)

where the \( T_{conv} \) is the cycle time for single read access of conventional system, and \( T_{DIMA} \) is the sum of the delays of FR, BLP, and CBLP. The \( T_{DIMA} \) is generally larger than \( T_{conv} \) as DIMA accesses multiple rows to fetch a vertically stored word and complete the following analog operations. The scaling factor \( B \) is due to the FR process reading \( B \) rows per access. In addition, the DIMA bypasses the \( L : 1 \) column mux resulting in scaling factor \( L \). The scaling factor \( \frac{1}{2} \) occurs because the sub-ranged read reduces effective throughput. Hence, higher throughputs \((S = 3.2 \sim 12.8)\) can be easily achieved, e.g., by setting \( B = 8 \), \( L = 4, 8, 16 \), and \( T_{DIMA} = 5T_{conv} \).

2.5.3 Energy Model

The energy consumptions of DIMA and conventional architecture per read operation and following processing of a \( B \)-bit word for MD computation are modeled, respectively, as follows:

\[ E_{conv} = BLC_{BL}\Delta V_{BL}V_{PRE} + BE_{SA} + E_{leak} + E_{logic} \]  

(2.25)

\[ E_{DIMA} = 4C_{BL}(\Delta V_{BL})V_{PRE} + E_{comp} + E_{leak}/S + E_{BLP} \]  

(2.26)

The energy efficiency of DIMA can be evaluated by comparing (2.25) and (2.26). The energy component \( E_{logic} \) in (2.25) corresponds to \( E_{BLP} \) in (2.26). The \( E_{comp} \) in (2.26) is the energy for an analog comparator, and this is almost the same as the sense amplifier energy \( E_{SA} \) in (2.25). The first term in (2.26) does not have the scaling factor of \( L \) as DIMA bypasses the \( L : 1 \) column muxing. The scaling factor \( B \) is missing in the first and second
terms of (2.26) because the FR processes a $B$-bit word per single $BL$ discharge. However, the scaling factor of four is created in the first term of (2.26) to read not only $D$ but also $T$ and employ the sub-ranged read. Energy savings can be observed by comparing the first and second terms because $LB$ ranges from 32 to 128 when $B = 8$. It is assumed that a deep-sleep mode is enabled during standby using techniques such as power gating or lowering the $V_{DD}$ for the BCA [47–49]. Therefore, $E_{\text{leak}} = P_{\text{leak}}T_{\text{conv}}$, where $P_{\text{leak}}$ is the leakage power consumption. The DIMA’s intrinsically parallel operation makes its effective read cycle time smaller by a speed-up factor of $S$ ($S > 3$), therefore the leakage energy is also reduced. The $E_{BLP}$ is smaller than the $E_{\text{logic}}$ due to a low-swing signal processing.

### 2.5.4 Prediction of Application Accuracies

In this section, the application accuracy is predicted in terms of probability of detection ($P_{\text{DET}}$) as a function of major design parameters based on the behavioral models described in the previous sections.

The decision of pattern matching with the DIMA based on distance metrics can be represented as follows:

$$u_{\text{opt}_{-}\text{DIMA}} = \arg \min_{u} \{ f(u) + \eta_{u}, \ u = 1, ..., U \}$$

(2.27)

where $u_{\text{opt}_{-}\text{DIMA}}$ is the decision from DIMA to find the index of the candidate with minimum distance. The $f(\cdot)$ is a error-free distance metric, $U$ is the number of candidates, and $\eta_{u}$s are independent random variables representing noise due to the non-idealities of FR, BLP, and BLP for the $u$-th candidate. The $P_{\text{DET}}$ of DIMA can be defined as follows:

$$P_{\text{DET}} = \Pr \{ u_{\text{opt}} = u_{\text{opt}_{-}\text{DIMA}} \}$$

$$= \Pr \{ f(u_{\text{opt}_{-}\text{DIMA}}) + \eta_{u_{\text{opt}_{-}\text{DIMA}}} < f(u) + \eta_{u}, \ u = 1, ..., U \}$$

(2.28)

where $u_{\text{opt}}$ is the correct index of the candidate with minimum distance. The circuit-aware behavioral models from (2.17) to (2.23) will be applied to (2.28) to predict the application-
tion level accuracy. In this section, the only local transistor $V_{th}$-mismatch across bitcells is considered as the only source of non-ideality in DIMA as it is the dominant source of error.

Equation (2.21) can be expressed with an error-free ideal $BL$ voltages as follows:

\[
\hat{V}_{BL}(D, P) = V_{PRE} - \left(\frac{M}{M+1}\right)\Delta V_{BLM}(D_M, P_M) - \left(\frac{1}{M+1}\right)\Delta V_{BLL}(D_L, P_L) + \eta_{\Delta V_{BL}}
\]

\[
= V_{BL}(D, P) + \eta_{\Delta V_{BL}} \tag{2.29}
\]

where $\eta_{\Delta V_{BL}}$ is additive Gaussian noise with mean zero and variance described by:

\[
\sigma^2_{\Delta V_{BL}(D, P)} = \frac{M^2}{(M+1)^2} \sigma^2_{\Delta V_{BL}(D_M, P_M)} + \frac{1}{(M+1)^2} \sigma^2_{\Delta V_{BL}(D_L, P_L)} \tag{2.30}
\]

Similarly, $\hat{V}_{BL}(D, P)$ can be denoted as follows:

\[
\hat{V}_{BL}(D, P) = V_{BL}(D, P) + \eta_{\Delta V_{BL}} \tag{2.31}
\]

The non-ideal BLP output $\hat{V}_{B,j}(D, P)$ in (2.22) can be described with equivalent additive noise $\eta_{B,j}$ as follows, but neglecting the comparator offset for simplicity:

\[
\hat{V}_{B,j} = g_1 \max\{V_{BL}(D_j, P_j) + \eta_{\Delta V_{BL}(D_j, P_j)}, V_{BLB}(D_j, P_j) + \eta_{\Delta V_{BLB}(D_j, P_j)}\} + g_0
\]

\[
= V_{B,j} + \eta_{B,j} \tag{2.32}
\]

where $V_{B,j} = g_1 \max\{V_{BL}(D_j, P_j), V_{BLB}(D_j, P_j)\} + g_0$, which is an error-free BLP output. The $\eta_{B,j}$ is not a Gaussian random variable due to the maximum operation although $\eta_{\Delta V_{BL}}$ and $\eta_{\Delta V_{BLB}}$ are independent Gaussian random variables [50]. However, the mean and variance of $\eta_{B,j}$ can be expressed as follows [51]:

\[
E[\hat{V}_{B,j}] = g_1[\mu_1\Phi(\alpha) + \mu_2\Phi(-\alpha) + a\phi(\alpha)] \tag{2.33}
\]
\[ \text{Var}[\hat{V}_{B,j}] = g_1^2(\mu_1^2 + \sigma_1^2)\Phi(\alpha) + g_1^2(\mu_2^2 + \sigma_2^2)\Phi(-\alpha) \\
+ g_1^2 a(\mu_1 + \mu_2)\phi(\alpha) - (E[\hat{V}_{B,j}])^2 \]  
(2.34)

where

\[ \mu_1 = V_{BL}(D_j, P_j); \quad \mu_2 = V_{BLB}(D_j, P_j); \quad \sigma_1^2 = \sigma_{\Delta V_{BL}(D,P)}^2; \quad \sigma_2^2 = \sigma_{\Delta V_{BLB}(D,P)}^2; \]

\[ a^2 = \sigma_{\Delta V_{BL}(D,P)}^2 + \sigma_{\Delta V_{BLB}(D,P)}^2; \quad \alpha = \frac{V_{BL}(D_j, P_j) - V_{BLB}(D_j, P_j)}{a} \]

\[ \phi(x) \text{ and } \Phi(x) \text{ are the Gaussian probability density function and cumulative density function, respectively.} \]

The \( \hat{V}_{B,j} \) in (2.23) can be substituted by (2.32) as follows:

\[ \hat{V}_C = V_C + \eta_C \]  
(2.35)

where \( V_C = \frac{1}{N} \sum_{j=1}^{N} V_{B,j} \) and \( \eta_C = \frac{1}{N} \sum_{j=1}^{N} \eta_{B,j} \) are equivalent additive noise present in \( V_C \). Although \( \eta_{B,j} \) is not a Gaussian random variable, the distribution of \( \eta_C \) approaches a Gaussian distribution \( \mathcal{N}(E[\eta_C], \text{Var}[\eta_C]) \) as \( L \) increases by the central limit theorem [50] with the following mean and variance:

\[ E[\eta_C] = \frac{1}{N} \sum_{j=1}^{N} E[\hat{V}_{B,j}] - V_C \]

\[ \text{Var}[\eta_C] = \frac{1}{N^2} \sum_{j=1}^{N} \text{Var}[\hat{V}_{B,j}] \]  
(2.36)

Now (2.28) can be expressed with \( \eta_C \), and the \( P_{DET} \) is given by:

\[ P_{DET} = \text{Pr}\{f(u_{\text{opt}}) + \eta_{C,u_{\text{opt}}} < f(u) + \eta_{C,u}, u = 1, \ldots, U\} \]  
(2.37)
where error-free MD computation kernel $f(\cdot)$ can be expressed as follows:

$$f(u) = \frac{1}{L} \sum_{j=1}^{L} [g_1 \max\{V_{BLB}(D_{u,j}, P_j), V_{BL}(D_{u,j}, P_j)\} + g_0] \quad (2.38)$$

For simplicity, the probability that every $f(u) + \eta_{C,u}$ in (2.37) is larger than constant value $x$ is defined by:

$$h(x) = \prod_{u \neq u_{opt}} \Pr\{x < f(u) + \eta_{C,u}\} \quad (2.39)$$

$$= \prod_{u \neq u_{opt}} Q\left(\frac{x - (f(u) + E[\eta_{C,u}])}{\sqrt{\text{Var}[\eta_{C,u}]}}\right)$$

where $x$ ranges from 0 to 1 as the voltage level of CBLP output cannot be less than 0 and higher than $V_{DD} = 1$.

The $f(u_{opt}) + \eta_{C,u_{opt}}$ in (2.37) is a random variable whereas $x$ in (2.39) is a constant value. Thus, $h(x)$ is integrated with probability density function of $f(u_{opt}) + \eta_{C,u_{opt}}$ over the entire dynamic voltage range as follows:

$$P_{DET} = \int_{x=0}^{1} h(x) \phi\left(\frac{x - (f(u_{opt}) + E[\eta_{C,u_{opt}}])}{\sqrt{\text{Var}[\eta_{C,u_{opt}]}}\right) dx \quad (2.40)$$

### 2.5.5 Simulation Results

This section provides validation of behavioral models from (2.17) to (2.23) by HSPICE Monte-Carlo simulations in a 65 nm CMOS process technology. Then, $P_{DET}$ of practical application is obtained via Monte-Carlo simulations using the validated behavioral models. Finally, the $P_{DET}$s from Monte-Carlo system simulations are compared with the $P_{DET}$ predicted by (2.40).
Table 2.2: Design and model parameters in (2.17)-(2.23).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{DD}$</td>
<td>1 V</td>
<td>$V_{WL}$</td>
<td>0.4 - 0.9 V</td>
</tr>
<tr>
<td>$V_{PRE}$</td>
<td>1 V</td>
<td>$L$</td>
<td>4, 8, 16</td>
</tr>
<tr>
<td>array size</td>
<td>256 × 512</td>
<td>$N$</td>
<td>2 - 256</td>
</tr>
<tr>
<td>$T_0$</td>
<td>300 ps</td>
<td>$M$</td>
<td>16</td>
</tr>
<tr>
<td>$B$</td>
<td>8</td>
<td>CLK freq.</td>
<td>1 GHz</td>
</tr>
<tr>
<td>$\sigma_{comp}$</td>
<td>10 mV</td>
<td>$g_0, g_1$</td>
<td>0.890, 0.086</td>
</tr>
<tr>
<td>$\frac{\sigma_{\Delta V_{BL}}}{\mu_{\Delta V_{BL}}(D = 1)}$</td>
<td>6.4%</td>
<td>$c_0 = -3.04 \times 10^{-3}$, $c_1 = 0.037$, $c_2 = -2.43 \times 10^{-4}$</td>
<td></td>
</tr>
</tbody>
</table>

2.5.5.1 System Configurations

Two banks with $512 \times 256$ BCA per bank are assumed to store a dataset of $D$s. Bit precision $B = 8$ is chosen for $D$ and $P$, and sub-ranged read is applied for every 4 bits as shown in Fig. 2.8(a). The $L$ is chosen to be 4, 8, and 16, which are the most widely used values. It is assumed that $\Delta V_{BL(B)} = 250$ mV and $V_{WL} = 1$ V in a conventional SRAM to achieve sensing margin. A metal oxide metal (MOM) capacitor is used to implement the sampling capacitor in order to balance leakage reduction and area efficiency. The analog comparator is sized to fit in the horizontal dimension of the SRAM bitcell, and 10 mV of input offset is assumed for system simulations [52]. Numerical values of design and model parameters validated in the following paragraphs are summarized in Table 2.2.

2.5.5.2 Model Validations

Figure 2.13 shows the result from HSPICE simulations for FR when addition (or subtraction) of 4-bit data $D$ and $P$ are obtained by FR. The non-linearity is measured by integral non-linearity (INL), and the dynamic range of $\Delta V_{BLB}$ is limited to 0.9 V in order to obtain INL within 2.5 LSB. The fitting parameters for the model of (2.17), $c_0 = -0.003041$, $c_1 = 0.037$, and $c_2 = -0.000243$, result in a modeling error less than 0.3 LSB. Figure 2.14 shows the accuracy of sub-ranged read for $D$ with $B = 8$ from HSPICE simulation. The INL is less than 6 LSBs and the modeling errors of (2.21) with $M = 16$ is smaller than 1.6 LSBs.

The complete behavioral models covering the entire analog signal processing chain from
Figure 2.13: $\Delta V_{BLB}$ from simulations and model (2.17) during FR with $B = 4$, $V_{PRE} = 1$ V, and $T_0 = 300$ ps.

Figure 2.14: Sub-ranged read with HSPICE simulations and behavioral models with $M = 16$ (2.21).
the bitcell \( (D) \) to the final output \( (\hat{V}_C) \) with \( N = 8 \) are validated in Fig. 2.15. The \( V_C \) is achieved from HSPICE Monte-Carlo simulations, and compared with the output obtained from the behavioral models described by (2.17)-(2.23) as shown in Fig. 2.15. Seven different combinations of \( D \) and \( P \) are chosen to measure the accuracy of behavioral models. The maximum error of the models is 4.4\% of the dynamic range of \( V_C \) and this is sufficiently accurate to measure relative magnitude of different \( V_Cs \).

2.5.5.3 Error Behavior

The \( \Delta V_{BL}(D) \) is measured with different values of \( V_{WL} \) from HSPICE simulations as shown in Fig. 2.16. The \( D = 14 = 1110(2) \) is chosen to maximize \( \sigma_{\Delta V_{BL}(D)}/\mu_{\Delta V_{BL}(D)} \) as the \( BL \) is discharged by a single transistor without an averaging-out effect. The \( \sigma_{\Delta V_{BL}(D)} \) is also measured from Monte-Carlo simulations with HSPICE to estimate \( \sigma^2_{\Delta V_{BL}(D,P)} \) via (2.19) and (2.20), which are used for the system simulations to estimate the \( P_{DET} \) of DIMA. Energy efficiency of DIMA is improved by reducing \( V_{WL} \) due to low \( BL \) swing whereas the variation of \( FR \) is increased as shown in Fig. 2.16 as the access transistor operates in near-threshold voltage regime.
Figure 2.16: $\Delta V_{BL(D)}$ and $\sigma_{\Delta V_{BL(D)}}/\mu_{\Delta V_{BL(D)}}$ for $D = 14$ with $N = 4$.

Figure 2.17: Expected hardware error in (2.35) for $D = 128$ and $P = 240$. 
Figure 2.18: Face recognition application.

Figure 2.17 shows the expected power of error $E[\eta_C^2]$ at CBLP output in the DIMA with different values of $M$ and $V_{WL}$. The $\eta_B$ of (2.32) are obtained from HSPICE Monte-Carlo simulations with $D = 128$ and $P = 240$, which is one of the input combinations, where the random component of hardware error at $BLB$ is maximized. Then, the $E[\eta_C^2]$ are obtained through system simulations with (2.23) to observe the trend of error. The power of random error is inversely proportional to $N$ and $V_{WL}$ as indicated in (2.36) and Fig. 2.16. This trend shows that the DIMA is favorable to large sized templates, where error from process variation during the FR can be averaged-out better by (2.36), and thus more scaling down of $V_{WL}$ is available to maximize energy efficiency.

2.5.5.4 Application

The DIMA is employed for face recognition with distance metrics of MD as (2.41) to evaluate the impact of non-idealities from the DIMA on the application accuracy.

$$MD(D_u, P) = \sum_{j=1}^{N} |D_{u,j} - P_j|$$  \hspace{1cm} (2.41)
As a face recognition is a multi-class classification (128 classes in this application), it requires much higher accuracy than simple applications such as binary classification. Furthermore, face recognition requires higher resolution than applications which need to distinguish entirely different types of objects, e.g., car vs. humans. Therefore, the face recognition is chosen to evaluate the accuracy of DIMA conservatively.

One hundred twenty eight face images \((D)\) from MIT CBCL data set [53], each of which is a gray scale image with \(\sqrt{N} \times \sqrt{N}\) pixels and \(B = 8\), are pre-stored in the BCA per candidate (see Fig. 2.18). If \(N\) pixels cannot be accessed within one FR, the entire stages including FR, BLP, CBLP, and following ADC operations are iterated to process all the \(N\) pixels. Here, it is assumed that \(N\) is a multiple of \(N_{col}\), which is the number of columns in the BCA. An 8-bit precision single ramp ADC is employed to convert and store the value of \(V_C\) for sufficient resolution to recognize the templates. The ADC conversion can be processed in parallel with other DIMA stages. The resulting digital MD values are compared with a temporary minimum in a digital logic to update the temporary minimum and its pointer address \((u)\) in (2.41). The entire process terminates when 128 such updates are completed.

2.5.5.5 System Performance \((P_{DET})\)

System simulations are performed with the Monte-Carlo method to measure the detection probability \(P_{DET}\) of template using the behavioral models in (2.17)-(2.23) with parameters from Table 2.2. Eight face image samples from the set of 128 \(D\) images are randomly chosen as the template \(P\), and the \(P_{DET}\) per template \(P\) is obtained by counting the number of correct detections out of multiple trials. The overall \(P_{DET}\) is obtained by averaging the 8 \(P\)-specific \(P_{DET}\) values. The \(P_{DET}\) is also measured with \(N = 64\) and 16, which are achieved by sub-sampling the images \(D\) and \(P\) with \(N = 256\).

