An In-Memory VLSI Architecture for Convolutional Neural Networks

Mingu Kang, *Member, IEEE*, Sungmin Lim,
Sujan Gonugondla, *Student Member, IEEE*, and Naresh R. Shanbhag, *Fellow*, *IEEE*

***Abstract*—This paper presents an energy-efficient and high throughput architecture for convolutional neural networks (CNN). Architectural and circuit techniques are proposed to address the dominant energy and delay costs associated with data movement in CNNs. The proposed architecture employs a deep in-memory architecture (DIMA), to embed energy-efficient low swing mixed-signal computations in the periphery of the SRAM bitcell array. An efficient data access pattern and a mixed-signal multiplier are proposed to exploit data reuse opportunities in convolution. Silicon-validated energy, delay, and behavioral models of the proposed architecture are developed and employed to perform large-scale system simulations. System-level simulations using these models show >97% detection accuracy on the MNIST dataset, along with 4.9× and 2.4× improvements in energy efficiency and throughput, respectively, leading to 11.9× reduction in energy-delay product (EDP) as compared to a conventional (SRAM + digital processor) architecture.**

***Index Terms*—Convolutional neural networks (CNN), in-memory computing, machine learning, analog processing, accelerator.**

I. Introduction

Emerging applications such as internet of things (IoT), health care, autonomous driving, and sensor-rich platforms demand local decision making capability using machine learning (ML) algorithms. These applications require real time processing of massive data volumes in limited form factors and tight energy budget to be performed on battery-powered electronics. Among the ML algorithms, convolutional neural networks (CNN) is one of the most widely used ML algorithms due to its state-of-the-art performance exceeding the human performance in cognitive and decision-making tasks [1], [2]. However, CNNs require complex interconnect, massive convolution computations, and access to a large data volume. Therefore, general purpose computing platforms such as CPU/GPUs can not implement CNNs in energy and throughput-efficient manner. As a result, a number of integrated circuit (IC) implementations have been presented recently [3]–[12] to realize energy-efficient CNNs in silicon.

The unique data-flow of CNN offers opportunities for massive data reuse [13], which can be exploited to reduce memory access energy and enhance throughput. The DianNao architecture [14] exploits the data reuse across the fetched tiles of input and output feature maps (FMs) to achieve 21× energy savings. Eyeriss [3], [15] extends the data reuse opportunities across convolutional, filter, and input FMs to demonstrate up to 2.5× energy savings in the realization of AlexNet, a state-of-the-art CNN in its time [1]. In addition to data reuse, low-power digital circuit and architectures

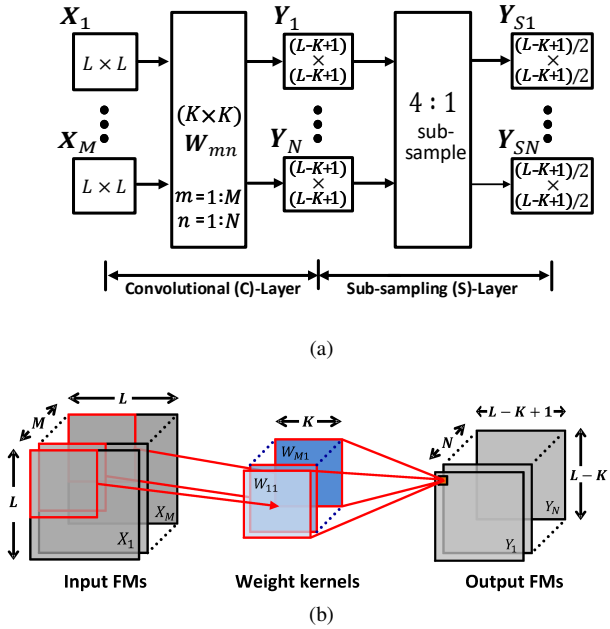


Figure 1: Convolutional neural network (CNN): (a) data flow, and (b) the convolutional (C-) layer.

for DNN implementations such as [9]–[11], [16] have also been proposed. ENVISION [11] employs dynamic voltage-accuracy-frequency scaling given bit-precision requirements. A speech recognizer using deep neural network (DNN) [9] employs a voice-activated power gating technique to achieve energy efficiency during stand-by mode. A sparse DNN engine [10] applied RAZOR technique [17] to minimize the guard band of supply voltage. These implementations [3], [9]–[11], [14], [15] inherit the scalability and programmability benefits of digital implementations while achieving significant energy savings over software realizations on CPU/GPU platforms. However, they miss out on the opportunities afforded by the intrinsic error-tolerance of ML algorithms. In addition, after applying the recent extensive data reuse efforts to minimize the DRAM accesses [3], [15], the energy and delay costs from on-chip data movement and memory access remains dominant. Therefore, a data and memory-driven VLSI implementation is required to address the on-chip data movement costs in a CNN architecture.