The conventional system with \(\Delta V_{BL(B)} = 250\, mV\) and \(V_{WL} = 1\) achieves \(P_{DET} = 1\) with \(N = 16, 64, \) and 256. The system simulations with the Monte-Carlo method shows that the DIMA achieves \(P_{DET} = 1\) with \(V_{WL} \geq 600\, mV\) and \(N = 256\), and achieves higher \(P_{DET}\) with higher \(N\) and \(V_{WL}\) as shown in Fig. 2.19. It can be concluded that the non-idealities from the DIMA is effectively compensated by the inherent error resiliency of the MD algorithm.
in this condition.

The predicted $P_{DET}$ by (2.40) is slightly higher than that from Monte-Carlo simulations as shown in Fig. 2.19 because the effects from non-linearity of FR and the comparator offset are excluded in this prediction. The predicted $P_{DET}$ shows the highest accuracy with the largest $L = 256$, where the maximum error magnitude of the predicted $P_{DET}$ is 0.011. This is because $\eta_C$ approaches a Gaussian distribution by the central limit theorem as $N$ increases [50]. The accuracy of prediction decreases with a smaller value of $N$, and the maximum error magnitude of the predicted $P_{DET}$ is 0.066 and 0.067 with $N = 16$ and 64, respectively.

2.6 Conclusion

This chapter describes the DIMA’s four processing stages: (1) FR, (2) BLP, (3) CBLP, and (4) thresholding, based on the common functional flow of ML algorithms. DIMA’s robustness in the low-SNR regime was demonstrated with design principles and measurement results of prototype IC. The design guidelines and techniques provided in this chapter are applied
to two prototype ICs, which will be introduced in the following chapters. This chapter also proposed energy, delay, and behavioral models of non-ideal behavior from analog circuitry. These models will be employed to estimate an application level’s accuracy and energy and delay benefits in various algorithms/applications in the rest of this dissertation.
Chapter 3

DIMA PROTOTYPE INTEGRATED CIRCUITS

Chapter 2 demonstrated DIMA’s versatility by enabling various VD computations and demonstrating significant energy and delay benefits in simulations. This chapter realizes the DIMA concept as two prototype ICs: (1) multi-functional DIMA [39], and (2) random forest (RF) DIMA [54] in a 65 nm process. This chapter begins with the description of multi-functional DIMA, which supports four different algorithms: support vector machine (SVM), template matching (TM), k-nearest neighbor (k-NN), and matched filter (MF). Design details including chip architecture, circuit techniques, and measured results are provided in the following sections. Then, those design principles are extended to enable the RF algorithm, which is an ensemble of many decision trees. The DIMA prototype IC for RF is described in the last section of this chapter. To the best of our knowledge, this is the first IC realization of the RF algorithm.

3.1 Multi-Functional DIMA Architecture

The following three sections present a multi-functional DIMA in a 65 nm CMOS process. The chip architecture (Fig. 3.1) comprises a DIMA core (CORE), a digital controller (CTRL), and an input register to stream in the operand \( P \). The CORE includes a 512\( \times \)256 BCA, the conventional SRAM read/write circuitry, the BLP and CBLP, and four 8-bit single-slope ADCs. The RDL is embedded in the digital CTRL.

The SRAM bitcell was custom-designed following standard design rules as the memory compiler did not allow modifications to be made to the peripheral circuitry. As a result,
the horizontal and vertical dimensions of bitcell were approximately $1.7 \times$ larger typical foundry-provided bitcells dimensions [55]. The column muxing ratio was chosen to be $L = 4$ to maximize the throughput for the standard SRAM read. Thus, the SA (and write driver) is shared by four columns.

An 8-bit precision is chosen for $D$ and $P$ in order to maintain almost the same accuracy as floating point [5, 56, 57]. Based on the design principles in Section 2.3.1, parameter values $V_{wl} = 0.65 \text{ V}$ and $T_0 \approx 250 \text{ ps}$ were chosen resulting in the longest PWM-WL pulse width $T_3 < 0.4 R_{BL} C_{BL}$ was achieved thereby ensuring sufficient linearity and avoiding destructive read. Sub-ranged FR was employed to access 4 MSBs and 4 LSBs simultaneously from adjacent column pairs. As mentioned earlier, tuning capacitors are attached to the BLs to realize the 16:1 capacitance ratio for the column pair. The area overhead due to DIMA circuitry in the CORE was found to be 25%.

The serially provided reconfiguration word $RCFG$ initializes the local controllers in the CTRL. Four slow but energy-efficient 8-bit single-slope ADCs [58] are employed to convert the analog CBLP outputs in parallel.
3.2 Multi-Functional DIMA Operations

3.2.1 Timing

The chip operations are sequenced via the CTRL which operates with a master 1 GHz CLK thereby providing a 1 ns time resolution for generating various control signals synchronized to CLK. Self-timed control [59, 60] can improve the throughput of both normal read and DIMA operations but we chose synchronous design for simplicity.

The timing diagram in Fig. 3.2(a) describes the series of ten events that occur during a word-row period, i.e., when processing a single word-row of \( B = 8 \) bits through the FR, BLP, and CBLP stages. The first event in both MD and DP modes is the BL precharge. Next, the FR, BLP, and CBLP stages are sequentially executed to generate the corresponding outputs \( V_{BL} \), \( V_B \), and \( V_C \), respectively. One difference between the two modes - the MD mode requires transferring \( P \) from the input buffer into replica BCA before initiating FR. On the other hand, the DP mode needs to make this transfer of \( P \) to the mixed-signal multiplier before initiating BLP and requires additional delay in the CBLP stage to support sub-ranged processing. The last event samples the CBLP output to generate the input voltage for the ADC. Each event requires an integer number of CLK cycles which are estimated via post-layout simulations. The prototype IC provides for tunability to allow additional CLK cycles to be introduced for each stage in order to accommodate deviations from the nominal process corner.

Figure 3.2(b) shows timing diagram for processing 256-dimensional \( D \) and \( P \) vectors. Each word-row consists of 128 8-b words (though \( N_{col} = 256 \), the use of sub-ranged read results in a 128 dimensional vector) generate the CBLP output \( V_C \). Two word-rows are processed consecutively and their CBLP outputs \( V_C's \) are sampled and charge-shared (Merge_SP step) to aggregate 256 scalar elements of the 256-dimensional vectors \( D \) and \( P \). This is followed by using an available 1-of-4 ADCs to digitize the analog CBLP output into an 8-b word. The single-slope ADC conversion takes 140 CLK cycles for both MD and DP modes, which is approximately 5.6 MD word-row periods. However, this slow conversion rate is not an issue as the ADCs operate in parallel. The ADC output is further processed in the RDL
Figure 3.2: DIMA timing diagrams for processing: (a) a single word-row, and (b) multiple word-rows (dotted red line: single thread to process 256 words).

block to realize the thresholding operation (Threshold\_EN step).

3.2.2 Algorithm and Application Mapping

The four tasks (face detection using SVM, gun shot detection using MF, face recognition using TM, and handwritten digit recognition using k-NN) (see Fig. 3.3) were mapped on to the prototype IC. These tasks cover both binary and multi-class (4-class and 64-class) scenarios, requiring both MD and DP modes of operations, and processing of both image and sound data sets [53, 61, 62] as summarized in Table 3.1. Table 3.2 defines the set of operations per stage of CORE that can be chosen using the RCFG word during one word-row period. In this process, the prototype IC is able to realize the four different algorithms as shown in Table 3.3.
The MF (Fig. 3.3(b)) creates the decision right after single DP processing and thresholding. On the other hand, SVM requires signed coefficients. Thus, the absolute values of positive and negative coefficients are stored in the separate rows as shown in Fig. 3.3(a). The positive and negative products are computed in consecutive cycles, and then compared in the RDL stage to obtain the sign of the DP. TM (Fig. 3.3(c)) and $k$-NN (Fig. 3.3(d)) make decisions after comparison (to find minimum) or majority voting across multiple candidates. All the data sets are processed fully on-chip except for $k$-NN, where the last step of majority voting was done off-chip.

3.2.3 Design Techniques for Re-configurability

Enabling the computation of multiple functions via mixed-signal circuitry with stringent column pitch-matching constraints is very challenging as shown in Fig. 2.10. Two design
Table 3.1: Measured data set for four applications [53, 61, 62].

<table>
<thead>
<tr>
<th>Task</th>
<th># of classes</th>
<th>Algorithm</th>
<th>Data set</th>
<th>Remarks (P: query input, D: data stored in array)</th>
</tr>
</thead>
</table>
| Face detection              | 2            | SVM       | MIT CBCL data set   | - 100 query inputs tested  
- D: feature extractor and classifier combined 23 X 22 8-b coefficient  
- P: 23 X 22 8-b pixel image (face / non-face)                                                                 |
| Event (gun shot) detection | 2            | MF        | Gun shot Sound      | - 100 query inputs tested  
- D: gun shot mono sound data with 256 8-b words  
- P1: gun shot sound contaminated by AWGN with 3 dB SNR  
- P2: Only AWGN with equal power of "signal + AWGN" in P1                                                                 |
| Face recognition            | 64           | TM        | MIT CBCL data set   | - 64 query inputs tested (due to array size limit)  
- 16 X 16 8-b pixel image for P and D  
- D: 64 candidate faces  
- P: one of the 64 candidate faces in D                                                                 |
| Hand-written number         | 4            | k-NN      | MNIST data set      | - 100 query inputs tested  
- 16 X 16 8-b pixel image for P and D  
- 4 classes from "0" to "3" (due to array size limit)  
- D: 16 images per class, - P: image from 4 classes                                                                 |

Table 3.2: Multi-functions in each processing stage.

<table>
<thead>
<tr>
<th>Stage</th>
<th>configurations</th>
</tr>
</thead>
</table>
| FR    | ① Normal read  
② Digital to analog conversion  
③ Scalar ADD or SUBT |
| BLP   | ① Scalar MULT  
② Bl-wise sampling  
③ Absolute value |
| CBLP  | ① Aggregation  
② Weighted aggregation |
| RDL   | ① MIN or MAX  
② Linear combination  
③ Send outside chip |

53
Table 3.3: Configurations of each stage to enable four algorithms (the operations corresponding to numbers are described in Table 3.2).

<table>
<thead>
<tr>
<th>Mode</th>
<th>Algorithm</th>
<th>FR</th>
<th>BLP</th>
<th>CBLP</th>
<th>RDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>DP</td>
<td>SVM</td>
<td>②</td>
<td>①, ②</td>
<td>②</td>
<td>②</td>
</tr>
<tr>
<td></td>
<td>MF</td>
<td>②</td>
<td>①, ②</td>
<td>②</td>
<td>②</td>
</tr>
<tr>
<td>MD</td>
<td>k-NN</td>
<td>②, ③</td>
<td>②, ③</td>
<td>①</td>
<td>③</td>
</tr>
<tr>
<td></td>
<td>TM</td>
<td>②, ③</td>
<td>②, ③</td>
<td>①</td>
<td>①</td>
</tr>
</tbody>
</table>

techniques are presented in order to support MD and DP modes in the BLP.

- Circuitry sharing with reconfiguration: Figure 3.4(a) shows the BLP (and CBLP) circuit implementation to support the DP and MD modes. In the MD mode, an analog comparator and mux implements the $\max$ operation in (2.14) to compute the absolute value $|D - P|$. In the DP mode, the comparator is bypassed and mux always chooses BLB. In both modes, the capacitor $C$ in the red box of BLP multiplier (see Fig. 3.4(b)) is shared with the CBLP to realize the $BL$-wise sampling capacitor $C_S$ in Fig. 2.12.

- Sub-ranged processing: the charge-based multiplier in Fig. 2.11(b) employs unit capacitors to meet the column pitch constraints necessitating sequential processing of multiplicand bits ($p_i$) and thereby limiting the throughput. Sub-ranged multiplication alleviates this problem by employing two 4-b MSB and LSB multipliers operating in parallel (Fig. 3.4(a)) and dump their $V_{fs}$ on the $MSB\_Rail$ and $LSB\_Rail$, respectively, while $\phi_1 = 1$. Then, the switch $\phi_1$ opens to make the capacitance of the $MSB\_Rail$ 16× larger than that of the $LSB\_Rail$. Subsequent charge sharing of the the $MSB\_Rail$ and $LSB\_Rail$ is done by setting $\phi_1 = 0$ and $\phi_2 = 1$ to generate the final output $V_C$. In this manner, the sub-ranged
Figure 3.4: BLP and CBLP implementations for reconfiguration: (a) overall structure, and (b) re-configurable charge-based multiplier for 4-b MSB (or LSB) and its enabling signals (only red marked area is used in MD mode).
processing improves the throughput of DP mode by a factor of two.

3.3 Measured Results of Multi-Functional DIMA IC

This section shows measured results of the prototype IC, packaged in an 88-pin QFN as shown in Fig. 3.5 and summarized in Table 3.4.

3.3.1 Accuracy of FR

The measured results of sub-ranged FR of 8-b word $D$ is shown in Fig. 3.6(a). The $BL$ voltage drop $\Delta V_{BL}$ generated by FR for all 256 values of $D$ was measured at the output of the column mux along the normal SRAM read path. The integral non-linearity (INL) was found to be less than 0.87 LSB. The sudden jump when $D$ transitions from 7 to 8 is due to the large change in the average transition time of the $WL$ pulses. The value of worst-case $\gamma_i$ ($\gamma_0$) was estimated by comparing the slope of the curve between $D = 0$ and $D = 1$, and between $D = 0$ and $D = 8$. Similarly, the value of worst-case $\rho_i(V_{BL})$ ($\rho_3(0.5 \text{V})$) was
Table 3.4: Multi-functional DIMA prototype IC summary.

<table>
<thead>
<tr>
<th>Technology</th>
<th>65 nm CMOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Die size</td>
<td>1.2 mm × 1.2 mm</td>
</tr>
<tr>
<td>CTRL operating freq.</td>
<td>1 GHz</td>
</tr>
<tr>
<td>SRAM capacity</td>
<td>16 KB (512 × 256-b)</td>
</tr>
<tr>
<td>Bitcell dimension</td>
<td>2.11 × 0.92 um²</td>
</tr>
<tr>
<td>Supply voltage</td>
<td>CORE: 1.0 V, CTRL: 0.85 V</td>
</tr>
</tbody>
</table>

estimated by comparing the slope of the curve between $D = 6$ and $D = 7$, and between $D = 14$ and $D = 15$. These worst-case values of $\gamma_i$ and $\rho_i$ in (2.9) were found to be less than 41% and 37%, respectively.

The variation in $\Delta V_{BL}$ due to $\delta_i$ was measured via following 4-step process: (1) store the same data across the entire BCA, (2) access the first word-row via with FR, (3) the $\Delta V_{BL}$s are aggregated via the CBLP and sampled to generate output voltage $\Delta V_C = V_{PRE} - V_C$, and (4) repeat this process for all 128 word-rows. The data $D_M = D_L = 7$ is chosen to generate the worst-case (maximum) variation in $\Delta V_{BL}$, as in this case, the $BL$ is discharged by a single SRAM bitcell. Fig. 3.6(b) shows that the variation in $\Delta V_{BL}$ has a standard deviation $\sigma = 2.5$ mV, which is only 0.6% of the dynamic range (410 mV) of CBLP output $V_C$ in this test mode. This small deviation arises because of the aggregation effect of CBLP as described in Section 2.2.3. In Section 3.3.3, we show that these errors result in negligible impact on the accuracy of inference tasks considered in this paper.

3.3.2 Accuracy of CORE Output

We characterize the accuracy of the CORE output that includes FR, BLP and CBLP stages. This is because it is difficult to isolate the BLP and CBLP outputs from each other. The
Figure 3.6: Measured FR accuracy of 8-b $D$: (a) sub-ranged read, and (b) impact of spatial variation on $\Delta V_C$. 
CORE output shown in Fig. 3.7 was measured for the same data \((D \text{ and } P)\) being stored in all the columns, and for the DP and MD mode computations. The measured error magnitudes at \(V_C\) (from ideal linear trend) in the DP and MD modes are < 18 mV and < 28 mV with the mean of 4 mV and 8 mV, respectively, over all the combinations of \((D, P)\). Though these errors are significantly larger than the chosen target resolution \(V_{res} = 1\) mV, these errors are easily masked by choosing the hyper-parameters of the inference task to provide a sufficiently large decision margin as will be discussed next. In addition, it is possible to training the engine in presence of these errors to obtain circuit-optimized hyper-parameters.

3.3.3 Energy, Delay, and Accuracy

We consider the energy consumption of the CORE block only because its energy scales up with the number banks and the BCA size. In contrast, the energy of the CTRL block is amortized over the number of banks and the BCA size. We measured the CORE decision energy and decision accuracy for SVM (face detection, binary class) and TM (face recognition, 64-class) tasks. The CORE decision energy was normalized by the number of 8-b data words processed per decision to obtain the energy-per-word as a function of \(BL\) swing per
Figure 3.8: Measured BL swing ($\Delta V_{lsb}$) vs. energy vs. probability of correct detection ($P_{det}$) trends: (a) $\Delta V_{lsb}$ vs. CORE energy, (b) $\Delta V_{lsb}$ vs. $P_{det}$, and (c) CORE energy vs. $P_{det}$.

LSB $\Delta V_{lsb} = \Delta V_{BL}(D_M = 15)/15$ in Fig. 3.6(a).

Figure 3.8(a) indicates that CORE energy reduces at a rate of 0.2 pJ (0.4 pJ) per 20 mV for binary or DP mode (64-class or MD mode) task. Furthermore, the greater slope of the energy vs. $\Delta V_{lsb}$ plot for the MD mode and its higher energy consumption for $\Delta V_{lsb} > 15$ mV is because the MD mode uses the replica BCA which causes additional voltage drop on the BL during FR.

The accuracy of the inference task is measured by the probability of detection obtained by normalizing the number of queries correctly classified by the total number of queries. Figure 3.8(b) shows that the binary task is more robust than the 64-class task at the same $\Delta V_{lsb}$. Furthermore, the binary and the 64-class task achieve $> 90\%$ detection accuracy for $\Delta V_{lsb} > 15$ mV and $\Delta V_{lsb} > 25$ mV, respectively. Figure 3.8(c) plots the probability of detection against the CORE energy per 8-b pixel and shows that accuracy and energy trade-off with each other.

Next, we compare the DIMA prototype with a conventional 8-b digital reference architecture (REF). REF is a 2-stage pipelined comprising an SRAM of the same size as the one in the DIMA prototype, and a digital block synthesized separately for realizing an SVM (DP mode) and a TM (MD mode). The energy and delay of the digital block in REF was
Figure 3.9: Energy comparison of DIMA with the reference architecture (REF). DIMA* represents the energy obtained from post-layout simulations while DIMA shows the measured values.

estimated from post-layout simulations. The energy and delay of the SRAM in REF was measured from the DIMA prototype in the normal read mode. Figure 3.9 shows the energy breakdown for REF, DIMA* (post layout simulations of the DIMA prototype IC, and the DIMA prototype IC. The measured energy savings in the DP and MD modes are 10× and 5.5×, respectively, due to small swing FR, BLP, and CBLP. Furthermore, we find that the DIMA energy estimates obtained from post layout simulations are close to that obtained from measurements.

Table 3.5 shows that the DIMA prototype IC achieves negligible (≤1%) accuracy degradation for all four tasks as compared to REF. DIMA requires 16× fewer read accesses as compared to REF for a fixed data volume, resulting in up to 5.8× throughput enhancement. This is because FR and BLP process data in massively parallel manner (128 8-b words per access) whereas the normal SRAM mode fetches only 8 8-b words through 4:1 column muxing. Smaller effective $\Delta V_{BL}$ and fewer read access reduces data access energy. The low-swing computations in the BLP and CBLP stages add to the energy savings. The DIMA prototype IC implements four different algorithms achieving better decision accuracy and comparable energy-delay product (scaled for 65 nm) than single function ICs [5,41] listed in Table 3.5.
Table 3.5: Application level gains of multi-functional DIMA in energy efficiency, delay, accuracy, and comparison with prior arts.

<table>
<thead>
<tr>
<th>Process (nm)</th>
<th># of algorithms</th>
<th>Memory size</th>
<th>Input bit precision</th>
<th>Decision throughput (Decisions/s)</th>
<th>Decision energy (pJ/decision)</th>
<th>Decision ED (fJ-s)</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>This work</td>
<td>65 CMOS</td>
<td>4 (SVM, MF, k-NN, TM)</td>
<td>SRAM 512 x 256-b</td>
<td>D: 8b, P: 8b</td>
<td>SVM: 9.3M</td>
<td>446 (963.1)†</td>
<td>0.05 (0.1)†</td>
</tr>
<tr>
<td>8 b digital* (REF)</td>
<td>1 (k-NN)</td>
<td>synthesized dedicated processor per algorithm</td>
<td>SRAM 512 x 256-b</td>
<td>D: 8b, P: 8b</td>
<td>SVM: 1.7M</td>
<td>4.5K</td>
<td>2.6</td>
</tr>
<tr>
<td>[5]**</td>
<td>14 Tri-gate</td>
<td>1 (k-NN)</td>
<td>128 byte</td>
<td>D: 8b, P: 8b</td>
<td>21.5M</td>
<td>3.4K</td>
<td>0.2</td>
</tr>
<tr>
<td>[41]**</td>
<td>130 CMOS</td>
<td>1 (AdaBoost)</td>
<td>SRAM 128 x 128-b</td>
<td>D: 1b, P: 5b</td>
<td>50M</td>
<td>633.4</td>
<td>0.01</td>
</tr>
</tbody>
</table>

* memory (digital) energy and delay measured from prototype IC (post-layout simulations);
† assumes a 32 bank configuration;
†† single function with SRAM memory access cost not included;
** single function with 1b weight vector

3.4 Random Forest (RF) DIMA IC\(^1\)

This section presents IC realization of a random forest (RF) ML classifier based on the DIMA platform [54]. The RF classifier [63] is attractive due to its high-accuracy, simple operations (comparisons), applicability to multi-class problems, and robustness to non-ideal computations due to its majority voting based-decision. However, realizing an energy-efficient implementation of the RF algorithm is challenging due to its high data access rate combined with its highly irregular data access pattern. This section presents an energy-efficient and high throughput RF classifier IC by employing: (1) deterministic subsampling (DSS) to reduce interconnect complexity, (2) a balanced decision tree to regularize memory access pattern, (3) deeply embedded analog computations [39] in the periphery of an SRAM bitcell array (BCA) to exploit the inherent algorithmic error tolerance. To the best of our knowl-

\(^1\)This section is adopted from M. Kang, S. Gonugondla, and N. R. Shanbhag, “A 19.4 nJ/decision 364 K decisions/s in-memory random forest classifier in 6T SRAM array,” in 47th IEEE European Solid-State Circuits Conference (ESSCIRC). © 2017 IEEE
Table 3.6: Number of required operations in proposed/conventional RF.

<table>
<thead>
<tr>
<th>OPs per tree</th>
<th>Memory accesses</th>
<th>Comp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>( p_{m,n} )</td>
<td>( c_{m,l} )</td>
</tr>
<tr>
<td>Bit precision</td>
<td>6/8</td>
<td>4/4</td>
</tr>
<tr>
<td>Size (Byte)</td>
<td>21/31</td>
<td>0.5/16</td>
</tr>
<tr>
<td># of OPs†</td>
<td>3/4</td>
<td>0.5/2</td>
</tr>
<tr>
<td>Cross bar</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- proposed / conventional

\( \tau_{(m,n)} \): threshold level of \( n \)-th node in \( m \)-th tree

\( p_{(m,n)} \): pixel index of \( n \)-th node in \( m \)-th tree

\( P_m \): \([p_{m,1}, p_{m,2}, \ldots, p_{m,N}]\)

RSS: Random subsampling by sample pattern \( P_m \)

\( x(p_{(m,n)}) \): \( p_{(m,n)} \)-th pixel of input image \( X \)

\( c_{(m,l)} \): label corresponding to \( l \)-th leaf node in \( m \)-th tree \((m: 1 \sim M, \ n: 1 \sim N, \ l: 1 \sim N+1)\)

edge, this is the first IC implementation of the RF algorithm as there are the only FPGAs, GPUs, and multi-core processor implementations of the RF algorithm [63]. These fail to take advantage of the opportunities afforded by analog computations.

### 3.4.1 Background

This section explains the RF algorithm and its implementation challenges.

#### 3.4.1.1 RF Algorithm

The RF algorithm (Fig. 3.10(a)) consists of \( M \) decision trees. The \( m \)-th tree processes data obtained by random subsampling (RSS) the input image \((X)\) using a pseudo-random pattern vector \( P_m \). The \( n \)-th node in the \( m \)-th tree compares \( x(p_{(m,n)}) \), which is the pixel (or feature)
indexed by \( p_{(m,n)} \), with a threshold \( \tau_{(m,n)} \) to obtain a node-level binary decision \( q_{(m,n)} \). Either the left or right branch is taken based on \( q_{(m,n)} \). This process is repeated until a leaf node is reached. The label \( c_{(m,l)} \) corresponding to the \( l \)-th leaf node is the tree-level decision. The final decision is obtained by majority-voting the \( M \) tree-level decisions.

### 3.4.1.2 Implementation Challenges

Two different architectures can be considered to implement the RF algorithm: serial and parallel architectures. A serial architecture needs to process nodes sequentially resulting in large delay and requires reading of two 11-b (for a 16 KB array) child node addresses per node, which takes roughly half of the storage space. On the other hand, a fully parallel architecture computes all \( q_{(m,n)} \) in parallel and uses these to address a look-up table (LUT) to obtain \( c_{(m,l)} \). Doing so requires a large number of memory accesses, e.g., seventy eight 8-b bytes per tree (Table 3.6), which in turn limits the achievable throughput and energy efficiency. Additionally, a complex (i.e., 256:1 with 16×16 image \( X \)) crossbar is needed to route the pixel indexed by \( p_{(m,n)} \) from \( X \) for comparison.

### 3.4.2 The Proposed RF Algorithm and Architecture

This section co-optimizes the algorithm and architecture to achieve energy and throughput benefits.

#### 3.4.2.1 The Proposed RF Algorithm

The modified RF algorithm (Fig. 3.10(b)) employs a fixed-pattern deterministic subsampling (DSS) step prior to RSS to solve the crossbar problem mentioned above. A 4:1 DSS factor
is chosen to balance the loss in classification accuracy by reducing the crossbar complexity. The complexity of the RSS crossbar is reduced from 256:1 to 64:1 when the input $X$ is a $16 \times 16$ image. Thus, the precision of $p_{(m,n)}$ is also reduced from 8-b to 6-b. Additionally, the decision trees are balanced (Fig. 3.10(b)) by filling some empty nodes in order to regularize the memory access pattern. The memory access problem is addressed by reducing the number of memory accesses via in-memory comparison (Fig. 3.11) eliminating the need to fetch $\tau_{(m,n)}$. The Class ADD generator (CAG) generates the address of chosen $c_{(m,l)}$ from $q_{(m,n)}$ eliminating the need to fetch all the $c_{(m,l)}$s. Only 24.5 bytes of data need to be fetched per tree compared to 78 bytes/tree in the parallel architecture.
Figure 3.11: In-memory comparison: bitcell column for in-memory comparison of $T$ and $X$, and measured accuracy of comparison.

3.4.2.2 Proposed Architecture and Operations

The proposed RF architecture (Fig. 3.12(a)) includes a $512 \times 256$ SRAM BCA, multi-row WL driver, 64-b I/O with a 4:1 column mux, DSS input buffer to store streamed $X$, RSS crossbars, CAG, label finder, majority voter, and the peripherals for standard read/write operations. A group of four trees are processed in parallel and 16 such groups are processed sequentially for a total of $M = 64$ trees. The classifier first: (1) writes the pixel index register, (2) enables the crossbar, (3) does in-memory comparison enabled by the multi-row WL driver and analog comparators, (4) sequentially fetches four tree-level labels using address generated by CAG, and (5) votes by majority in the final tree.
3.4.2.3 In-Memory Comparison

In-memory comparison requires the 8-b thresholds $T_{(m,n)}$ ($T$ in Fig. 3.11) and the indexed pixels $x(p_{(m,n)})$ ($X$ in Fig. 3.11) to be stored in a column major pattern, i.e., bits of a word are stored in a column. The comparison begins with the simultaneous application of WL access pulses with binary-weighted pulse widths to all the rows storing $T$ and $X$. Here, the pulse width is proportional to the bit position. Doing so creates a bitline (BL) voltage swing $\Delta V_{BLB}$ ($\Delta V_{BL}$) proportional to $T - X$ ($X - T$) [35, 39]. Linearity of this 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 $16 \times$ greater weight to the MSBs. The WL voltage is reduced (e.g., 0.65 V) to prevent destructive read and improve the linearity further. Storing the $X$ in the replica bitcell array allows fast writing through a separate write

![Figure 3.12: Proposed RF: (a) architecture, and (b) timing diagram.](image-url)
**Figure 3.13: RF DIMA die micrograph.**

**BL (WBL)** and wordline (WWL) by eliminating the overheads of slow write operation into normal BCA. The **BLs** feed into analog comparators to generate node-level decisions \( q \). In-memory comparison is an intrinsically and massively parallel operation as it processes all 128 8-b words in parallel from 256 columns whereas conventional memory fetches only 64 bits (= 8 words) per read access when the sense amplifier is shared across four columns. In addition, multi-row read saves energy by accessing 4 bits per precharge.

### 3.4.3 Chip Measured Results

The in-memory RF classifier is implemented in a 65 nm CMOS process (chip micrograph in Fig. 3.13 and summarized in Table 3.7) to prove the application-level’s benefits.
Table 3.7: RF prototype IC summary.

<table>
<thead>
<tr>
<th>Technology</th>
<th>65 nm CMOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Die size</td>
<td>1.2 x 1.2 mm</td>
</tr>
<tr>
<td>SRAM capacity</td>
<td>16 KB (512 x 256 bitcells)</td>
</tr>
<tr>
<td>Bitcell size</td>
<td>2.11 x 0.92 um²</td>
</tr>
<tr>
<td>CTRL CLK freq.</td>
<td>1 GHz</td>
</tr>
<tr>
<td>Supply voltage (V)</td>
<td></td>
</tr>
<tr>
<td>CORE</td>
<td>1.0</td>
</tr>
<tr>
<td>CTRL</td>
<td>0.75</td>
</tr>
<tr>
<td>Energy per decision (4 trees, 64 trees)(nJ)</td>
<td>CORE: (0.9, 14.4)</td>
</tr>
<tr>
<td></td>
<td>CTRL: (0.3, 5.0)</td>
</tr>
<tr>
<td>Decision throughput (decisions/s) (4 trees, 64 trees)</td>
<td>(5.6M, 364k)</td>
</tr>
</tbody>
</table>

Figure 3.14: Energy vs. error rate w.r.t $\Delta V_{BL}$ with 64 trees (eight-class traffic sign recognition).
3.4.3.1 Component-Level Accuracy Characterization

Measured in-memory comparison results show (Fig. 3.11(b)) the comparator error rate increasing from 1.6% to 14.5% as $\Delta V_{BL}$ reduces from 25 mV to 5 mV. The RF algorithm with 64 trees needs an error rate of less than 9.5% at comparator output $q$ to avoid a discernable eight-class classification accuracy loss. Four trees tolerate only 4% error thereby restricting further reduction in $\Delta V_{BL}$.

3.4.3.2 Application-Level Accuracy, Energy, and Throughput

Measured results (Fig. 3.14) of energy vs. accuracy trade-off for the eight-class traffic sign recognition with 64 trees show the proposed IC achieves a $3.1 \times$ energy savings over the conventional architecture (SRAM + digital processor). The energy of the conventional architecture is obtained via post-layout simulations of the digital blocks and read access energy measured from the prototype IC. This energy savings come from multi-row read, in-memory comparison, and low-complexity crossbar. Fewer memory accesses also reduce the decision delay by $2.2 \times$ over a conventional architecture, thereby providing a $6.8 \times$ lower energy-delay product (EDP) at the same accuracy of $> 93\%$ as the conventional architecture. The prototype IC achieves a throughput of 364 K decisions/s and energy efficiency of 19.4 nJ/decision, achieving at least $5.6 \times$ smaller EDP compared to prior multi-class classifier ICs [5, 64] as listed in Table 3.8.

3.5 Conclusion

This chapter describes two DIMA prototype ICs: (1) multi-functional DIMA for SVM, TM, $k$-NN, and MF, and (2) single-function DIMA for RF. Potentially, more algorithms can be
Table 3.8: Application level gains of RF DIMA in energy efficiency, delay, accuracy, and comparison with prior arts.

<table>
<thead>
<tr>
<th>Prior art</th>
<th>Process</th>
<th>Algorithm</th>
<th>Dataset</th>
<th>Input Size (8b)</th>
<th>Throughput (decision/s)</th>
<th>Energy (nJ/decision)</th>
<th>EDP (fJ/decision)</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>[64]</td>
<td>130nm CMOS</td>
<td>Support Vector Machine</td>
<td>Traffic sign video</td>
<td>320\times240</td>
<td>33\ ([40K]*</td>
<td>1.5M [1250]*</td>
<td>45G [31250]*</td>
<td>90%</td>
</tr>
<tr>
<td>[5]*</td>
<td>14nm tri-gate</td>
<td>K-nearest Neighbor</td>
<td>Not reported</td>
<td>128</td>
<td>21.5M [498.8K]*</td>
<td>3.4 [145.3]*</td>
<td>0.2 [292.3]*</td>
<td>Not reported</td>
</tr>
<tr>
<td>Ours (M=64)</td>
<td>65nm CMOS</td>
<td>Random Forest</td>
<td>KUL traffic signs</td>
<td>16\times16</td>
<td>364.4K</td>
<td>19.4 (w/ CTRL)</td>
<td>52.4</td>
<td>94%</td>
</tr>
</tbody>
</table>

* memory (digital) energy and delay measured from prototype IC (post-layout simulations) 
† assumes a 32 bank configuration 
†† single function with SRAM memory access cost not included 
** single function with 1b weight vector

covered by simply modifying or adding functionality in each processing stage. Measurement results of multi-functional DIMA IC demonstrate up to 31\times EDP reduction as compared to the conventional digital architecture optimally designed for each algorithm. Furthermore, the EDP benefit is expected to be even higher (up to 56\times) in multi-bank scenarios by sharing the controller overhead over many banks. The prototype IC of RF also achieves a 3.1\times energy savings and 2.2\times speed-up at the same time providing a 6.8\times lower EDP at the same accuracy of > 93% compared to conventional digital architecture, leading to a throughput of 364 K decisions/s and energy efficiency of 19.4 nJ/decision for eight-class traffic sign recognition problem. A trade-off between the energy and accuracy was also observed in both ICs by controlling the $\Delta V_{BL}$. This indicates that there is a potential to push DIMA's energy savings further by statistical error compensation techniques. Specifically, the ML coefficients can be re-trained to compensate the deterministic error patterns of non-ideal analog circuitry. On the other hand, the DIMA with an on-chip trainer will be able to compensate time-dependent noise sources. In particular, the feasibility of multi-functional DIMA indicates the potential to realize programmable DIMA instruction set architecture (ISA), which will be explored in Chapter 5.
Chapter 4

MAPPING INFEERENCE ALGORITHMS TO DIMA

In Chapter 3, it was shown that the multi-functional DIMA enables four algorithms with similar functional flow. In this chapter, the DIMA is applied to two algorithms with more complex functional flow as follows: (1) convolutional neural network (CNN) [36] - deep neural network based on convolutional operations, and (2) sparse distributed memory (SDM) [37,38] - a computational model inspired by the human brain. The DIMA-based CNN demonstrates that the error-aware training can compensate the non-ideal behavior of DIMA effectively. On the other hand, the SDM generates the final decision via majority-voting across decisions from many weak classifiers. This ensemble nature makes the system more robust to the hardware noise allowing further low-SNR processing to achieve aggressive energy savings. In addition, the SDM algorithm is modified to maximize the benefit from DIMA-based architecture.

4.1 Convolutional Neural Network (CNN)\(^1\)

Convolutional neural networks (CNN) is one of the most widely used pattern recognition algorithms due to its state-of-the-art performance in computer vision applications such as handwriting recognition and face detection [65, 66]. However, the CNN requires complex interconnect, massive inner product computations, and access to a large data volume. GPU [65] and FPGA-based [66] implementations were proposed recently in order to speed up CNN computation over a purely software implementation. It is well known that the energy and throughput of general-purpose computing platforms such as GPU and FPGA are at least

---

\(^1\)This section is adopted from M. Kang, S. Gonugondla, M.-S. Keel and N. R. Shanbhag, “An energy-efficient memory-based high-throughput VLSI architecture for Convolutional Networks,” in 40th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). © 2015 IEEE
one to two orders-of-magnitude worse than dedicated VLSI implementations [35].

This section presents a dedicated VLSI architecture based on DIMA, where computation is embedded inside the memory array. Our DIMA-based CNN implementation [36] is shown to provide a $24.5 \times$ reduced EDP as compared to the conventional system.

### 4.1.1 Background: CNN

A CNN is a multi-layer network (see Fig. 4.1(a)) consisting of interleaved convolutional layers (C-layers) and sub-sampling layers (S-layers). The C-layer is computationally intensive and is described as follows:

$$
\begin{bmatrix}
  y_1 \\
  \vdots \\
  y_N
\end{bmatrix} = \phi \left( \begin{bmatrix}
  w_{11} & \cdots & w_{1M} \\
  \vdots & \ddots & \vdots \\
  w_{N1} & \cdots & w_{NM}
\end{bmatrix} \times \begin{bmatrix}
  x_1 \\
  \vdots \\
  x_M
\end{bmatrix} + \begin{bmatrix}
  b_1 \\
  \vdots \\
  b_N
\end{bmatrix} \right)
$$

where $x_m$ ($m = 1, \ldots, M$) and $y_n$ ($n = 1, \ldots, N$) are the $L \times L$ input and $(L - K+1) \times (L - K+1)$ output feature maps, respectively, $w_{mn}$ is a $K \times K$ kernel function, $\ast$ is a convolutional operator, and $b_n$ is a bias term. Here, $\phi$ is a non-linear, typically sigmoid, activation function. The sub-sampling layer (S-layer) simply reduces the dimensions of the input feature map $x'_n$. As indicated in (4.1), large data volumes need to be processed by the CNN. Hence, a memory-based architecture, as proposed in this section, can be highly...
effective in implementing CNNs.