This paper employs an alternative approach referred to as the *deep in-memory architecture* (DIMA) [18]–[20]. DIMA embeds mixed-signal computations in the periphery of the

memory bitcell array to minimize the costs of data access and processing. DIMA simultaneously reduces the energy and delay costs due to data movement without losing the storage density and standard read/write functions. DIMA's versatility was demonstrated by mapping ML algorithms such as template matching [18] and sparse distributed memory [21], [22] in simulations. Recently, IC implementations of DIMA [20], [23]–[25] were reported demonstrating its benefits. A multi-functional inference IC in a 65 nm CMOS [20], [26] demonstrates up to $53\times$ reduction in energy-delay product (EDP) for support vector machine, template matching, k-nearest neighbor, and matched filter algorithms. The Random Forest algorithm was also implemented in a 65 nm CMOS [25] demonstrating a $6.8\times$ EDP reduction. Similarly, the AdaBoost algorithm was implemented in [23] using 1-b weight and 5-b input data in a 130 nm CMOS to demonstrate a $113\times$ improved energy efficiency. Additionally, a DIMA IC with on-chip learning overcomes process variations to achieve up to $80\times$ reduction in EDP [24].

This paper employs DIMA to implement energy-efficient and high-throughput CNNs. Preliminary work on DIMA for CNN was presented in [27]. Specifically, the contributions of this paper are: 1) a multi-bank DIMA to implement the LeNet-5 [28] (DIMA-CNN), 2) an efficient data storage format for parallel processing, 3) a mixed-signal multiplier to enable the reuse of functional READ outputs, 4) efficient data movement techniques for input and output FMs, 5) energy, delay, and behavioral models validated in silicon [26] for large-scale system simulations to estimate the application-level benefits, and 6) the use of CNN retraining to minimize the impact from non-ideal analog behavior. The proposed DIMA-CNN architecture is shown to provide up to $4.9\times$ energy saving, a $2.4\times$ throughput improvement, and $11.9\times$ EDP reduction as compared to a conventional architecture (SRAM + digital processor).

The rest of paper is organized as follows. Section II provides the background on CNN and DIMA. Section III describes the proposed DIMA-CNN. Section IV develops silicon-validated behavioral, energy, and delay models for the entire DIMA signal processing chain. Section V presents circuit and system simulation results demonstrating the delay reduction and energy savings of the proposed architecture. Section VI concludes this paper.

II. Background

This section introduces the necessary background on the topics of CNN [28], and DIMA [18]–[20].

A. Convolutional Neural Networks (CNN)

A CNN is a multi-layer network (see Fig. 1(a)) consisting of interleaved convolutional (C-) and sub-sampling (S-) layers, followed by a few fully-connected (F-) layers. The C-layer is computationally intensive and its computation is described as follows:

Table I: List of acronyms.

BCA	bitcell array
BL	bitline
BLB	bitline bar
BLP	bitline processing
CBLP	cross bitline processing
CNN	convolutional neural networks
DIMA	deep in-memory architecture
DNN	deep neural network
FIFO	first-in, first-out
FM	feature map
FM-REG	feature map register
FR	functional READ
I-REG	input register
PWM	pulse-width modulation
RDL	residual digital logic
SA	sense amplifier
WL	wordline

$$\begin{bmatrix} \boldsymbol{Y}_1 \\ \vdots \\ \boldsymbol{Y}_N \end{bmatrix} = \phi \left\{ \begin{bmatrix} \boldsymbol{W}_{11} & \cdots & \boldsymbol{W}_{1M} \\ \vdots & \cdots & \vdots \\ \boldsymbol{W}_{N1} & \cdots & \boldsymbol{W}_{NM} \end{bmatrix} * \begin{bmatrix} \boldsymbol{X}_1 \\ \vdots \\ \boldsymbol{X}_M \end{bmatrix} + \begin{bmatrix} C_1 \\ \vdots \\ C_N \end{bmatrix} \right\} \tag{1}$$

where $\boldsymbol{X}_m$ $(m = 1, ..., M)$ and $\boldsymbol{Y}_n$ $(n = 1, ..., N)$ are the $L \times L$ input FMs and $(L - K + 1) \times (L - K + 1)$ output FMs, respectively, $*$ is the convolution operator, and $\boldsymbol{W}_{nm}$ is a $K \times K$ weight kernel that contributes to the generation of $\boldsymbol{Y}_n$ (the n^{th} output FM) via convolution with $\boldsymbol{X}_m$ (the m^{th} input FM). Here, C_n is a bias term for the n^{th} output FM, and ϕ is a non-linear kernel, typically sigmoid, tanh, or a ReLU function. The S-layer reduces the dimension of input FM via averaging or by finding the maximum of several neighboring pixels. In this work, the S-layer averages four neighboring pixels based on the LeNet-5 network [28].