Figure 4.1(b) shows the block diagram of a conventional CNN system [66], where a conventional SRAM stores the weights $w_{mn}$, and the input feature map $x_m$ is stored in a register bank. The register contents are updated with the output feature map $y_n$ at the completion of one layer.

The energy consumption to process a single feature map in a conventional system can be expressed as

$$E_{conv} = K^2 E_{read} + E_{leak} + (L - K + 1)^2 K^2 E_{MAC} + E_{reg} \quad (4.2)$$

where $E_{read}$ and $E_{leak} = P_{leak} T_{conv}$ represent the single word SRAM read energy and the SRAM leakage energy per feature map computation, respectively. Here, $P_{leak}$ is the leakage power consumption and $T_{conv}$ is the time needed to generate a feature map. It is assumed that a deep-sleep mode is enabled during standby using techniques such as power gating or lowering the supply voltage for the BCA [49]. $E_{MAC}$ and $E_{reg}$ are the multiplier and accumulator (MAC) and register bank energies, respectively.

4.1.2 Proposed DIMA-Based CNN System

4.1.2.1 The DIMA-Based CNN Architecture

Figure 4.2 shows DIMA-based CNN architecture, where $X_c$ is the number of columns in the SRAM array. The $K^2$ coefficients of $w_{mn}$ are stored in a block of $B_w \times K^2$ bitcells, where $B_w$ is the bit precision of $w_{mn}$, and each $B_w$-bit word is stored in one column. In addition, $w_{mn}$ with the same value of $n$ are horizontally aligned in a single row occupying $NK^2$ columns. If $NK^2 > X_c$, the $w_{mn}$ are stored in $\lceil NK^2 / X_c \rceil$ rows. The $w_{mn}$s required to compute a single pixel of $y_n(x, y)$ are FR and multiplied with $x_m$ provided in the digital domain from feature map registers.

In the following, we employ $D$ and $P$ to represent specific values of $w_{mn}$ and $x_m$, respectively, in order to simplify the exposition. In the DIMA, negative values are difficult to represent in analog domain. An 1’s complement representation is employed for $D$ and an
Figure 4.2: DIMA-based architecture for CNN.

unsigned representation for $P$. Thus, $|D| \times P$ and $S_D = \text{sign}(D)$ is computed separately. Here, $|D|$ is computed as follows:

$$|D| = \begin{cases} \sum_{k=0}^{B_D-1} 2^k d_k \propto \Delta V_{BLB}(D), & \text{if } D \geq 0 \\ \sum_{k=0}^{B_D-1} 2^k d_k \propto \Delta V_{BL}(D), & \text{if } D < 0 \end{cases} \quad (4.3)$$

The $S_D$ is obtained by using a differential amplifier with $\Delta V_{BL}(D)$ and $\Delta V_{BLB}(D)$ as its inputs, which is then used as a select signal of the multiplexer to select the greater of $V_{BL}(D)$ and $V_{BLB}(D)$ thereby generating $|D|$ as the output $V_{mux}$ as shown in Fig. 4.2.

Next, the outputs of multipliers are transferred to a positive or negative rail via demultiplexers based on the $S_D$. The rails are shared with multiple columns so that the absolute values of positive and negative products in (4.3) are added separately via charge-sharing on each rail. Finally, each value on the rail is converted into a digital number through two analog-to-digital converters (ADCs), whose outputs are subtracted to generate the convolution sum. These steps are repeated $\lceil NK^2/X_c \rceil$ times if $NK^2 > X_c$ to fetch all the required $w_{mn}$s. Then, the sequentially generated outputs of the subtractor and the $b_n$ are accumulated. The activation function is implemented with three additions and two shifts in
Figure 4.3: Capacitive multiplier behavioral model (4.4) validation with circuit simulation in 45 nm ($C = 10 fF$).

the digital domain as introduced in [66].

4.1.2.2 Inner Products via BLP

The product of $\Delta V_{BLB}(D)$ (or $\Delta V_{BL}(D)$) from the multiplexer and a $B_p$-bit digital value $P$ is obtained via the capacitive multiplier shown in Fig. 2.11(a). The multiplier output $\Delta V_m$ from $V_{PRE}$ is

$$\Delta V_m = (0.5)^{B_p} P \Delta V_{BLB}(D) = \alpha P D$$

(4.4)

where $\alpha$ is a constant depending on $T_{LSB}$, $C_{BL}$, and $R_{BL}$.

The voltage level $V_{PRE} - \alpha \sum_{j=0}^{K^2-1} D_j P_j$ corresponding to the inner product between $\vec{D} = D_0, D_1, ..., D_{K^2-1}$ and $\vec{P} = P_0, P_1, ..., P_{K^2-1}$ can be achieved by charge-sharing the multipliers’ outputs in $K^2$ columns.
Table 4.1: Design and model parameters of DIMA.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{DD}$</td>
<td>1.1 V</td>
<td>$f$</td>
<td>1 GHz</td>
</tr>
<tr>
<td>input $L$</td>
<td>32</td>
<td>$K$</td>
<td>5</td>
</tr>
<tr>
<td>$B_w$</td>
<td>8</td>
<td>$B_x$</td>
<td>6</td>
</tr>
<tr>
<td>$N$</td>
<td>$C_1 : 6, C_3 : 16, F_5 : 120, F_6 : 10$</td>
<td>$f_0, \ldots, f_4$</td>
<td>$1, 1.11 \times 10^{-2}, -5.4684 \times 10^{-4}, 4.0506 \times 10^{-6}$</td>
</tr>
<tr>
<td>$c_0, \ldots, c_4$</td>
<td>$-9.5 \times 10^{-3}, 3.2 \times 10^{-2}, 3.5 \times 10^{-4}, -1.7 \times 10^{-5}, 1.3 \times 10^{-7}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.1.3 Energy and Behavioral Models with Circuit Non-Idealities

The analog-intensive DIMA operation is subject to a number of circuit-level non-idealities. Dominant among these are: (a) non-linearity of the multi-row READ process, which is caused by voltage-dependent discharge path resistance $R$, (b) local transistor threshold voltage $V_t$-mismatch across bitcells caused by random dopant fluctuations, and (c) non-ideality of analog multiplication.

The non-linearity of FR was previously modeled in [35] by a polynomial fit as $\Delta V'_{BLB}(D) = \sum_{k=0}^{4} c_k D^k$, where $\Delta V'_{BLB}(D)$ is a distorted version of $\Delta V_{BLB}(D)$, and $c_k$s are the fitting parameters. In this section, we model the $V_t$-mismatch and non-ideality of the analog multiplier.

The impact of $V_t$-mismatch is modeled as a Gaussian distributed random variable as shown below:

$$\Delta \hat{V}_{BLB}(D) \sim N(\Delta V_{BLB}(D), \sigma_D^2)$$  \hspace{1cm} (4.5)

where $\sigma_D^2$ is the variance of $\Delta V_{BLB}$ due to $V_t$-mismatch across bitcells corresponding to the stored value $D$.

The behavior of multiplier can be captured by a polynomial model with fitting parameters $f_{0,1,2,3}$ as follows:

$$\Delta V_m = f_0 \Delta V_{BLB}(D) P + f_1 \Delta V_{BLB}(D) + f_2 P + f_3$$  \hspace{1cm} (4.6)

These models are employed in the Section 4.1.4 to study the impact of circuit non-idealities.
Figure 4.4: Estimated relative delays for CNN.

on application level behavior.

The energy consumption of DIMA to process a feature map is given by

\[ E_{DIMA} = (L - K + 1)^2 K^2 E_{FR} + E_{\text{leak}_{DIMA}} + (L - K + 1)^2 K^2 E_{\text{MAC}_{add}} + E_{\text{reg}} \]  

where \( E_{FR} \) is the energy consumed to read a single word by the FR. The scaling factor of the first term in (4.7) is larger than that of (4.2). This is because the DIMA reads the \( w_{mn} \)s from SRAM again whenever the processing window slides as the analog level from FR cannot be sustained. The leakage energy \( E_{\text{leak}_{DIMA}} = P_{\text{leak}} T_{DIMA} \), where \( T_{DIMA} \) is the processing time of a feature map by the DIMA, and smaller than \( T_{\text{conv}} \). Thus, \( E_{\text{leak}_{DIMA}} \) is also smaller than \( E_{\text{leak}} \). The \( E_{\text{MAC}_{add}} \) is the energy consumed for the analog multiplier and charge sharing based adder. This is also smaller than the \( E_{\text{MAC}} \) due to the operation with small voltage swing.

4.1.4 Simulation Results

In this section, a handwritten digit recognition with MNIST database [61] is chosen as an application to measure the performance of DIMA system. All the design and model
parameters are summarized in Table 4.1. The variant of LeNet5 [65] is employed including a total of six layers.

Horizontally aligned four banks of the SRAM array with a size of 512×256 bitcells are employed for the DIMA to store trained kernel \( w_{mn} \). Thus, roughly 40 \( w_{mn} \)s can be aligned in one row and processed at a time \( (X_c = 256 \times 4)/K^2 \approx 40 \). The embedded SRAM’s IO is 32 bits in the conventional system. The number of multipliers is \( K^2 = 25 \) in the conventional system to achieve an area comparable to the DIMA.

4.1.4.1 Model Validation

HSPICE simulations are performed in 45 nm SOI process technology to obtain the behavioral models. The normalized standard deviation \( (\sigma_D/\mu_D) \) of \( \Delta V_{BLE} \) in the model (4.5) is measured by Monte Carlo HSPICE simulations. The minimum value 7\% is achieved with \( D = 15 \), and a maximum value 12.5\% is obtained with \( D = 1 \). The simulated and modeled behavior of capacitive multiplier with \( C = 10fF \) is described in Fig. 4.3 with fitting parameters \( f_{0,1,2,3} \) in Table 4.1.

4.1.4.2 Recognition Accuracy

The \( w_{mn} \) are obtained with 60000 training images from MNIST dataset [61] with back propagation algorithm through roughly 80 iterations. The error rates are measured on
MNIST test data set by the system simulations with the behavioral models in following configurations: (1) conventional system with a floating-point numbers, (2) with fixed-point ($B_w$ and $B_e$), (3) DIMA system with fixed point and (4) with fixed point numbers and $w_{mn}$ trained reflecting non-linearity of FR. Error rates of 0.8% and 0.85% are achieved in the first and second configurations, respectively. In the third configuration, the error rate is degraded to 1.36% due to circuit non-idealities. However, the error rates is 0.87% in the fourth configuration. It indicates that the non-idealities from DIMA is effectively compensated by the inherent error resiliency of CNN.

4.1.4.3 Energy and Delay Savings

The SRAM access and multiplication of conventional system require two cycles and one cycle of clock, respectively, and those can be pipelined. On the other hand, the FR and BLP of DIMA require a total of 20 cycles, where the DIMA processes $X_c = 1024$ words reading and multiplications in parallel. Based on the previous specifications, each delay from the convolutional and fully connected layers and cumulative delays are estimated in Fig. 4.4. The DIMA achieves roughly $4.9 \times$ reduced total delays achieving higher throughput in the memory intensive F5 layer due to the parallel read and computations.

Based on the energy consumption to process a feature map modeled in (4.2) and (4.7), the energy consumptions to process all the feature maps of layers are estimated in Fig. 4.5. About $5.0 \times$ energy saving in the overall system is achieved mostly by the low-power inner product computation and the reduced leakage energy due to high throughput.

In conclusion, $24.5 \times$ smaller EDP is achieved by the DIMA as compared to the conventional system with 0.02% larger error rate.

4.2 Sparse Distributed Memory (SDM)\textsuperscript{2}

There is much interest in exploring brain-inspired models of computation that can provide robust system behavior for inference applications while achieving high energy efficiency [67–

\textsuperscript{2}This section is adopted from M. Kang and N. R. Shanbhag, “In-memory computing architectures for sparse distributed memory,” IEEE Transactions on Biomedical Circuits and Systems. © 2016 IEEE]
The Sparse Distributed Memory (SDM) [71] (see Fig. 4.6) is one such computational model of the human brain. An SDM can be trained to remember sparse data vectors and retrieve these when presented with noisy or incomplete versions of the stored vectors. This is similar to human brain’s ability to associate related memory given noisy sensory input by conceptualizing/categorizing incomplete information [72].

Being a memory array, the SDM input is a 2-tuple \((p, d)\), where \(p\) and \(d\) are the \(J\)-bit address and \(K\)-bit data, respectively (we assume \(J = K\) in the rest of this section). In a SDM, data vectors \(d\) are first stored (WRITE operation). The address decoder (AD) projects the \(J\)-bit address vector \(p\) on to a higher \(I\)-dimensional \((I \gg J)\) space, and then uses this high dimensional representation \(s\) of \(p\) as the decoded address into the counter array (CA), where \(d\) is stored in a distributed fashion. In the READ mode, the address \(p\) is first decoded by the AD, and the decoded address \(s\) used to retrieve the stored data from the CA. The sparse and distributed nature of data processed and stored in a SDM provides inherent robustness to noise or imprecision in the input data. The SDM can also be employed in an auto- or hetero-associative mode to achieve even greater robustness to data errors.

However, a straightforward SDM implementation will consume much energy and will be slow because the SDM operates in a high (hyper)-dimensional space [71], e.g., typical SDM parameters are: \(I = 2 \times 10^3\) to \(10^6\), \(J \geq 256\), \(B_c \geq 5\), where \(B_c\) is a bit precision of each counter in the CA [72,73]. Such an implementation in a 65 nm CMOS process would consume 77 \(\mu\)J and have a delay of 2 ms per READ. In fact, the dominant (about 80\% as shown in Section 4.2.4) source of energy consumption and delay in the SDM can be attributed to the AD. Hence, several high throughput architectures for the AD based on SRAM and DRAM have been proposed. These achieve speed-up by parallelizing the AD using multiple memory blocks [74]. However, these architectures suffer from an inter-block throughput bottleneck. To remove memory read operation, a shift register-based AD architecture [75] has also been proposed. However, this architecture suffers from large dynamic energy consumption and occupies a large area compared to memory-based architectures. Mixed-signal AD implementations [75,76] employ a current mirror to evaluate the Hamming distances in parallel thereby achieving high throughput. However, the large content addressable memory bitcell dimension (i.e., 11 transistors including the current mirror) results in a loss of storage den-
Figure 4.6: Sparse distributed memory (SDM): (a) architecture ($S_H$: number of selected rows), and (b) example of SDM operation with address decoder (AD) and counter array (CA) ($I = 8$, $J = K = 7$, and $S_H = 3$).
sity, and the bias currents results in high DC power consumption. The design of SDM is also implemented by employing resistive memory devices [73].

Implementing the SDM model requires large storage capacity closely integrated with computation. Traditional processor-memory architectures separate low-swing memory storage functionality from high-swing logic. This separation exists even in the so-called processor-in-memory architecture [23, 25], and is the source of both a throughput bottleneck and energy consumption. In fact, conventional architectures fail to exploit an important feature of the SDM [72] - the ability to compensate for hardware noise/errors in addition to noise/errors in the input data. The DIMA preserves the storage density, the conventional SRAM’s read/write functionality, and is well-suited for inference kernels such as SDM which can compensate for non-deterministic hardware operations.

In this section, we describe an architecture and circuit implementation of a DIMA-based SDM (DIMA-SDM) [37, 38], which incorporates two proposed techniques (1) DIMA-based AD (DIMA-AD), and (2) CA with a hierarchical binary decision (CA-HBD). Circuit and system simulations in a 65 nm CMOS process show that the DIMA-SDM reduces energy and delay simultaneously by a factor of up to 25× and 12×, respectively, over the conventional SDM architecture in the auto- and hetero-associative modes with negligible loss in accuracy.

The rest of the section is organized as follows. Section 5.1 provides the necessary background on SDM, and associative memory. Section 4.2.2 describes the proposed DIMA-AD and CA-HBD architectures. Section 4.2.3 develops circuit-aware energy, delay, and behavioral models for the entire signal processing chain. Section 4.2.4 presents circuit and system simulation results demonstrating the performance, delay reduction, and energy savings of the architecture over the conventional SDM.

4.2.1 Background

This section introduces the necessary background on the topics of SDM [71].
Figure 4.7: The conventional SDM: (a) an $M$-parallel block architecture, and (b) architecture of a single block.

### 4.2.1.1 SDM Operations

Figure 4.6 shows that the SDM accepts as input a 2-tuple $(p, d)$ with a $J$-bit address $p$ and $J$-bit data $d$. The SDM architecture (see Fig. 4.6(a)) includes: (1) a $J$-bit address decoder AD to evaluate the Hamming distance between $p$ and the $I, J$-bit addresses stored in a $I \times J$ memory array $A$ in the AD, and (2) a counter array $CA$ with a counter and a memory array $C$ to store the $IJ B_c$-bit counts.

**WRITE Operation:** During the WRITE operation, the AD generates an $I$-bit decoded row address $s = [s_1, s_2, \ldots, s_I]$ (see Fig. 4.6), as follows:

$$s_i = \text{sgn}\left\{ R - \sum_{j=1}^{J} (a_{ij} \oplus p_j) \right\}, \ (i = 1, 2, \ldots, I) \quad (4.8)$$

$$\text{sgn}(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

where $\oplus$ is the binary EXOR operator, $a_i = [a_{i1}, a_{i2}, \ldots, a_{iJ}]$ is the $i$-th address stored in $A$, $p = [p_1, p_2, \ldots, p_J]$ is the input address, and $R$ is a user-defined radius/threshold.

The decoded row address $s$ and the data $d$ are employed in the CA to update the count,
as follows:

\[
c_{ij} \leftarrow \begin{cases} 
  c_{ij} + s_i, & \text{if } d_j = 1 \\
  c_{ij} - s_i, & \text{otherwise}
\end{cases}
\]  

(4.9)

where \( d = [d_1, d_2, \ldots, d_J] \). Note that the contents of \( C \) are updated only when \( s_i = 1 \), i.e., only the selected rows are updated.

**READ Operation:** During the READ operation, AD generates the row address \( s \) in the same manner as in the WRITE operation described by (4.8). Then, the SDM output is read out as:

\[
y_j = sgn(s \cdot c_j), \ (j = 1, 2, \ldots, J)
\]  

(4.10)

where \( c_j \) is the \( j \)-th column vector of \( C \), and \( y = [y_1, y_2, \ldots, y_J] \) is the output word.