Figure 1(b) describes the convolutional layer, where $\boldsymbol{Y}_n$ is generated by sliding the $K \times K$ input window (red box) across $\boldsymbol{X}_m$ and summing the output of M such channels. Here, $Y_n(p, q)$ is the (p,q)-th element of $\boldsymbol{Y}_n$, computed as follows:

$$\widetilde{Y}_n(p, q) = \sum_{m=1}^{M} \sum_{i=1}^{K} \sum_{j=1}^{K} W_{nm}(i, j) X_m(p + i, q + j)$$

$$Y_n(p, q) = \phi \left\{ \widetilde{Y}_n(p, q) + C_n \right\} \tag{2}$$

where $W_{nm}(i, j)$ and $X_m(i, j)$ are the $(i, j)^{th}$ elements of $\boldsymbol{W}_{nm}$ and $\boldsymbol{X}_m$, respectively. Since the C-layer requires a large number of read accesses and computations, the proposed DIMA-CNN is highly effective in realizing CNNs.

B. CNN Implementation Challenges

A number of challenges need to be addressed to enable energy-efficient and high-throughput VLSI implementation of CNNs. In the following, we employ VGGNet [29] parameters to illustrate these challenges.

A1) Frequent Data Access and Processing: As seen in (2), large data volumes need to be accessed and processed, e.g., MNK^2 (up to 3×10^5 [29]) bytes of data access and $MNK^2(L-K+1)^2$ (up to 2×10^9 [29]) multiply-and-accumulation (MAC) operations per C-layer. Therefore, solutions for energy-efficient data access and processing are required.

A2) Realizing Sliding Window for Convolution: To realize the sliding window function described in Fig. 1(b), efficient data access is required to feed the matched portion of input FMs $\boldsymbol{X}_m$ into the processor for convolution ($*$) operations.

A3) Enabling Data Reuse in Analog: As indicated in Fig. 1(b), the weight kernel $\boldsymbol{W}_{nm}$ can be reused up to $(L-K+1)^2$ times across the window sliding once it is fetched from memory. Analog-heavy architectures make data reuse challenging as analog computations are vulnerable to various noise sources such as leakage and coupling noise.

A4) Parallelizing Convolutions: As the C-layer requires a large number of MAC operations (up to 2×10^9 [29]), massive parallelism is required to achieve high throughput. Efficient data storage format and architecture to expose and exploit parallelism are challenging.

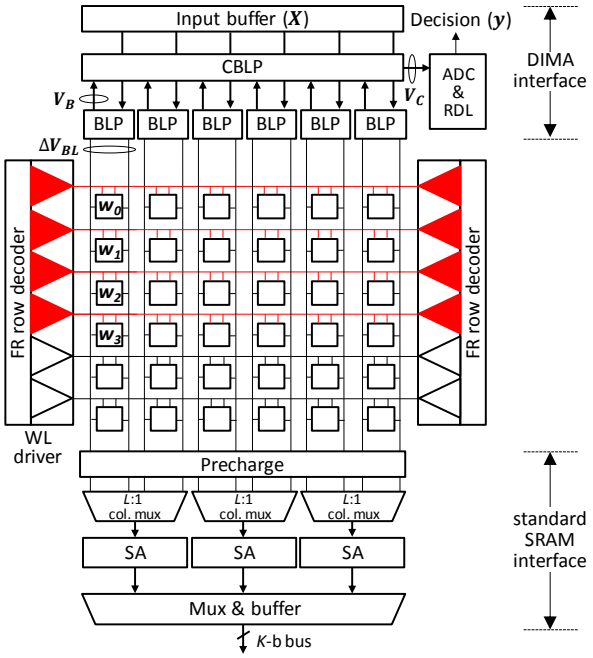
C. The Deep In-memory Architecture (DIMA)

The DIMA [18]–[20] employs a standard $N_{row} \times N_{col}$ SRAM bitcell array and standard read/write circuitry at bottom as shown in Fig. 2(a), and stores the B_W bits of the scalar weight W (m, n, i, j omitted for simplicity) in a column-major format opposed to a row-major format used in the conventional SRAM architecture. DIMA has four sequential processing stages: 1) *multi-row functional READ*: fetches a function of stored B_W bits by accessing B_W rows, so called a *word-row*, per bitline (BL) precharge (read cycle) from each column (Fig. 2(a)) to perform digital-to-analog conversions (D/A), 2) *BL processing*: performs word-level arithmetic operations, e.g., addition, subtraction, or multiplication, by processing the BL voltage level generated via functional READ in the column pitch-matched analog processors in parallel, 3) *cross BL processing*: aggregates the outputs of multiple BL processors via charge-sharing to obtain a scalar output, 4) *ADC* or *thresholding*: generates the digital output (decision) from the previous analog computations.

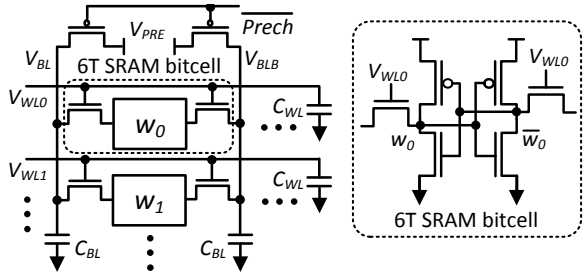
The functional READ stage generates the BL voltage drop $\Delta V_{BL}(W)$ proportional to the weighted sum $W = \Sigma_{i=0}^{B_W-1} 2^i \overline{w}_i$ of the stored data $\{w_0, w_1, ..., w_{B_W-1}\}$ (see Fig. 2(b)). The drop $\Delta V_{BL}(W)$ is generated via simultaneous application of binary-weighted access pulses with widths $T_i \propto 2^i\,(i \in [0, B_W-1])$ to B_W rows (word-row) per BL precharge cycle (see Fig. 2(c)). Provided that $T_i \ll R_{BL}C_{BL}$, where C_{BL} and R_{BL} are the BL capacitance and the discharge path resistance, respectively, $\Delta V_{BL}(W)$ is given by:

$$\Delta V_{BL}(W) = \frac{V_{PRE}}{R_{BL}C_{BL}} T_0 \sum_{i=0}^{B_W-1} 2^i \overline{w_i} = \Delta V_{lsb} \overline{W} \quad (3)$$

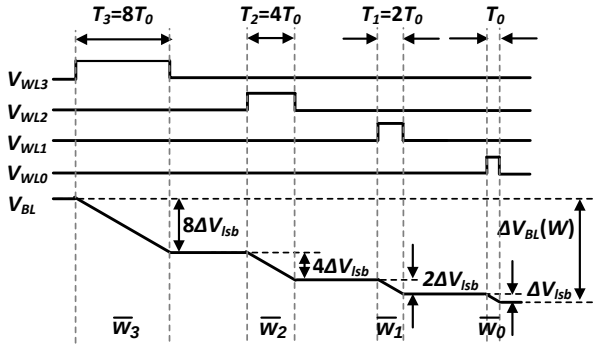
where $\Delta V_{lsb} = \frac{V_{PRE} T_0}{R_{BL} C_{BL}}$, and $\overline{W}$ is the decimal value of the one's complement of W. Similarly, $\Delta V_{BLB}(W)$ can be obtained by replacing $\overline{w_i}$ with w_i in (3). A sub-ranged read



(a)



(b)



(c)

Figure 2: Deep in-memory architecture (DIMA) [18]–[20]: (a) single-bank showing functional READ (FR), bitline processor (BLP), cross BL processor (CBLP) and sense amplifier (SA), where functional READ wordline (WL) drivers shaded in RED are turned on simultaneously, (b) column architecture and bitcell, and (c) idealized functional READ using pulse-width modulated (PWM) WL access pulses during a 4-bit word ($W = 0000b'$) read-out. The WL pulses can be overlapped in time but are shown as being non-overlapped to enhance clarity.

technique [20] can be employed to improve the linearity of the functional READ process when $B_W > 4$ by storing $B_W/2$ LSB bits $\{w_0,\ w_1,...,w_{B_W/2-1}\}$ and $B_W/2$ MSB bits $\{w_{B_W/2},\ w_{B_W/2+1},...,w_{B_W-1}\}$ of a word W in the neighboring columns (column pair) of bitcell array. The MSB and LSB columns are read separately, and then merged by allocating 2^{B-1} times more impact on the MSBs via capacitive charge redistribution.