### 4.2.1.2 Associative Memory

In associative memories, the data read is the stored data that is most strongly associated with the contents of the input rather than a specific address. There are two types of associative memories: (1) auto-associative memory, and (2) hetero-associative memory. The SDM can operate in both modes of associative recall and when it does, the SDM exhibits even stronger robustness to noise/errors in data.

In the auto-associative mode, the SDM is trained by selecting its input 2-tuple \( (p, d) \) from the training set \( S_t = \{(t_1, t_1), (t_2, t_2), \ldots\} \), i.e., both the address \( p \) and the data \( d \) are assigned the same value [77]. On the other hand, in the hetero-associative mode, the SDM is trained by selecting its input 2-tuple \( (p, d) \) from the training set \( S_t = \{(t_{11}, t_{12}), (t_{21}, t_{22}), \ldots\} \), i.e., \( p \) and the \( d \) are assigned different values.

During the classification/decision-making phase, in both associative modes, the SDM is operated in an iterative manner where initially \( p = l \), where \( l \) is a noisy/incomplete version of the stored data. Then, in subsequent iterations, \( p \) is set to the current output of the
SDM. Thus, the classification phase of the SDM is described as follows:

\[ p[n] = \begin{cases} 
  l, & \text{if } n = 1 \\
  y[n-1], & \text{if } n > 1 
\end{cases} \quad (4.11) \]

where \( n \) is the time index, and \( l \) is an initial input. The output \( y[n] \) converges to an error-free/closest version of \( l \) that was stored during the training phase. The SDM’s auto- and hetero associative modes can be interpreted as the human brain’s ability to extract a pattern from noise and locate the next pattern in a certain sequence given the current pattern [72].

4.2.1.3 Conventional SDM Architecture

The conventional SDM architecture is multi-block [74] (see Fig. 4.7(a)) in order to enhance throughput. The multi-block SDM architecture comprises \( M \) blocks (\( B_1, B_2, \ldots, B_M \)) that operate in parallel, where each block has its own address decoder \( \text{AD}_m \) with memory array \( A_m \) of size \( (I/M) \times J \) bits, a counter array \( \text{CA}_m \) with memory array \( C_m \) of size \( (I/M) \times (J B_c) \) bits, and decoded row address \( s_m \) (\( m = 1, \ldots, M \)). The memory array \( A \) in AD is implemented via SRAMs for high throughput, while the memory array \( C \) in CA is implemented using DRAM, Flash, and PRAM, in order to achieve high storage densities.

The architecture of each block (see Fig. 4.7(b)) indicates that \( \text{AD}_m \) computes the Hamming distance between the input address \( p \) and \( (I/M) \) stored addresses in \( A_m \), while \( \text{CA}_m \) generates a partial sum, which is then accumulated and thresholded during the READ operation. The each partial sum requires an additional \( B_x \) bits in addition to \( B_c \) bits per single counter in order to prevent overflow, where \( B_x \) depends upon the sparsity of stored data and \( R \). Thus, the multi-block SDM architecture requires \( J(B_c + B_x) \) global bitlines (GBLs) per block to transfer partial sums to the decision block.

The conventional architecture in Fig. 4.7(a) has a number of drawbacks. Key among these are the following:

1. The Hamming distance computation in the \( \text{AD}_m \) requires access to all the memory locations. The throughput of \( \text{AD}_m \) is limited by the SRAM read out bandwidth. In
particular, multiple read out cycles are required to read a single $a_i$ in conventional memory and processor architecture.

This is because conventional SRAMs need to employ column multiplexing, whereby multiple bitlines (BLs) share a single sense amplifier (SA). Reliability constraints force the SA and other peripheral circuits to be designed with area that is 4×-to-8× of that of a bitcell, thereby necessitating column multiplexing. Additionally, there is another throughput bottleneck due to limited memory I/O port or bus width in von Neumann architectures [21]. Typically, $J/B_{IO} \geq 4$ read outs are required to read the entire data in single row even in application processors with custom-designed on-chip SRAM [49], where $B_{IO}$ is the bit width of the SRAM I/O port or the bus width.

2. The additional digital blocks in the AD, such as the adder tree and EXOR gates, lead to energy consumption and area overhead.

3. Routing the $GBL$s in the $CA_m$ is made difficult because of their large number ($J(B_c + B_x)$ per block), and because of the small bitcell area ($4F^2$, $F$: 1/2 of BL pitch) due to the use of high density memories [78, 79].

Section 4.2.2 describes how these drawbacks of the conventional architecture can be overcome.

4.2.2 Proposed Architecture

In this section, a DIMA-based SDM (DIMA-SDM) is proposed (see Fig. 4.8(a)) to address the drawbacks of conventional architecture listed in Section 4.2.1.3. In particular, DIMA-SDM employs the following key techniques:

- The AD is designed using DIMA (DIMA-AD) (see Fig. 4.8(b)) in order to overcome its bandwidth limitation and eliminate the use of digital logic.

- The CA is implemented using a hierarchical binary decision (HBD) technique (CA-HBD) as shown in Fig. 4.8(c) in order to minimize the routing overhead of $GBL$s.
Figure 4.8: Proposed SDM architecture (DIMA-SDM): (a) architecture of single block, (b) AD with DIMA (DIMA-AD) including deeply embedded mixed signal processing units, and (c) CA with hierarchical binary decision (CA-HBD) (N_{GBL}: number of GBLs).
4.2.2.1 DIMA-Based Address Decoder (DIMA-AD)

The proposed DIMA-AD generates the Hamming distance per (4.8) via a three-step process: (1) FR process generates BL voltages $V_{BL}$ and $V_{BLB}$ that are proportional to the sum $a_{ij} + p_j$ over the field of real numbers, followed by (2) the use of BLP to compute $a_{ij} \oplus p_j$ and (3) finally the Hamming distance (via a capacitive adder). These steps are described next.

The FR step begins with the application of access pulses simultaneously to the rows storing $a_{ij}$ and $p_j$ such that the pulse width $T \ll R_{BL}C_{BL}$, where $R_{BL}C_{BL}$ is the $RC$ time constant of $BL/BLB$ [35]. This results in a $BL/BLB$ voltage (see Fig. 4.9) given by:

$$V_{BL} = V_{PRE} - (a_{ij} + p_j)\Delta V_{BL} \quad (4.12)$$
$$V_{BLB} = V_{PRE} - (a_{ij} + p_j)\Delta V_{BL} \quad (4.13)$$

where $\Delta V_{BL} = V_{PRE}(T/R_{BL}C_{BL})$. A replica bitcell is employed to avoid writing $p$ into the main array $A$ (see Fig. 4.8(b)).

The second step (BLP) begins with the $BL/BLB$ provided as inputs to differential comparators [80] sized to fit within a single bitcell pitch with an appropriately selected reference voltage $V_{ref} = V_{PRE} - \Delta V_{BL}/2$. Doing so results in binary valued comparator outputs:

$$X_{BL} = a_{ij}p_j = sgn(V_{diff,BL}) = sgn\{0.5 - (\overline{a_{ij}} + \overline{p_j})\} \quad (4.14)$$
$$X_{BLB} = a_{ij} + p_j = sgn(V_{diff,BLB}) = sgn\{0.5 - (a_{ij} + p_j)\}$$

A NOR2 gate that combines the comparator outputs generates $a_{ij} \oplus p_j$ as follows:

$$a_{ij} \oplus p_j = sgn\{0.5 - (a_{ij} + p_j)\} + sgn\{0.5 - (\overline{a_{ij}} + \overline{p_j})\} \quad (4.15)$$

Next, a $J$-bit capacitive adder (see Fig. 4.8(a)) accepts $a_{ij} \oplus p_j$ from the NOR2 gate output and employs charge redistribution to compute the summation in (4.8) as follows:

$$V_{sum,i} = \frac{1}{J} \sum_{j=1}^{J} (1 - a_{ij} \oplus p_j)V_{PRE} \quad (4.16)$$
The last step involves an analog comparator that generates the decoded address bit $s_i$ as shown below:

$$s_i = \text{sgn}(V_{\text{sum},i} - V_R)$$ (4.17)

This sequence of operations is repeated $I/M$ times.

Thus, DIMA-AD reads $a_i$ in a single read cycle (single precharge) and has $\approx J/B_{\text{IO}}$ times higher throughput as compared to the conventional AD. Additionally, DIMA-AD is more energy-efficient than a digital implementation because the capacitive adder employs small capacitances (i.e., $C = 10 \text{ fF}$) and requires a simple switching operation.

4.2.2.2 Counter Array Using Hierarchical Binary Decision (CA-HBD)

The proposed CA-HBD architecture minimizes the inter-block data transfer as shown in Fig. 4.8(c), where GDB and LDB are global and local decision blocks, respectively. The CA-HBD architecture requires recording the row access count $N_i$, i.e., the number of accesses to each physical address $a_i$ in $A$ during the WRITE operation. The row access count $N_i$ is recorded in an additional column in the CA.

During the READ operation, the LDB of the $m^{th}$ block generates a local binary decision
Figure 4.10: Global decision block (GDB) to incorporate local decisions \( (y_{m,j}) \) with impact factor \( N_m \).

\( y_{m,j} \) and \( N_m \) as follows:

\[
N_m = \sum_{i \in H_m} N_i \quad \text{(4.18)}
\]

\[
y_{m,j} = \text{sgn}(\sum_{i \in H_m} c_{ij}) \quad \text{(4.19)}
\]

where \( H_m \) is the set of row indices in the \( m^{th} \) block \( CA_m \) that were selected during READ, and \( N_m \) represents the sum of the row access counts for these rows.

Finally, the GDB (see Fig. 4.10) generates the final SDM output bit \( y_j \) as follows:

\[
y_j = \text{sgn}\{\sum_{m=1}^{M} \text{sign}(y_{m,j})N_m\}, \text{ where} \quad \text{(4.20)}
\]

\[
\text{sign}(x) = \begin{cases} 
1, & \text{if } x > 0 \\
-1, & \text{otherwise}
\end{cases}
\]

Thus, the GDB weights \( CA_m \)'s contribution \( y_{m,j} \) by \( N_m \)'s in order to assign more weight to those blocks which were accessed more frequently during the WRITE phase. In this manner, the LDB transmits compressed information to the GDB as \( y_{m,j} \)'s are binary numbers. Thus, \( J \)-bits are required instead of \( J(B_c + B_x) \)-bits as shown in Fig. 4.8(c), thereby minimizing
the delay and the energy penalty for the data transfer.

4.2.3 Circuit-Aware Behavioral, Energy, and Delay Models

The analog-intensive FR and BLP operations of the DIMA-AD are intrinsically vulnerable to various sources of noise due to its low-SNR operation. The dominant sources of noise in the DIMA-AD are: (1) local transistor threshold voltage $V_{th}$-variation across bitcells caused by random dopant fluctuations, and (2) input offset of the analog comparator. This section derives behavioral models of the non-ideal behavior of DIMA-AD to predict system performance. Energy and delay models are also provided.

4.2.3.1 Behavioral Model

In the FR operation, the $V_{th}$ variations were modeled as a Gaussian distributed random variable in [35]. In this section, two binary numbers $a$ and $p$ (we omit indices $i$ and $j$ for simplicity) are FR. The impact of $V_{th}$-mismatch on the $BL/BLB$ voltages is modeled as follows:

$$f_{V_{BL}}(V_{BL}; a, p) = \mathcal{N}(V_{PRE} - (\pi + p)\Delta V_{BL}, (\pi + \pi)\sigma^2_{cell})$$
$$f_{V_{BLB}}(V_{BLB}; a, p) = \mathcal{N}(V_{PRE} - (a + p)\Delta V_{BL}, (a + p)\sigma^2_{cell})$$

(4.21)

where $f_{V_{BL}}(V_{BL}; a, p)$ and $f_{V_{BLB}}(V_{BLB}; a, p)$ are the probability density functions of $V_{BL}$ and $V_{BLB}$, respectively, parametrized by $a$ and $p$. $\mathcal{N}(\mu, \sigma^2)$ is the normal distribution with mean $\mu$, and variance $\sigma^2$, and $\sigma^2_{cell}$ is the variance of $\Delta V_{BL}$ due to $V_{th}$ variation across the storage array $A$. It is assumed that $V_{th}$ variations for the bit- and replica cells are identical.
The comparator outputs $X_{BL}$ and $X_{BLB}$ are obtained as:

$$X_{BL} = \begin{cases} 
0 & \text{if } V_{BL} < V_{ref} + V_{offset} \\
1 & \text{otherwise}
\end{cases}$$

$$X_{BLB} = \begin{cases} 
0 & \text{if } V_{BLB} < V_{ref} + V_{offset} \\
1 & \text{otherwise}
\end{cases}$$

$$f(V_{OS}) = \mathcal{N}(0, \sigma_{comp}^2) \quad (4.22)$$

where an input offset voltage ($V_{offset}$) of the comparator is modeled as a zero mean Gaussian random variable with variance $\sigma_{comp}^2$.

The charge injection noise in the switches and thermal noise/mismatch of capacitors in the capacitive adder are made negligible by ensuring $C > 10 \text{ fF}$ [81]. The single comparator at the output of capacitive adder can be designed to have a small input offset by using large transistor sizes and calibration techniques.

The behavioral models in this section are validated in Section 4.2.4.

### 4.2.3.2 Delay and Energy Models

The delay per READ of the conventional SDM and the DIMA-SDM are described as follows:

$$T_{SDM} = T_{AD} + T_{CA} \quad (4.23)$$

$$T_{AD} = (I/M)(J/B_{IO})T_{read}$$

$$T_{CA} = S_{H,\text{max}}(J/B_{IO})T_{read} + M \left[ J(B_c + B_x)/N_{GBL} \right] T_{GBL}$$

$$T_{DIMA-SDM} = T_{DIMA-AD} + T_{CA-HBD} \quad (4.24)$$

$$T_{DIMA-AD} = (I/M)T_{read}$$

$$T_{CA-HBD} = S_{H,\text{max}}(J/B_{IO})T_{read} + M \left[ J/N_{GBL} \right] T_{GBL}$$
Figure 4.11: Normalized delay of single READ operation with $B_{IO} = 8 - 64$ ($B_{IO} : J = 1 : 4 - 32$).

where $T_{AD}$ ($T_{DIMA-AD}$) and $T_{CA}$ ($T_{CA-HBD}$) are the delay for AD (DIMA-AD) and CA (CA-HBD). In addition, $T_{read}$ is the delay for a single read access for memory arrays A and C, $T_{GBL}$ is the delay in transferring a single bit via the GBLs, $N_{GBL}$ is the number of GBLs, and $S_{H,max}$ is the maximum number of selected addresses per block. It is assumed that all other blocks are operating in parallel while the memories are being accessed. Hence, the delay of blocks such as the logic blocks in the conventional AD and the capacitive adder in DIMA-AD are not included in (4.23)-(4.24). The factors $(I/M)$ and $(J/B_{IO})$ in $T_{AD}$ are equal to the number of rows and the number of read outs per row, respectively, in AD$_m$. The partial sums per block are transferred serially through $N_{GBL}$ GBLs, thus requiring $\lceil J(B_c + B_x)/N_{GBL} \rceil$ cycles.

The throughput enhancement of DIMA-SDM over SDM derives from: (1) $J/B_{IO} \geq 4$ in $T_{AD}$, and (2) $B_c + B_x \geq 8$ in $T_{CA}$. The delay models in (4.23)-(4.24) are plotted in Fig. 4.11, where it is assumed that $T_{read}$ and $T_{GBL}$ take two clock cycles. DIMA-SDM demonstrates a $25\times$ smaller delay compared to SDM with $B_{IO} = 8$ due to the high bandwidth of DIMA-AD when $M = 4$. The benefit of HBD at $M = 2048$ is evident as there is a $3.2\times$ additional delay reduction as compared to DIMA-SDM without HBD.
The energy consumption per READ of the conventional SDM and the DIMA-SDM are modeled as follows:

\[ E_{SDM} = E_{AD} + E_{CA} \]  
\[ E_{AD} = I[(J/B_{IO})(E_{PRE} + E_{leak}) + J E_{SA} + E_{logic}] \]  
\[ E_{CA} = S_H B_c [(J/B_{IO})(E_{PRE} + E_{leak}) + J E_{SA}] \]  
\[ E_{PRE} = J C_{BL} \Delta V_{BL} V_{PRE} \]  
\[ E_{leak} = I J P_{leak-cell} T_{read} \]

\[ E_{DIMA-SDM} = E_{DIMA-AD} + E_{CA-HBD} \]  
\[ E_{DIMA-AD} = I (2E_{PRE} + E_{leak} + 2JE_{comp} + E_{a_add}) \]  
\[ E_{CA-HBD} < E_{CA} \]

where \( E_{AD} \) (\( E_{DIMA-AD} \)) and \( E_{CA} \) (\( E_{CA-HBD} \)) are the energy consumptions of the AD (DIMA-AD) and CA (CA-HBD), respectively. \( E_{PRE} \) and \( E_{leak} \) are the energy consumptions of the precharge and bitcell leakage for the entire memory array \( A \), respectively, and \( E_{SA} \) (\( E_{comp} \)) is for a single unit of sense amplifier (analog comparator). \( P_{leak-cell} \) is the leakage power of each bitcell. The energy consumptions of the analog capacitive adder in the DIMA-AD and logic blocks in the conventional AD per single Hamming distance computation are denoted by \( E_{a-add} \) and \( E_{logic} \), respectively. Energy consumptions from other blocks such as WL drivers [82], CA’s decision blocks are assumed to be negligible. It is assumed that the AD and CA can be placed into a deep sleep mode independently [49]. Note that \( E_{AD} \gg E_{CA} \) because \( I \gg S_H B_c \). The \( E_{DIMA-AD} \) has a scaling factor of two for the first and third terms as DIMA-AD reads \( a_{ij} \) and \( p_{ji} \), and employs two comparators per bitcell column.

The energy efficiency of DIMA-AD derives from the fact that: (1) the first term in \( E_{AD} \) and \( E_{DIMA-AD} \) is the largest, and because \( J/B_{IO} \geq 4 \), (2) \( E_{a-add} \ll E_{logic} \) as the capacitances in the capacitive adder are very small, e.g., 10 fF, and the capacitive adder requires only simple switching operations, and (3) leakage energy in \( E_{DIMA-AD} \) is smaller than that in
Figure 4.12: Normalized energy based on models (4.25), (4.26) with $\Delta V_{BL} = 75$ mV and 125 mV (obtained from Section 4.2.4.4) for SDM and DIMA-SDM, respectively.

$E_{AD}$ because the high-throughput (see delay models (4.23)-(4.24)) of DIMA-SDM permits it to be placed into a deep sleep mode much quicker than SDM [49].

The energy models (4.25) and (4.26) using typical design parameters from Table 4.2 are plotted in Fig. 4.12. The component values of (4.25) and (4.26) obtained from Section 4.2.4. Figure 4.12 indicates that DIMA-SDM achieves energy reductions of $2.1 \times$ to $12.4 \times$ over SDM.