DIMA overcomes the column muxing requirement of the standard SRAM bitcell array by performing pitch-matched BL computations in parallel. With a sub-ranged functional READ stage, DIMA needs $B_W L/2\times$ fewer precharge cycles to read the same number of bits compared to a standard SRAM, where L is a column muxing ratio in the standard SRAM needed to accommodate large area SAs. This leads to improvements in both energy and throughput. Low voltage swing analog computations in the BL processing improves the energy-efficiency further. However, in exchange, DIMA relaxes the fidelity of its read and computations due to its non-ideal behavior such as non-linearity and noise. The impact of these non-idealities is reduced due to averaging by the charge-sharing operation in the cross BL processing stage. In addition, the CNN algorithm has inherent error resiliency, and circuit-aware retraining minimizes the impact of these non-idealities as shown in Section V.

However, the standard DIMA [20] is inefficient for realizing convolutions because the BL processing that follow functional READ corrupts $\Delta V_{BL}(W)$. In addition, there is no support for data access pattern and data movement required to enable convolutions. Furthermore, previous papers on DIMA [18]–[20] have a single bank whose limited capacity makes it difficult to fully parallelize computations in each layer.

III. Proposed Deep In-memory Architecture for CNN (DIMA-CNN)

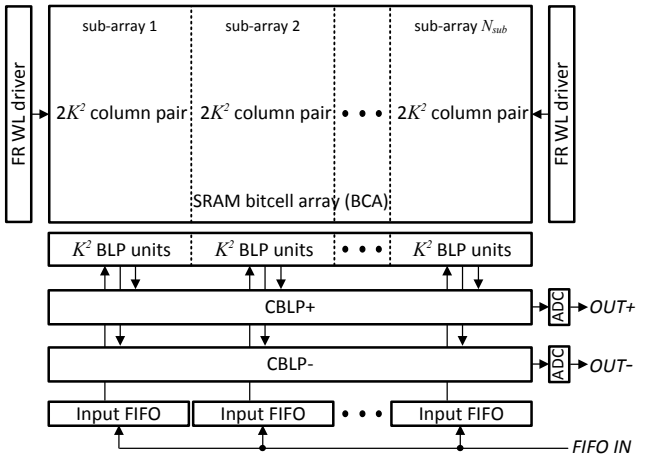
This section proposes the DIMA-CNN to address the implementation challenges described in Section II-B: *A1) frequent data access and processing*, *A2) realizing sliding window for convolution*, *A3) enabling data reuse in analog computation*, and *A4) parallelizing convolutions*. To solve these challenges, the proposed architecture employs:

- DIMA for its high energy efficiency and intrinsic parallelism to address *A1)* and *A4)*,
- a sliding window FM register and optimized FM access patterns to address *A2)*,
- a charge-recycling mixed-signal multiplier enabling the reuse of functional READ outputs to address *A3)*,
- multiple bitcell array banks and an optimized kernel data storage format to address *A4)*.

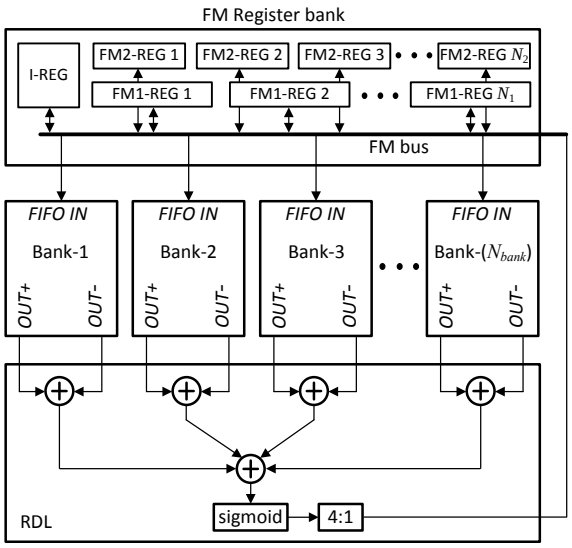
The architecture and operation of DIMA-CNN are explained in following sections.

A. Multi-Bank DIMA-CNN Implementation

Figure 3(a) shows a single-bank DIMA-CNN based on Fig. 2(a) comprising the bitcell array, BL processing, cross BL processing, ADC, and input first-in, first-out (FIFO) registers. The $N_{row} \times N_{col}$ bitcell array is partitioned into $N_{sub} = \lfloor N_{col}/(2K^2) \rfloor$ sub-arrays, each of which has $2K^2$ columns (the factor 2 is from the use of sub-ranged functional READ), to store $\boldsymbol{W}_{nm}$ whereas the input FIFO stores the K^2 pixels of the current input window in $\boldsymbol{X}_m$ (red box in Fig. 1(b)). Figure 3(b) describes the multi-bank DIMA-CNN architecture including N_{bank} banks, the RDL block, and FM register bank.



(a)



(b)

Figure 3: DIMA-based CNN architecture with: (a) single bank, and (b) multiple (N_{bank}) banks, where RDL comprises an accumulator, the sigmoid, and 4:1 sub-sampling blocks.

The output FMs of S-layers are stored in the FM register to avoid the high energy and delay cost of the write operations into the SRAM bitcell array. The FM register also supports the sliding window functionality, which is difficult to realize in a SRAM. The FM register bank includes an input register (I-REG) to store the input image pixels, and N_1 FM1 registers (FM1-REG) and N_2 FM2 registers (FM2-REG) to store the output FMs of the first and second S-layers, respectively. Here, N_1 and N_2 are the number of output FMs of the first and second S-layers, respectively. The output FM of subsequent layers, e.g., the fully-connected layer, can be stored in the FM1-REG by overwriting its contents as the previous layer's

output FMs are not required once the output FM of the current layer is computed.

DIMA's functional READ operation is inherently 1-dimensional (1-D) whereas the convolution operation in (2) is a 2-dimensional (2-D) operation. Therefore, the convolution in (2) is re-formulated as a 1-D operation by transforming the $K \times K$ 2-D kernel $\boldsymbol{W}_{nm}$ into a 1-D vector $\boldsymbol{W}_{1D,nm}$, and the $K \times K$ input window in $\boldsymbol{X}_m$ starting at (p, q)-th pixel into a 1-D vector $\boldsymbol{X}_{1D,m}(p, q)$ as follows:

$$\widetilde{Y}_{1D,n}(p,q) = \sum_{m=1}^{M} \sum_{i=1}^{K^2} W_{1D,nm}(i) X_{1D,m}(p,q,i) \qquad (4)$$

where $W_{1D,nm}(i)$ and $X_{1D,m}(p, q, i)$ are the i^{th} element of vectors $\boldsymbol{W}_{1D,nm}$ and $\boldsymbol{X}_{1D,m}(p, q)$, respectively. The FM register bank provides the input window in $\boldsymbol{X}_m$ starting at (p, q)-th pixel to input FIFOs through the FM bus and *FIFO IN* port so that the FIFO contains the 1-D vector $\boldsymbol{X}_{1D,m}(p, q)$. Each sub-array in Fig. 3(a) processes the inner summation of (4), while N_{sub} such sub-arrays process the outer summation of (4) by aggregating the results from all the sub-arrays in the cross BL processing block via charge-sharing. When $M > N_{sub}$, the outputs of ADCs in each bank are accumulated in RDL (Fig. 3(b)) in the digital domain to avoid the degradation in the analog outputs during its transmission through lossy inter-bank interconnects. When $M \neq N_{sub}$, an appropriate data storage pattern and processing sequence will be presented in Section III-B.

Each BL processing unit multiplies $W_{1D,nm}(i)$ from the functional READ and $X_{1D,m}(p, q, i)$ from the input FIFO, followed by aggregation of the BL processing results in the cross BL processing blocks. The BL processing outputs are charge-shared on either CBLP+ or CBLP- rails in Fig. 3(a), based on the sign of the weight detected by the comparator, which will be shown in Fig. 5.

The RDL (Fig. 3(b)) accumulates the partial sums computed by N_{bank} such banks if needed, and then performs the sigmoid function, which is implemented digitally using a piece-wise linear approximation composed of three additions and two shifts [30]. To perform the S-layer computation (4:1 sub-sampling operation in LeNet-5), four consecutive outputs of the sigmoid processing block, corresponding to four neighboring output FM pixels of C-layer, are averaged. These outputs of the S-layer computation are fed back into the corresponding FM registers through the FM bus to be used as the input FM of the next C-layer.

Note that the proposed architecture realizes the most compute-intensive operation in a CNN, the dot-products in (4), in the analog domain for energy and delay efficiency, and the rest in digital domain, in order to provide maximum flexibility.

B. Efficient Data Storage for Parallel Computations

The data storage format shown in Fig. 4 aims to compute (4) via a single functional READ and BL processing step. The K^2 coefficients of $\boldsymbol{W}_{1D,nm}$ are stored horizontally in a block of $(B_W/2) \times 2K^2$ bitcells (dotted box in Fig. 4), where a B_W-bit word is stored per two columns (column pair) to enable sub-ranged read (Section II-C). Accordingly, a total of $N_{bank}N_{sub}$ sub-arrays can be processed per read access. The $\boldsymbol{W}_{1D,nm}$s with the identical value of index n are horizontally aligned in the same word-row occupying M sub-arrays. If $M < N_{sub}$, the $\boldsymbol{W}_{1D,nm}$s with the same N are accommodated in a word-row of each bank as shown in the 1-st row of Fig. 4. Empty sub-arrays do not contribute to the cross BL processing stage by disabling the switches ϕ_{S_W}s and $\phi_{\overline{S}_W}$s in Fig. 5. On the other hand, if $M > N_{sub}$, the $\boldsymbol{W}_{1D,nm}$s are stored across multiple (up to N_{bank}) banks as shown in 4-th row. If $M > N_{bank}N_{sub}$, the $\boldsymbol{W}_{1D,nm}$s are stored across $\lceil M/(N_{bank}N_{sub}) \rceil$ word-rows.