4.2.4 Simulation Results

In this section, we apply SDM for handwritten digit recognition. Monte Carlo circuit (HSPICE) simulations in 65 nm CMOS process technology are employed to validate the models in (4.21) and (4.22). These models are employed in system simulations to estimate the output bad pixel ratio ($B_o$). Energy and throughput benefits are demonstrated via circuit simulations and the energy/throughput models (4.23), (4.24), (4.25), and (4.26).
4.2.4.1 System Configuration

Nine $16 \times 16$ binary shapes of numbers from 1 to 9 are employed to generate $p$ and $d$ as shown in Fig. 4.13(a) and (b). For each of the nine patterns, 225 noisy copies with input bad pixel ratio $B_i = 0.25$ are generated by randomly flipping 25% of the bits. These images form the training data set $S_t$ of size $255 \times 9 = 2295$ and are written into the SDM in the auto-associative mode ($p = d$).

In hetero-associative mode, during the training phase, $p$ and $d$ are assigned images corresponding to consecutive numbers, e.g., if $p$ is assigned image corresponding to 4 then $d$ is assigned the image corresponding to 5. Thus, during the READ operation, the SDM retrieves the image corresponding to the number that is one greater than the input number, as shown in Fig. 4.13(b).

After the training, 100 contaminated copies of each pattern (total of 900 inputs) with $B_i = 0.15, 0.25, 0.3$ are generated and provided as the address $p$ for classification. Four READ iterations of the auto- and hetero-associative memory are performed. The error immunity against faulty hardware increases with larger $R$ in (4.8) as each data vector $d$ is distributed across greater number of physical addresses. On the other hand, $R$ needs to be small enough not to create excessive intersection between physical addresses for different number’s images. To balance this trade-off, $R$ for WRITE and READ operations are set to 79 and 82, respectively.

The block size $I/M = 512$ is chosen to balance the read out delay and area efficiency of memory. The value of $B_{IO} = 64$ is chosen as its typical value of $J/B_{IO}$ ranges from 4 to 32 in conventional SRAM architectures [49], in order to permit the maximum bandwidth. In this specific application, $M = 4$ and $S_H \approx 0.1I$, which will be used in the rest of this section. The design parameters (other than $B_{IO}$ and $M$) used in the simulations are summarized in Table 4.2.

4.2.4.2 Model Validation

Monte-Carlo circuit simulations show that $\sigma_{cell}/\Delta V_{BL} = 6.5\%$ and the analog comparator has an input offset $\sigma_{comp} = 18 \text{mV}$. 

97
Figure 4.13: Behavior of SDM, DIMA-SDM (without HBD) and DIMA-SDM with $M = 4$ in: (a) auto-associative mode, (b) hetero-associative mode, and (c) the output bad pixel ratio $B_o[n]$ in the auto-associative mode.
Table 4.2: Design parameters for SDM.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{DD} (=V_{PRE})$</td>
<td>1 V</td>
<td>$M$</td>
<td>4-2048</td>
</tr>
<tr>
<td>$I/M$</td>
<td>512</td>
<td>$J$</td>
<td>256</td>
</tr>
<tr>
<td>$B_c (= B_x)$</td>
<td>4</td>
<td>$B_{IO}$</td>
<td>8-64</td>
</tr>
<tr>
<td>Clock frequency</td>
<td>1 GHz</td>
<td>$N_{GBL}$</td>
<td>256</td>
</tr>
<tr>
<td>$C$</td>
<td>10 fF</td>
<td>$C_{BL}$</td>
<td>230 fF</td>
</tr>
</tbody>
</table>

Figure 4.14: $V_{sum}$ from circuit simulations and system simulations with behavioral models (4.21) to (4.22) with $J = 8$ and $\Delta V_{BL} = 50 mV$.

The behavioral models of the entire analog signal processing chain from the bitcell to the final output $V_{sum}$ are validated as shown in Fig. 4.14, where the results of Monte-Carlo circuit simulations are compared with those from system simulations employing the behavioral models (4.21)-(4.22). Nine different combinations of $p$ and $a_i$ (with $J = 8$) are chosen as inputs. Figure 4.14 indicates that the maximum modeling error is 4.5% of the dynamic range of $V_{sum}$. This level of accuracy is sufficient for system performance estimation, as it is much less than the smallest non-zero Hamming distance.
Figure 4.15: Energy vs. $B_o$ trade-off with $n = 4$. Here $\Delta V_{BL} = 25 \text{mV}-125 \text{mV}$ for SDM and $75 \text{mV}-175 \text{mV}$ for DIMA-SDM.

4.2.4.3 System Performance

The output bad pixel ratio $B_o[n]$ in the $n$-th READ iteration is computed as:

$$B_o[n] = \frac{1}{900} \sum_{k=1}^{900} H_k[n]$$  \hspace{1cm} (4.27)

where $H_k[n]$ is the Hamming distance between the SDM output $y_k[n]$ at time index $n$ and the ideal output for the $k$-th input image.

Figure 4.13(c) shows that the conventional SDM, the DIMA-SDM (without HBD) and the DIMA-SDM, all converge to achieve a $B_o$ less than 2% for $n \geq 3$ when $B_i \leq 25\%$. Similar results were observed for the hetero-associative mode as well. Furthermore, SDM and DIMA-SDM were found to achieve $B_o[n]$ that were within $<5\%$ from each other for $n \leq 3$ (the $B_o$ of DIMA-SDM is slightly worse), for all three values of $B_i$. The $B_o$ of DIMA-SDM was higher than SDM by only $0.4\%$ for $n = 4$ and $B_i = 25\%$ indicating that the non-ideal behavior of DIMA-SDM is successfully compensated by the inherent noise immunity of SDM and the associative mode of operation. The $B_o$ degradation of the DIMA-SDM can be reduced by increasing the number of blocks $M$ with large $I$ so that more averaging can occur.
4.2.4.4 Delay and Energy Savings

The conventional SRAM read access and FR require two clock cycles, and the data transfer from the LDB to the GDB also requires two cycles. The proposed DIMA-SDM achieves $3.1 \times$ smaller delay over SDM as shown in Fig. 4.11 due to high bandwidth of DIMA-SDM with $M = 4$.

The various components of the energy models in (4.25) and (4.26) are measured via HSPICE simulations. To do so, the parasitic capacitance of $BL$ ($C_{BL} = 230 \text{ fF}$) is extracted from the layout of an SRAM bitcell. These energy components are a function of the $BL$ swing $\Delta V_{BL}$. The intrinsic robustness of SDM and the associative mode of operation enable a lower value of $\Delta V_{BL}$ to be employed as compared to a typical value in standard SRAM, thereby resulting in even greater energy savings.

Figure 4.15 shows the trend of $B_o$ with $\Delta V_{BL}$ scaling, and the $B_o > 2\%$ when $\Delta V_{BL} < 75 \text{ mV}$ and $125 \text{ mV}$ in the conventional SDM and DIMA-SDM, respectively. Thus, energy-optimal $\Delta V_{BL}$s are applied to both conventional and DIMA-SDM in order to obtain the energy breakdowns in Fig. 4.16. This figure shows that DIMA-SDM achieves approximately $2.1 \times$ reduced energy as compared to SDM.

Figure 4.16: Energy breakdown for a single READ operation with $B_{IO} = 64$, and $\Delta V_{BL} = 75 \text{ mV}$ for SDM and $125 \text{ mV}$ for DIMA-SDM.
4.3 Conclusion

In this chapter, the versatility of DIMA has been proven with two applications: CNN and SDM. More specifically, the SDM provides great potential to address stochastic and unreliable behavior of nanoscale fabrics due to its inherent robustness from ensemble decision-making process. However, conventional digital architectures fail to exploit the error tolerance for throughput and energy benefits. This chapter proved that DIMA can be a possible solution to such memory-intensive inference algorithms/applications. It was also demonstrated that the energy and throughput benefits can be further improved by co-optimizing the algorithm as shown in the HBD for SDM and error-aware re-training for CNN. The benefits of proposed architectures are expected to increase with data volume and input data size by saving the data movement costs more. Extensions to architectures based on emerging memory topologies are potential future directions.
Chapter 5

MATI: DIMA INSTRUCTION SET ARCHITECTURE

The significant energy and throughput benefits achieved by multi-functional DIMA naturally motivated us to consider a programmable deep in-memory instruction set architecture (ISA). However, as DIMA relies on array pitch-matched analog computations, their benefits have been demonstrated only for four fixed-function scenarios. This raises the question: *Can DIMA be made programmable without losing much of their energy and throughput benefits over their digital counterparts?* Answering this question is complicated because such architectures rely heavily on highly area-constrained (array pitch-matched) low-swing analog computations, and therefore introducing programmability into such architectures is a major challenge.

In this chapter, we propose MATI, a programmable deep in-memory ISA, that addresses programming challenges in DIMA via a synergistic combination of instruction set, architecture and circuit design. MATI builds upon the multi-functional (four-function) DIMA [39] introduced in Chapter 3, which will be called Compute Memory (CM) in this chapter, in order to enable a wide range of ML algorithms to be executed largely within the low-swing mixed-signal domain. To achieve this goal, a number of challenges in the design of the instruction set, architecture and circuit need to be addressed synergistically, including the following:

1. **analog ISA** challenge: the instruction set needs to accommodate the intrinsically sequential and analog nature of the four-stage architecture, while enabling a diversity of tasks that may be encountered across benchmarks, and at the same time expose the underlying CM mechanisms to be exploited by the compiler.

2. **functional density** challenge: each stage needs to implement many more functions
and options (CM has 8 while MATI realizes 32 functions spread across four stages) while satisfying geometric pitch-matching constraints imposed by the BCA and the inter-stage dynamic range matching issues.

3. **throughput-accuracy loss** challenge: the number of possible tasks increases dramatically (CM has 4 while MATI has more than 120) which causes a loss in throughput due to the increase in the worst-case delay. This loss in throughput leads to a loss in the accuracy of analog computations due to increased leakage.

We show that MATI achieves a high level of programmability without losing the efficiency benefits of deep in-memory computing via a synergistic combination of instruction set, architecture and circuit design. In particular, we propose the following techniques to address the above mentioned issues:

1. We address the analog ISA challenge by the use of a 51-bit macro (task level) instructions comprising four classes that are closely matched to MATI’s four-stage architecture and one additional class that allows an independent control of each stage function, as well as control of the data format, BL voltage swing, and ADC precision.

2. We address the functional density challenge by node sharing and reconfiguration to minimize the number of circuit elements needs per stage.

3. We address the throughput-accuracy loss challenge by a combination of analog pipelining, as late as possible (ALAP) scheduling of operations within each stage, and dynamic task period (DTP) assignment.

Employing silicon-validated energy, delay and behavioral models of deep in-memory components, we demonstrate that MATI is able to realize nine (potentially many more) ML benchmarks while incurring negligible overhead in energy (< 0.1%), and area (4.5%), and no overhead in throughput, over a pipelined version of CM [39]. In this process, MATI is able to simultaneously achieve enhancements in both energy (2.5× to 5.5×) and throughput (1.4× to 3.4×) for an overall energy-delay product improvement of up to 12.6× over fixed-function digital architectures. This work indicates the potential benefits of designing
Table 5.1: Machine learning (ML) algorithms.

<table>
<thead>
<tr>
<th>$f(D(W,X))$</th>
<th>Inner loop kernel</th>
<th>$f(\cdot)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>$\sum_{i=1}^{N} w[i] x[i]$</td>
<td>sign</td>
</tr>
<tr>
<td>Temp. Match. (L1)</td>
<td>$\sum_{i=1}^{N}</td>
<td>w[i] - x[i]</td>
</tr>
<tr>
<td>Temp. Match. (L2)</td>
<td>$\sum_{i=1}^{N} (w[i] - x[i])^2$</td>
<td>min</td>
</tr>
<tr>
<td>DNN</td>
<td>$\sum_{i=1}^{N} w[i] x[i]$</td>
<td>$-\ldots-$</td>
</tr>
<tr>
<td>Feature extraction (PCA)</td>
<td>$\sum_{i=1}^{N} w[i] x[i]$</td>
<td>sigmoid</td>
</tr>
<tr>
<td>$k$-NN (L1)</td>
<td>$\sum_{i=1}^{N}</td>
<td>w[i] - x[i]</td>
</tr>
<tr>
<td>$k$-NN (L2)</td>
<td>$\sum_{i=1}^{N} (w[i] - x[i])^2$</td>
<td>majority vote</td>
</tr>
<tr>
<td>Matched filter</td>
<td>$\sum_{i=1}^{N} w[i] x[i]$</td>
<td>min</td>
</tr>
<tr>
<td>Linear Regression</td>
<td>$\sum_{i=1}^{N} w[i]$</td>
<td>accumulate</td>
</tr>
<tr>
<td></td>
<td>$\sum_{i=1}^{N} w[i]^2$</td>
<td>accumulate</td>
</tr>
<tr>
<td></td>
<td>$\sum_{i=1}^{N} w[i] x[i]$</td>
<td>accumulate</td>
</tr>
</tbody>
</table>

a full-fledged compiler to map a wide variety of applications onto MATI, a key goal of our current work.

In Section 5.1, we describe various ML algorithms and provide an introduction to DIMA. MATI and its instruction set are described in Section 5.2. Section 5.3 describes the validation methodology followed by experimental results in Section 5.4. Section 5.5 concludes this chapter.

5.1 Background

In this section, we first analyze a variety of ML algorithms in order to identify commonalities in their data flow and then review DIMA briefly to show that these architectures are well-suited for ML algorithms.

ML algorithms [83] can be described by a series of nested loops where the innermost loop is the computation of the vector distance (VD) denoted as $D(W,X)$, between $N$-dimensional input vector $X$ and weight vector $W$ as shown in Table 5.1 and Listing 5.1. Commonly used VDs include the dot product (DP), L1 distance (Manhattan distance), L2 distance (Euclidean distance), and Hamming distance (HD). In ML algorithms such as the SVM, template matching, DNN, $k$-NN, and matched filter, VD computation dominates the
Listing 5.1: Pseudo code for core computations of inference / machine learning (ML) algorithms.

```plaintext
// x : element of input vector X
// y : element of output vector Y
// w[i] : i-th element of j-th weight vector W

y[1:No] = 0;
for (j = 0 ; j < No ; j++) {
    for (i = 0 ; i < N ; i++) {
        y[j] += d(w[j][i], x[i]);
    }
    y[j] = f(y[j]);
}
```

computational and memory access cost as $N$ can be quite large (e.g., 128 to 1024) and, being located in the innermost loop, this computation is invoked very frequently. Additionally, most ML algorithms have the following data-flow properties in common:

1. A single VD is obtained by first computing $N$ element-wise scalar distances (SDs) ($d(w[i], x[i])$) followed by an aggregation step, such as a sum or average, that generates the final scalar VD $D(W,X) = \sum_{i=1}^{N} d(w[i], x[i])$.

2. The VD between a single query vector $X$ and multiple (say $N_o$) weight vectors $W_j$ ($j = 1, 2, ... N_o$) needs to be computed, e.g., template matching computes VDs between one query vector $X$ and a large number of $W_j$s, while a DNN needs to compute the dot-product between an input feature map $X$ and many weight kernels $W_j$ to obtain $N_o$ output feature maps.

The conventional and DIMA architectures for inference algorithms are shown in Fig. 5.1(a) and (b), respectively. As described in Chapter 2, DIMA minimizes memory access costs associated with $W_j$ via FR and the cost of VD computations drastically by realizing them in the analog domain via BLP in close proximity to the BCA.
Figure 5.1: Block diagrams of: (a) a conventional digital (SRAM-based) architecture, and (b) deep in-memory architecture (DIMA). Analog (shaded) and digital (unshaded) blocks are indicated. The SRAM size is $N_{\text{row}} \times N_{\text{col}}$, and $B_w$ and $B_x$ are the bit precisions of scalar weight $w$ and scalar data $x$, respectively. Note that DIMA realizes the bulk of its computations in analog.
Figure 5.2: Sequential four-stage processing in DIMA for various cases: (a) un-pipelined fixed-function, (b) with operational diversity per stage, (c) with pipelining, and (d) as late as possible (ALAP) processing for S1, and dynamic task period (DTP) assignment.

\[ T_P = T_{S1} + T_{S2} + T_{S3} + T_{S4} \]

(a)

\[ T_P = T_{S1_{\text{max}}} + T_{S2_{\text{max}}} + T_{S3_{\text{max}}} + T_{S4_{\text{max}}} \]

(b)

\[ T_P = \max(T_{S1_{\text{max}}}, T_{S2_{\text{max}}}, T_{S3_{\text{max}}}, T_{S4_{\text{max}}}) \]

(c)

\[ T_{P,avg} = \frac{N_1 T_{P1} + N_2 T_{P2}}{N_1 + N_2} \]

(d)
5.2 MATI: A Programmable DIMA

We begin this section by describing in greater detail the various challenges that arise when programmability needs to be introduced into DIMA [39] along with proposed solutions to those challenges. These solutions are then leveraged to develop MATI, a programmable DIMA and its ISA.

5.2.1 Programmability Challenges in DIMA

The combination of algorithmic diversity and analog-heavy operations in DIMA creates a number of challenges that need to be addressed via a synergistic application of techniques at the instruction set, architecture, and circuit levels.

5.2.1.1 Analog ISA

DIMA’s analog processing chain imposes an intrinsic sequentiality in the order in which each of its four stages (S1-S4) needs to operate, i.e., S1:MR-READ → S2:BLP+CBLP → S3:ADC → S4:residual digital logic (RDL). Additionally, in order to cover a broad range of ML benchmarks, each stage needs to provide a sufficient number of options for the operations (operational diversity) it can execute. For example, the distribution of the number of operations in CM [39] is (S1=3, S2=2, S3=1, S4=2) whereas the corresponding distribution for MATI is (S1=6, S2=10, S3=9, S4=7). Therefore, MATI’s instruction set needs to accommodate this sequentiality along with analog circuit constraints for any arbitrary combination of operations required to realize a specific ML algorithm. By describing ML algorithms in terms of tasks, where each task is a specific sequence of operations obtained by selecting one in each stage, we can meet this sequentiality constraint. This is why MATI defines macro instructions which are composed of four Classes, closely matched to its four-stage architecture. In addition, consecutive processing stages need to be physically co-located to avoid substantial degradation in analog levels from one stage to the next. Furthermore, large capacitor ratio between consecutive stages (input-output 20:1) is required to limit the voltage drop $<5\%$. 

Figure 5.3: The MATI architecture (analog blocks shaded) in a single bank configuration comprising a 16 KB SRAM BCA, standard read and write paths (bottom), in-memory computation blocks (middle), and residual digital logic (RDL) and the controller (CTRLer) on the right. The natural sequentiality imposed by MATI's analog processing is exploited by defining Class shown next to each stage.