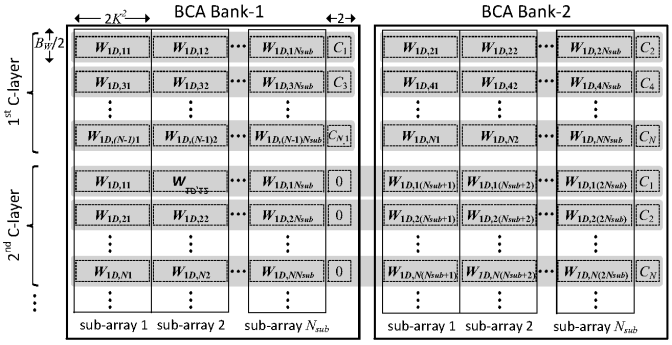


Figure 4: Efficient data storage format for a two-bank (N_{bank} = 2) configuration.

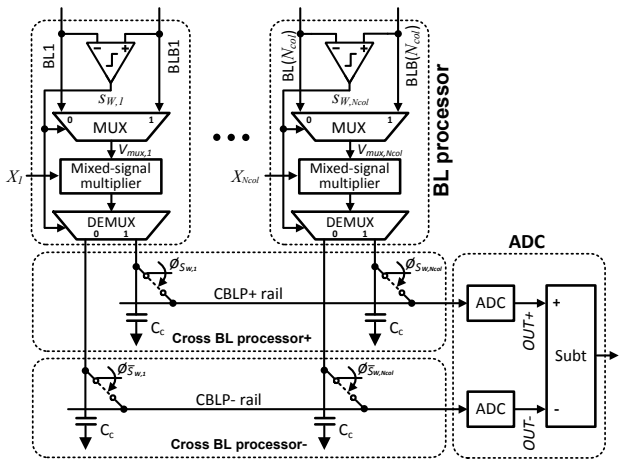


Figure 5: BL processing and cross BL processing architecture for signed dot product.

C. Functional READ, BL Processing, and Cross BL Processing for Signed Dot Product

The CNN computation requires a signed representation for W whereas X is an unsigned number (m, n, i, j omitted for simplicity). One's complement representation is employed to obtain the magnitude ($|W|$) and sign (S_W) of W separately. Figure 5 shows the BL processing and cross BL processing architecture to enable signed number processing. Here, $|W|$ is computed in the functional READ stage by exploiting the complementary nature of SRAM bitcell as follows:

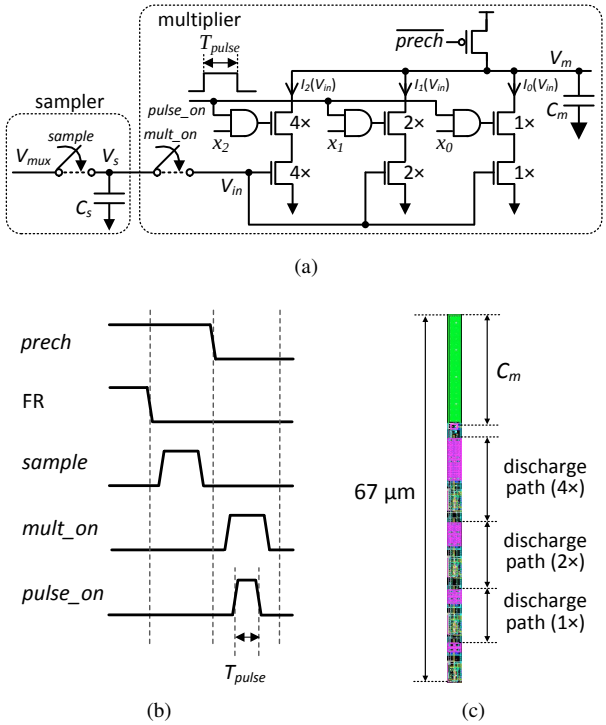


Figure 6: Charge-recycling mixed-signal multiplier to enable data reuse with $B_X = 3$: (a) multiplier circuit diagram with a sampler, (b) timing diagram, and (c) layout fabricated in the 65 nm CMOS prototype IC of [20] as a separate test module.