5.2.1.2 Functional Density

While each stage needs to provide a sufficient number of operations, unlike conventional architectures, this operational diversity needs to be provided while meeting stringent geometric pitch-matching constraints imposed by the BCA. The functional density challenge is addressed by sharing circuit components within each stage. For example, a single comparator in BLP is used for three operations: comparison, absolute value, and signed multiplication. Also, the capacitive multiplier in BLP is also used as an analog latch to support pipelined operations. Through such component sharing techniques, we were able to provide a five-function BLP vs. five-function BLP in CM [39] with an additional 16% increase in the area of the BLP. Note that the BLP itself consumes approximately 25% of the overall area and thus the impact of such area increases led to about a 4.5% increase in the total area.
5.2.1.3 Throughput-Accuracy Loss

The increased operation diversity of each stage leads to an increase in the task period \( T_P \) when accommodating the worst-case delay of each stage as shown in Figs. 5.2(a) and (b). This loss in throughput can be severe - up to \( 2 \times \) degradation when designing for the 9 ML benchmarks addressed in this chapter. Additionally, incorporating operational diversity also leads to idle times in each stage. The presence of these idle times is problematic in analog computation as analog values, which are typically stored on (area-constrained) capacitors, will degrade due to various leakage mechanisms, eventually leading to a loss in accuracy at the application level. Thus, there is a throughput and accuracy loss caused by introducing programmability features in DIMA.

We propose analog pipelining to address the loss in throughput (Fig. 5.2(c)). To accomplish such pipelining, we employ pre-existing capacitor-based samplers that were previously used for isolation in CM stages [39], as an analog latch between each stage. As this technique is straightforward to realize and could have been used in CM, we compare MATI with a pipelined CM (pipe-CM) when evaluating its benefits in section 5.4.

Though analog pipelining increases throughput, it does not solve the accuracy problem caused by idle times. In fact, the accuracy problem may even be worsened as all stages need to operate with a common period \( T_P \) (Fig. 5.2(c)), which is equal to the worst case delay across all stages. The idle time (Fig. 5.2(c)) of a specific stage is equal to the difference between \( T_P \) and S1 delay for the task. This problem is worst for S1 (MR-READ) as each \( BL \) is subject to the leakage contributions from all the bitcells in that column. To address the accuracy problem, we employ an as-late-as-possible (ALAP) (Fig. 5.2(d)) schedule for S1 so that its idle time is minimized. In order to obtain further throughput gains and minimize the accuracy problem, we employ dynamic task period (DTP) assignment (Fig. 5.2(e)), where the task period \( T_P \) is adapted in a task-dependent manner as described in Section 5.2.3.3.

The rest of this chapter employs the solutions outlined in this section to develop the MATI architecture and its instruction set.
5.2.2 MATI Architecture

The MATI architecture shown in Fig. 5.3 employs a 1GHz main clock frequency, the same as [39], to be able to generate control signals with a fine resolution. It preserves the standard SRAM read and write functionality for additional flexibility. MATI's components are described as follows:

1. **Multi-row READ (MR-READ):** This stage corresponds to fetching $W_j$s in Listing 5.1 along with the option of realizing element-wise addition and subtraction during the read process itself. All 256 columns in a word row are accessed simultaneously. An 8-bit word is split across two columns in a word row by storing 4-bit MSB and 4-bit LSB in neighboring columns, to support sub-range reads to enhance linearity. Thus, a 128-element vector of 8-bit elements is read to generate 128 analog $BL$ values per access.

2. **Operand data buffer (ODB):** The operand data buffer (ODB) holds eight 128-element vectors representing the second operand $X$ in Listing 5.1, which can be reused as many times as necessary.

3. **BL processor (BLP) and cross BLP (CBLP):** The BLP stage computes the 128 SDs $d(w, x)$ such as absolute value, compare, square, and multiplication. The CBLP stage aggregates the 128 vector elements by charge-sharing to compute $D(W, X) = \sum_{i=1}^{N} d(x[i], w[i])$.

4. **Sampler and ADC:** MATI has four analog samplers and eight slow but area-efficient single-slope ADCs to convert the CBLP outputs into a digital format for further processing.

5. **Residual digital logic (RDL):** The RDL computes the non-linear functions $f()$ in Listing 5.1 as well as aggregation needed when the vector length $N > 128$. Thus, the RDL consumes negligible energy as it is infrequently activated. Non-linear operations such as sigmoid are implemented in the RDL via piece-wise linear approximation [84].

We describe the MATI instruction set next.
Table 5.4: Instruction set of MATI: (a) instruction format, and (b) operations in each class.

(a) Table Format

<table>
<thead>
<tr>
<th>Class</th>
<th>Operation</th>
<th>Bit length</th>
<th>Opcode</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>none</td>
<td>3 bits</td>
<td>000</td>
<td>B_PRD: B_ADD1 circulates from 0 to B_RPD-1</td>
</tr>
<tr>
<td></td>
<td>write(A_ADD)</td>
<td></td>
<td>001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>read(A_ADD)</td>
<td></td>
<td>010</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mr_read(A_ADD)</td>
<td></td>
<td>011</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sr_read(A_ADD)</td>
<td></td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mr_subt(A_ADD, B_ID1)</td>
<td></td>
<td>101</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mr_add(A_ADD, B_ID1)</td>
<td></td>
<td>110</td>
<td></td>
</tr>
<tr>
<td></td>
<td>avg</td>
<td>4 bits</td>
<td>000</td>
<td>Charge Sharing (C.S) bit:</td>
</tr>
<tr>
<td>Analog</td>
<td></td>
<td>(OPCODE + C.S)</td>
<td>001</td>
<td>0: No charge-share</td>
</tr>
<tr>
<td></td>
<td>abs</td>
<td></td>
<td>010</td>
<td>1: charge-share for reduction</td>
</tr>
<tr>
<td></td>
<td>square</td>
<td></td>
<td>011</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sign_dot(B_ID2)</td>
<td></td>
<td>100</td>
<td>B_PRD: B_ADD2 circulates from 0 to B_RPD-1</td>
</tr>
<tr>
<td></td>
<td>unsign_dot(B_ID2)</td>
<td></td>
<td>101</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADC</td>
<td>1 bit</td>
<td>0</td>
<td>ADC_FREQ, ADC_PREC</td>
</tr>
<tr>
<td></td>
<td>accumulation</td>
<td>3 bits</td>
<td>000</td>
<td>DES, ACC_NUM</td>
</tr>
<tr>
<td></td>
<td>mean</td>
<td></td>
<td>001</td>
<td>DES</td>
</tr>
<tr>
<td>Digital</td>
<td>threshold</td>
<td></td>
<td>010</td>
<td>DES, THRES_VAL</td>
</tr>
<tr>
<td></td>
<td>max</td>
<td></td>
<td>011</td>
<td>DES</td>
</tr>
<tr>
<td></td>
<td>min</td>
<td></td>
<td>100</td>
<td>DES</td>
</tr>
<tr>
<td></td>
<td>sigmoid</td>
<td></td>
<td>101</td>
<td>DES</td>
</tr>
<tr>
<td></td>
<td>ReLu</td>
<td></td>
<td>111</td>
<td>DES</td>
</tr>
<tr>
<td></td>
<td>ADC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Diagram Format

Figure 5.4: Instruction set of MATI: (a) instruction format, and (b) operations in each class.
5.2.3 MATI Instruction Set

The MATI architecture is unconventional, and the ISA reflects the unique features of the architecture. Below, we describe the ISA and justify our design choices.

5.2.3.1 Fields of Instruction Set

The key features of the MATI instruction set (Fig. 5.4) are as follows:

1. **Tasks**: A MATI program consists of one or more tasks, or macro-instructions. A single task consists of sequential operations from five Classes 0-4, described below. Classes 1-3 perform the distance computation, \( D(W,X) = \sum_{i=1}^{N} d(x[i], w[i]) \), of Listing 5.1 whereas Class 4 completes the \( f(D(W,X)) \). A single algorithm may sometimes require several tasks for different distance metrics, e.g., linear regression requires three tasks, as explained in Section 5.3. MATI can process multiple different tasks sequentially.

2. **Classes**: Each MATI task is partitioned into six fields: Classes 0-4 and a repetition count, \( RPT\_NUM \). Class 0 specifies parameters to control the behavior of Classes 1-4, including configuration parameters (e.g., BL voltage swing \( \Delta V_{BL} \), ADC control, Class 4 threshold), memory addresses, and other operand specifiers. The Class 0 parameters are key to achieving flexible programmability in the MATI architecture.

Classes 1-4 execute sequentially and specify the behavior of the four hardware mechanisms: BCA access and optional element-wise addition/subtraction (Class 1), BLP and CBLP (Class 2), optional analog-to-digital conversion (Class 3), and a residual digital operation (Class 4). An operation for each class can be chosen out of available options listed in Fig. 5.4(b). MATI's hardware supports a nearly arbitrary combination of operations from each class. There are >120 possible combinations, but the hardware imposes one restriction: the ODB can provide an operand to only one of either Class 1 or Class 2 in each Task. This restriction is irrelevant in practice as the distance computation, \( D(W,X) \) has just two operands: the first operand comes from Class 1, and the second operand \( X \) can come from either Class 1 or Class 2, but not both.
3. **RPT_NUM**: This parameter specifies the iteration count for the outer FOR loop of Listing 5.1, i.e., how many times the task should be executed. The addresses for the bitcell array \((A_{ADD})\) and ODB \((B_{ADD1} and B_{ADD2})\) are incremented sequentially for each iteration. Although unconventional for modern RISC architectures, this is a natural choice for the typical loop structure of MATI algorithms, which iterate sequentially through data \((w[i])\) in memory to perform the distance computation, \(\sum_{i=1}^{N} d(w[i], x[i])\). This choice keeps the ISA simple by eliminating explicit operand specifiers because that generality is unnecessary.

A more conventional ISA might define the operations as separate instructions, and (usually) express the **RPT_NUM** iterations using explicit control flow. We chose the combined “macro” representation because *all* MATI computations are straight-line code (i.e., a non-branching loop body) executing for a number of iterations that can be computed before entering the loop. An entire such loop can be mapped to a single MATI task (one or more tasks per kernel), with no loss of flexibility and no extra complexity in a code generator. This is unlike a conventional general-purpose architecture executing arbitrary combinations of operations, often including control flow, where mapping to a single complex instruction would incur far more complexity during the instruction selection.

Two configuration parameters in Class 0 give software the flexibility to trade-off application accuracy for energy and latency. **ADC_PREC** specifies the bit precision for ADC output, which depends on the accuracy required by the algorithm. **SWING** controls WL voltage level \((V_{WL})\), which controls BL swing \(\Delta V_{BL}\) and directly impacts both the accuracy and the energy. (Section 5.4 evaluates this accuracy-energy trade-off for several algorithms.) We envisage that compiler or autotuner techniques can be used to map high-level application metrics (e.g., accuracy of decisions) to these hardware-level parameters; this is a subject of our ongoing work.

### 5.2.3.2 Application Example

We present an example of template matching with the L1 distance kernel to find the closest 128-pixel image to input query image \((X)\) out of 64 candidate images \((W_j)\)s. The template
matching is mathematically defined as

\[ j_{opt} = \arg \min_{j} \sum_{i=1}^{128} |x[i] - w[j, i]| \]  
\[ (5.1) \]

The instruction of Task includes RPT_NUM, which specifies the number of candidate images, the "MR-READ and subtract" (mr-subt) operation (Class 1); the "absolute computation followed by aggregation (charge sharing)" (Class 2); ADC (Class 3), and finally a digital-domain min operation that computes the \( f(\cdot \cdot) \) operator, \( \arg \min_j \) (Class 4), and parameters for Classes 1-4 (Class 0). Thus, the binary instruction for this application can be expressed by "RPT_NUM:1000000, Classes 1-4: 101 0101 1 100". Akin to a VLIW (a very large instruction word) architecture, a single task (wide-word macro instruction) specifies multiple classes of spatially separable operations to be performed; unlike a VLIW architecture, however, the operations in an instruction are sequential rather than parallel.

5.2.3.3 Dynamic Task Period (DTP)

As explained in Section 5.2.1.3, if \( T_P \) needs to accommodate the worst-case delay of all possible combinations, the idle time will be increased as some of these require less time. This leads to loss in throughput and an accuracy problem due to the leakage from the sampled analog value at the analog latch. Thus, we employ a dynamic task period (DTP) (Fig. 5.5) to apply optimal \( T_{Pi} \) per given \( i \)-th task as follows:

\[ T_{Pi} = \max(T_{i,\text{Class1}}, T_{i,\text{Class2}}) \]  
\[ (5.2) \]

where \( T_{i,\text{Class1,2}} \) is a delay of Class 1 or 2 in \( i \)-th task. Note that the multiple ADC operates in parallel with other blocks’ operations due to its high latency (unlike the conceptual diagram in Fig. 5.2). The delay of Class 4 is negligible as it is a simple scalar computation. Thus, Classes 3 and 4 are excluded in (5.2).

As shown in Fig. 5.3, the \( T_{Pi} \) is calculated in instruction register, which stores the delay table per operation. The MAIN the CTRLer enables Class CTRLers with the period of \( T_{Pi} \). The Class 4 CTRLer is not synced by \( T_{Pi} \) as its delay is negligibly small, and thus it starts
Figure 5.5: Timing diagram for MATI with DTP.

Listing 5.2: C++ interface for template matching (L1 distance).

```c
/* X: template query image
 * W[j]: j’th candidate image
 * W.size() = ROW_NUM(W) x COL_NUM(W)
 * minVal, maxLoc: outputs of temp. match.
 * Best match is W[minLoc].
 */
templateMatching_L1(W, X, &minVal, &minLoc);
```

as soon as the ADC operation is completed (Fig. 5.5).

5.2.3.4 ISA to Program Mapping

We have built a library to map programmer-written C++ code for each of the benchmarks in Section 5.3.3. More specifically, each kernel of interest is written in a MATI assembly language we have defined, and is wrapped into a library function that can be called from C or C++. This allows the application programmer to simply use the library call in C or C++ code without being exposed to hardware details. For example, the following code excerpt shows the C++ code for calling the template matching library operation (after setting up the appropriate data structures).

The library performs the following steps:

1. Generate Class 0 instruction: Class 0 and RPT_NUM are dependent on the algorithm and input data, and are thus generated at runtime. For example, RPT_NUM
2. Generate Class 1-4 opcodes: Classes 1-4 opcodes for each benchmark kernel are independent of the input data: they essentially encode the algorithm. For now, these have been hand-coded in the library.

3. Generate MATI kernel: After generating Class 0, RPT_NUM and Classes 1-4 instructions, the library concatenates them and generates the MATI ISA representation of the application kernel. This is then shipped to the MATI simulator. This is equivalent to the host processor constructing the MATI kernel code after computing the Class 0 fields and then transferring the kernel code to the MATI accelerator for execution.

For the present, MATI kernels must be written as assembly code by hand. A full-fledged compiler back-end for MATI is under development.

5.2.4 Extension to Large-Scale Applications

MATI is well suited to process 128-dimensional vector processing. Longer vectors can be processed by repeating the 128-dimensional vector processing sequentially by setting $RPT\_NUM = (\text{w.size()}/128)$ and other parameters as shown in Listing 5.2.

The multi-bank configuration can be employed to process long vectors in parallel by distributing the vector elements in many banks (e.g., neighboring four banks). The multi-bank also gives a potential energy benefit by sharing one CTRLer over multiple banks amortizing the CTRLer energy overhead. This mode is employed by assigning five more bits in the instruction set: one bit turns on the multi-bank mode; the other four bits define which bank is turned on.

Even larger and more complex applications such as AlexNet require frequent DRAM accesses. However, several prior works [26, 85] minimized off-chip DRAM access by extensive data reuse and achieved parallelization by many processing elements including a local memory. More specifically, it was shown that the local memory (SRAM) access and processing energy left as the dominant portion after applying the data reuse techniques [26]. In this
case, MATI has strong potential to improve system energy efficiency and throughput by replacing those processing elements and local memory.

5.3 Validation Methodology

This section describes our methodology for validating MATI’s energy, delay, and accuracy benefits. Validating MATI’s benefits is made challenging by its intrinsic analog mixed-signal nature and by the fact that ML algorithms need to process large data sets in order to obtain application-level accuracy. The key challenge lies in estimating the application-level metrics, specifically accuracy, from component-level metrics in an efficient manner.

Our methodology shown in Fig. 5.6 addresses these challenges by: (1) developing silicon-validated energy, delay, and behavioral models of MATI components in a TSMC 65 nm GP process including analog non-idealities, (2) incorporating the delay and behavioral models to develop the timing and behavior accurate component-level Verilog-A models, (3) incorporating these component-level Verilog models into a timing and behavior accurate architectural-level MATI Verilog model (this model is employed to ensure correct function-
Table 5.2: Energy and delay per operation (1 cycle = 1 ns).

<table>
<thead>
<tr>
<th>Class</th>
<th>Operation</th>
<th>Delay (# of cycles)</th>
<th>Energy/Bank (pJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>write</td>
<td>2</td>
<td>72.6</td>
</tr>
<tr>
<td></td>
<td>read</td>
<td>2</td>
<td>33.4</td>
</tr>
<tr>
<td></td>
<td>mr_read</td>
<td>7</td>
<td>61.4</td>
</tr>
<tr>
<td></td>
<td>sr_read</td>
<td>5</td>
<td>22.6</td>
</tr>
<tr>
<td></td>
<td>mr_subt</td>
<td>7</td>
<td>103.3</td>
</tr>
<tr>
<td></td>
<td>mr_add</td>
<td>7</td>
<td>103.3</td>
</tr>
<tr>
<td>2</td>
<td>avg</td>
<td>6</td>
<td>5.2</td>
</tr>
<tr>
<td></td>
<td>abs</td>
<td>6</td>
<td>12.1</td>
</tr>
<tr>
<td></td>
<td>square</td>
<td>8</td>
<td>37.9</td>
</tr>
<tr>
<td></td>
<td>sign_dot</td>
<td>14</td>
<td>16.4</td>
</tr>
<tr>
<td></td>
<td>unsign_dot</td>
<td>14</td>
<td>16.4</td>
</tr>
<tr>
<td>3</td>
<td>adc</td>
<td>$2^N+10$</td>
<td>5.7</td>
</tr>
<tr>
<td>4</td>
<td>accumulation</td>
<td>4</td>
<td>≈ 0</td>
</tr>
<tr>
<td></td>
<td>mean</td>
<td>3</td>
<td>≈ 0</td>
</tr>
<tr>
<td></td>
<td>threshold</td>
<td>2</td>
<td>≈ 0</td>
</tr>
<tr>
<td></td>
<td>max</td>
<td>4</td>
<td>≈ 0</td>
</tr>
<tr>
<td></td>
<td>min</td>
<td>4</td>
<td>≈ 0</td>
</tr>
<tr>
<td></td>
<td>sigmoid</td>
<td>3</td>
<td>≈ 0</td>
</tr>
<tr>
<td></td>
<td>ReLu</td>
<td>3</td>
<td>≈ 0</td>
</tr>
<tr>
<td></td>
<td>Leakage energy per cycle (1 ns)</td>
<td>0.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CTRLer energy per cycle (1 ns)</td>
<td>5.4</td>
<td></td>
</tr>
</tbody>
</table>

...ality and estimate accuracy over small data sets), and (4) developing a MATI C++ model incorporating component-level behavioral models for verifying accuracy over large data sets.

5.3.1 Component-Level Models

MATI comprises both analog (BCA, BLP, CBLP, sampler, ADC) and digital components (CTRLer, RDL). The entire analog chain was post-layout simulated in SPICE in TSMC 65 nm GP process to obtain the energy and delay values as shown in Table 5.2. The total energy and delay for the analog blocks were compared with the measured results reported in [39] and the differences were found to be within 10% and 9%, respectively, thereby validating the energy and delay numbers shown in Table 5.2. In order to capture the behavior of the analog components in the presence of analog non-idealities, we conducted Monte-Carlo SPICE simulations of all the analog components. For example, the variation of the MR-READ process due to transistor threshold voltage mismatch was obtained in this manner.
The impact of charge injection and coupling noise in MR-READ, BLP, and CBLP stages were captured via post-layout simulations. Behavioral models incorporating these non-ideal analog effects were extracted from SPICE data in the form of look-up tables (LUTs). These behavioral LUTs and delays from Table 5.2 were then incorporated into timing and behavior accurate component-level Verilog-A models.

Verilog models of all the digital components including CTRLer and RDL bank were developed and synthesized with the TSMC 65 nm GP library via the Synopsys Design Compiler. Cadence Encounter was used to place and route the synthesized netlist. Gate-level Verilog simulations were performed using the Standard Delay Format (SDF) file incorporating post-layout delay information. Finally, the Value Change Dump (VCD) file obtained from Verilog simulations was employed to obtain energy estimates through the Cadence Encounter.
5.3.2 Application-Level Validation

Verilog and Verilog-A models described in Section 5.3.1 were integrated to obtain a cycle and functionally accurate MATI Verilog model. This model was executed on small data sets, e.g., $RPT_{NUM} < 10$, to ensure that: (1) operations in all the classes are completed within the period $T_{Pi}$, (2) the digital blocks generate the correct CTRL signals at the right time in the presence of post-layout parasitics when presented with the appropriate MATI instruction word, and (3) inter-block interfaces including digital CTRLers-to-analog and analog-to-analog are correctly synchronized.

In addition, a functional MATI C++ model incorporating the LUT-based analog behavioral models described in Section 5.3.1 was also developed. This C++ model was run on a large data sets to obtain MATI's application-level accuracy, and to validate the MATI Verilog model by comparing their outputs on a small data sets. Finally, a C++ model of the reference digital architecture (see Section 5.3.3) was developed in order to obtain its application-level accuracy. This model’s output was compared with the MATI C++ model output in order to estimate the impact of MATI’s analog non-idealities.
5.3.3 Reference Architecture

We evaluate the energy, speed-up, and accuracy of MATI in one and four bank scenarios with respect to: (1) the reference digital architecture (Fig. 5.7) with minimum bit precision required per algorithm (in the last column of Table 5.3) (CONV-OPT), (2) CONV with 8-bit precision (CONV-8b), (3) CM [39], and (4) pipe-CM (pipelined version of CM).

The digital architecture (Fig. 5.7) consists of a standard SRAM with the pipelined interface to a fixed-function digital processor. In this chapter, pipe-CM was emulated using the MATI component models. These ML applications are memory-bound (i.e., there is little data reuse of \( W_j \)s during inference, as in many cases each \( W_j \) stored in SRAM is used only once per given input \( X \)). Thus, we use the same number of banks for both MATI and the digital ASIC to provide the same external bandwidth via the conventional SRAM interface (1-bank and 4-banks scenarios with a 512×256 array). Note that MATI can employ more parallel computations, since its in-memory computation is not limited by the external SRAM interface.

On the other hand, the operand \( X \) is stored in the ODB and reused for both architectures. We assume the SRAM is self-timed as doing so minimizes the read access delay [59, 60] of the reference architecture. This delay was found to be 2 ns/64-bit fetch via post-layout simulations of the SRAM. The SRAM read access energy was obtained from post-layout simulations and found to be consistent with that in [39].

5.3.4 Benchmarks

The commonly employed ML algorithms listed in Table 5.3 were mapped to MATI. As MATI is programmable, it employs 8-bit input data to cover diverse applications and algorithms. This value of precision has been employed in many other implementations [5, 56, 57, 86]. For simplicity, \( B_x = B_w \) is assumed in this chapter. Though MATI’s accuracy improves when processing high-dimensional vectors as the aggregation step in CBLP has a noise-averaging effect, we assume a conservatively small vector dimension \( N = 256 \).
5.4 Evaluation Results

MATI executes 128-dimensional vector operation within $T_p$, i.e., its throughput is $f_{\text{MATI}} = 128/T_p$. On the other hand, the energy consumption is estimated as follows:

$$E_{\text{MATI}} = \sum_{i=1}^{4} E_{\text{Class},i} + E_{\text{LEAK}} + E_{\text{CTRL}}$$  \hspace{1cm} (5.3)

where $E_{\text{Class},i}$ is the energy consumed by Class,$i$ instruction, $E_{\text{CTRL}}$ and $E_{\text{LEAK}}$ are the controller and leakage energies, respectively. From Table 5.2, the energies of Classes 1 and 2 dominate. We ignore the CTRLer energy in the reference architecture but include it in MATI’s architecture.

5.4.1 Comparison with CONV

The reference architecture fetches $N_{\text{COL}}/L$-bits per access (access time: $T_{\text{SRAM}}$), and thus the throughput is as follows:

$$f_{\text{CONV}} = \left( N_{\text{COL}}/L \right) \left( 1/T_{\text{SRAM}} \right)$$  \hspace{1cm} (5.4)

where the throughput is limited by the memory access. Figure 5.8(a) shows that MATI provides a speed-up of $1.4 \times$-to-$3.4 \times$ compared to CONV-OPT across the benchmarks for a single bank scenario. MATI’s speed-up is the least for linear regression because it needs to re-access the same SRAM data every task because analog data cannot be stored due to leakage. CONV would store the data in a local register and reuse it. Since increasing the number of banks does not impact the relative speed-up (but improves throughput of all the architectures proportionally), we put the results for the single bank only.

Figure 5.8(b) shows that MATI achieves a $2.5 \times$-to-$4 \times$ energy savings compared to CONV-OPT in the single bank case (up to $5.5 \times$ in the four bank case) thereby leading to an energy-delay product (EDP) improvements ranging from $3.4 \times$-to-$12.6 \times$ compared to CONV-OPT. MATI’s energy savings with respect to CONV-8b is less than CM in [39] for SVM, template matching, $k$-NN and matched filtering. This is because the self-timed CONV-8b in this
Figure 5.8: MATI (with $SWING = 111$) compared to CONV in terms of: (a) speed-up, and (b) energy savings.
Figure 5.9: Energy breakdown of MATI (with SWING = 111 in single bank) with respect to CONV-8b.

chapter (2 ns/64-bits) is faster than the conventional SRAM in [39] (9 ns/64-bits), enabling CONV-8b to go into sleep mode faster, thereby reducing leakage energy. From Fig. 5.9, it is clear that the key reason for MATI's energy efficiency is due to its MR-READ (Class 1) and BLP/CBLP (Class 2) instructions executing in the low-swing analog.

5.4.2 Comparison with CM

We compare MATI with CM and pipe-CM in order to estimate the overhead due to programmability for two algorithms: SVM and template matching (L1). The energy and throughput of CM and pipe-CM are estimated from Table 5.2 and Fig. 5.2. Figure 5.10 shows that both MATI and pipe-CM achieve a speed-up of $3.8 \times$ over CM. The reason why MATI does not suffer from a throughput penalty as compared to pipe-CM is due to its use of DTP assignment. This is clearly seen for template matching where MATI without DTP leads to a loss of $2 \times$ in throughput when compared to pipe-CM. The throughput gain of pipe-CM was found to result in a 5.5% energy savings over CM due to reduced leakage. In spite of its operational diversity, MATI’s energy overhead compared to pipe-CM is $< 0.1\%$ as the BLP in both activates only one operation, and it leads to a 4.5% area overhead as compared to CM, as the BCA in both dominates the area.
Figure 5.10: Speed-up factors of pipe-CM, MATI without DTP, and MATI (with DTP) over CM [39].

Figure 5.11: Accuracy vs. energy trade-off of MATI enabled via the SWING parameter for: (a) $k$-NN, (b) template matching (L1), (c) matched filtering, and (d) DNN. The horizontal dotted line marks the accuracy of CONV-OPT for reference.
5.4.3 Energy vs. Accuracy Trade-off

MATI’s analog processing provides an interesting energy vs. accuracy trade-off at the application level. This is achieved via the SWING parameter and the vector size $N$ in the ISA, as shown in Fig. 5.11. Here, the detection accuracy $p_{\text{det}}$, defined as $p_{\text{det}} = \frac{\text{(# of correct decisions)}}{\text{(total # of decisions)}}$, with energy/word (total energy / number of words processed per decision) is plotted for template matching, $k$-NN, matched filtering, and DNN algorithms. We see that multi-category classification algorithms such as template matching need a higher SWING value, and hence more energy, to achieve the same accuracy. Additionally, accuracy improves for the same SWING value as $N$ increases. This clearly indicates the noise averaging effect of CBLP, and enables one to reduce energy as a function of $N$. Finally, in case of a DNN, the accuracy is less sensitive to SWING as the number of hidden layers increases. This trade-off can be exploited to reduce the voltage swing and obtain energy savings as a function of the network size. In practice, energy-optimal values of SWING and ADC\_PREC can be obtained by employing the training set in a calibration mode. Potentially, this process can be automated by the compiler via auto-tuning [87].

Figure 5.11 also predicts MATI’s accuracy in advanced process technology nodes, where a drop in BL swing has a similar effect on the accuracy of its analog computations as process variations. It shows that MATI’s $p_{\text{det}}$ degrades gracefully for all benchmarks in spite of a 50% drop in BL swing ($\text{SWING} = 111$ to 100).

5.5 Conclusion

This chapter has proposed MATI, a programmable ISA based on the multi-functional DIMA called Compute Memory (CM) [39]. Though, we successfully mapped nine popular ML algorithms across computer vision and deep learning applications on MATI, we believe many more algorithms such as the recurrent neural network, random forest, and sparse distributed memory can be mapped as well with minor modification in the instruction set and supporting blocks. This work also sets the stage for developing compiler support for MATI and for extending MATI to address a diverse set of large-scale applications.
Emerging applications such as in health care, social networks, smart infrastructure, surveillance/monitoring and others leverage the ubiquitous presence of sensing and data storage to generate massive data volumes. These data sets are being subjected to ML techniques to extract informative patterns of interest. The implementation of energy-efficient ML in silicon is made challenging especially as such algorithms require processing of large data volumes as reported in recent IC implementations [4, 20, 88]. While much of work in the area of energy-efficient ML accelerators has focused on digital architectures, this dissertation explores a unique alternative - the deep in-memory architecture (DIMA) - where high energy and latency costs of data movement between processor and memory are addressed by embedding mixed-signal computations deeply into the periphery of the memory core.

6.1 Dissertation Contributions

This dissertation first identifies the common functional flow across a diverse set of ML algorithms and maps it to the sequential flow of the four processing stages in DIMA: (1) functional read (FR) – data access, (2) bitline processing (BLP) – element-wise distance calculations, (3) cross BLP (CBLP) – aggregation of vector elements, and (4) thresholding – decision making. This dissertation also presents a system-level rationale to show the DIMA’s robustness despite low-SNR processing caused by tightly pitch-matched analog processing. This is explained in terms of three principles: delayed decision, non-uniform bit protection, and aggregation.
Design guidelines are provided for key parameters such as pulse width and amplitude for \( WL \) enabling signal in the FR stage, capacitor size for BLP and CBLP stages based on the analysis of various noise sources such as process variation, charge injection, thermal noise, and coupling noise. Design techniques such as sub-ranged read and processing, replica bitcell, and shielding line are also provided for further improvement of accuracy, energy, and throughput.

Energy, delay, and mixed-signal circuit behavioral models are introduced to predict the energy and delay trends and application level's accuracy as a function of major design parameters. The HSPICE simulations in a 65 nm process show that the maximum error of the behavioral models is 4.4% of the dynamic range of CBLP output, but the relative magnitude of outputs was maintained. These models are expected to be sufficiently accurate as the relative vector distance is important in the ML algorithms. The predicted accuracy of pattern matching in terms of probability of detection shows a maximum prediction error of 0.066 with vector dimension 256.

The multi-functional DIMA prototype IC was successfully fabricated with 16 KB SRAM array in a 65 nm process achieving up to \( 31 \times \) (56\( \times \) in multi-bank scenario) smaller energy-delay product with 5.8\( \times \) and 5.4\( \times \) smaller delay and energy, respectively, in four algorithms: support vector machine, template matching, \( k \)-nearest neighbor, and matched filter. The prototype IC of RF also achieves a 3.1\( \times \) energy savings and 2.2\( \times \) speed-up at the same time providing a 6.8\( \times \) lower EDP at the same accuracy of \( > 93\% \) compared to conventional digital architecture, leading to a throughput of 364 K decisions/s and energy efficiency of 19.4 nJ/decision for an eight-class traffic sign recognition problem. Measurement results also show the clear trade-off between the application accuracy and energy consumption by changing the \( BL \) voltage swing \( \Delta V_{BL} \), which is the dominant controlling knob of DIMA’s energy.

The DIMA’s benefits were also proven in more complex ML algorithms. The DIMA-based CNN achieves roughly 5\( \times \) energy saving mostly by the low-power inner product computation and the reduced leakage energy due to high throughput. In conclusion, 24.5\( \times \) smaller EDP is achieved as compared to the conventional system with 0.02\% larger error rate in a 10-class handwritten character recognition problem.
This dissertation also put an effort to co-optimize the algorithm and DIMA’s architecture rather than simply adopting the DIMA platform. As a result, circuit and system simulations in a 65 nm CMOS process show that the DIMA-based SDM reduces energy and delay simultaneously by a factor of up to $25\times$ and $12\times$, respectively, over the conventional SDM architecture in the auto- and hetero-associative modes with negligible accuracy loss ($\leq 0.4\%$).

This dissertation also extended DIMA to a programmable ISA called MATI. Employing silicon-validated energy, delay and behavioral models of deep in-memory components, we demonstrate that MATI is able to realize nine ML benchmarks while incurring negligible overhead in energy ($< 0.1\%$), area (4.5\%), and throughput over fixed-function DIMA [39]. In this process, MATI is able to simultaneously achieve enhancements in both energy ($2.5\times$ to $5.5\times$) and throughput ($1.4\times$ to $3.4\times$) for an overall EDP improvement of up to $12.6\times$ over fixed-function digital architectures.

6.2 Future Work

The DIMA platform has strong future potential as described as follows.

6.2.1 Extension DIMA to Other Memory Technology

In this dissertation, an SRAM has been focused to demonstrate the benefits of the DIMA platform as it is closest to the processor in the memory hierarchy. However, the DIMA paradigm has strong potential for other memory technologies. This is especially true for high-density storage systems such as NAND and NOR flash, where the cost of data movement is significantly high. However, in the storage systems, the pitch-matching constraint is expected to be more severe due to the smaller memory bitcell dimension (e.g., $4F^2$). In addition, those memory systems rely on current-based sensing schemes whereas an SRAM exploits the $BL$ voltage swing. The serially connected NAND array structure presents an additional challenge as the FR mechanism in this dissertation assumes the parallel-connected bitcell array. Therefore, circuit innovations are required for novel FR and BLP stages to overcome the difference from SRAM topology. The FR operation along with multi-level-cell
(MLC)-based NAND flash can be a great opportunity to achieve further throughput benefit. Extensions to emerging memory technologies such as PRAM and RRAM are expected to be the natural next step as those use the difference in the threshold voltage and resistance of bitcells, respectively, similar to the flash memory.

6.2.2 Circuit-Level Techniques for Enhanced Read, Write, and Processing

The DIMA achieves significant energy and throughput benefits during the read operations. This is the optimal choice for the ML inference, where a large number of memory accesses are required whereas the write operation is infrequent. On the other hand, the ML training requires not only reading but also writing to update the coefficients. An energy-efficient write technique is needed to apply the DIMA with on-chip training.

The current FR processes 4 bits per $BL$ and covers 8-bit precision with the sub-range read technique. The sub-range read causes significant (e.g., $2 \times$) throughput degradation, limiting bit precision. New FR mechanism or sub-ranged read is expected to address the limited bit-precision issue. The separate read and write paths of the 8T SRAM bitcell can be an opportunity to open the new FR mechanism.

The DIMA assumes that computations require two operands: (1) digital data stored in the memory and (2) streamed-in digital data in the operand buffer. However, many systems accept the sensory input as analog data and converts it into digital format through expensive analog digital conversion (ADC) process. If the DIMA can accept both analog and digital inputs, it will open broad opportunities for the sensory systems.

Although most ML algorithms require the aggregation step for dimensionality-reduction, there are several exceptions, such as belief propagation-based algorithms [89]. These algorithms will require cascaded BLP stages, where input and output dynamic ranges and accumulated noise sources need to be managed by novel analog circuit techniques.
6.2.3 Error Resiliency

The robustness of DIMA heavily relies on the inherent error resiliency from the aggregation, where random noise sources are effectively averaged out. However, a trade-off between the energy and accuracy was demonstrated by the measurement results in Chapter 3. This indicates that there is a potential to push DIMA’s energy savings further by statistical error compensation techniques. The techniques are particularly required when the number of elements in the aggregation is not large enough or unreliable emerging device technology is employed.

Employing some redundancy in the stored data will be one of the straightforward approaches for the error-resiliency. The redundancy can be applied for the bit-level by storing the MSB bits in multiple locations. Alternatively, the redundancy can be applied to the important features after the feature extraction by principle component analysis (PCA) or non-negative matrix factorization (NMF). The DIMA with an on-chip trainer will be an interesting direction to handle both inter- and intra-chip variations with aggressively low-SNR operations. It will be also beneficial to manage the transient noise sources such as temperature, input statistics, and supply voltage fluctuation.

6.2.4 Architecture

The programmable DIMA, MATI, gives software the flexibility to trade-off application accuracy for latency and energy with the bit precision for ADC output and WL voltage level ($V_{WL}$) to control $BL$ swing, respectively. In this dissertation, energy-optimal values of these hardware-level parameters can be obtained with the training data set in a calibration stage. However, we envisage that compiler or auto tuner techniques can be used to map high-level application metrics (e.g., accuracy of decisions) to the parameters. The compiler-based automation techniques can be extended to optimize the sequence of instructions to minimize the $BL$ discharge and the movement of operands. It will be also an interesting direction to combine the flash and SRAM-based DIMA architectures for synergistic use cases. Further optimized architectures for large neural network such as AlexNet and long short-term memory (LSTM) can be considered as well.
REFERENCES