$$|W| = \begin{cases} \Sigma_{k=0}^{B_W-1} 2^k w_k \propto \Delta V_{BLB}(W), & if\ W \geq 0 \\ \Sigma_{k=0}^{B_W-1} 2^k \overline{w}_k \propto \Delta V_{BL}(W), & if\ W < 0 \end{cases} \quad (5)$$

The sign S_W is obtained by a differential analog comparator with $\Delta V_{BL}(W)$ and $\Delta V_{BLB}(W)$ as its inputs. The sign S_W is used as the select signal into the multiplexer to choose the greater of $V_{BL}(W)$ and $V_{BLB}(W)$. Thus, the output $\Delta V_{mux} \propto |W|$.

The mixed-signal multiplier in the BL processing stage of Fig. 5 operates with two input operands: the analog input V_{mux} (representing $|W|$) and the digital input X from the input FIFO. Next, the multiplier outputs are dumped on the cross BL processing capacitors C_c of either CBLP+ or CBLP- rail via de-multiplexers controlled by the comparator output S_W. The rails are shared with multiple columns so that the absolute values $|W|$ of positive and negative products in (4) can be added separately. Finally, the charge-shared cross BL processing rail voltages is converted into digital through two ADCs, whose outputs positive number *OUT+* and negative *OUT-* are added to generate the convolution sum. These steps need to be repeated for every input window leading to significant energy consumption especially from the functional READ process. To avoid the functional READ iterations, a charge-recycling mixed-signal multiplier is introduced as described in the following section.

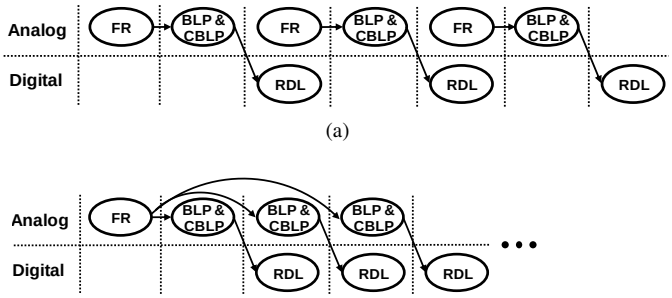


Figure 7: Data flow of DIMA-CNN: (a) without data ($\boldsymbol{W}_{nm}$) reuse, and (b) with data reuse.

D. Charge-recycling Mixed-signal Multiplier

Figure 6 shows the charge-recycling mixed-signal multiplier to compute the product of analog input V_{in} and a B_X(= 3)-bit digital value X. A single unit of the mixed-signal multiplier was fabricated in the prototype IC of [20] as a separate test module to investigate its potential use for convolution operation. This multiplier is designed to be pitch-matched to the horizontal dimension of a bitcell (Fig. 6(c)) to enable column-wise parallelism. In practice, $B_X = 6$ is employed in this paper by cascading two (3-b MSB and 3-b LSB) multipliers in parallel to reduce the delay. Outputs of these multipliers are merged in the cross BL processing stage, and are combined with 8× higher weight given to the MSB outputs [26]. The multiplier first samples the analog input V_{mux} obtained from functional READ stage on a sampling capacitor C_s (Fig. 6(b)). The sampling process isolates the sampled node V_s from the BL/BLB, which has high leakage paths. Therefore, the sampled analog value V_s corresponding to W can be reused up to 200 times with multiple inputs (Xs) across window slidings until V_s drops by 10% due to leakage incurring noticeable degradation in the application-level accuracy as shown in Section V.

The multiplier has B_X pull-down paths corresponding to each bit position. Multiplication begins by disabling the precharge path and enabling the switch $mult\_on$. Each pull-down path discharges the precharged capacitance C_m for the duration of T_{pulse} by enabling the switch *pulse_on* if the corresponding binary data $x_i = 1$. Binary weighted transistor sizing (see Fig. 6(a)) results in the discharge current of the path corresponding to i-th bit position $I_i(V_{in}) = 2^i I_0(V_{in})$, where $I_0(V_{in})$ is the discharge current corresponding to the LSB position ($i = 0$) given by $I_0(V_{in}) = bV_{in} + c$ within the dynamic range of V_{in} = 0.6 V-to-1 V, where b and c are fitting parameters, analyzed in Section IV. Provided that the $T_{pulse} \ll C_m R_m(V_{in}, X)$, where $R_m(V_{in}, X)$ is a discharge path resistance, the output voltage drop ΔV_m is given by:

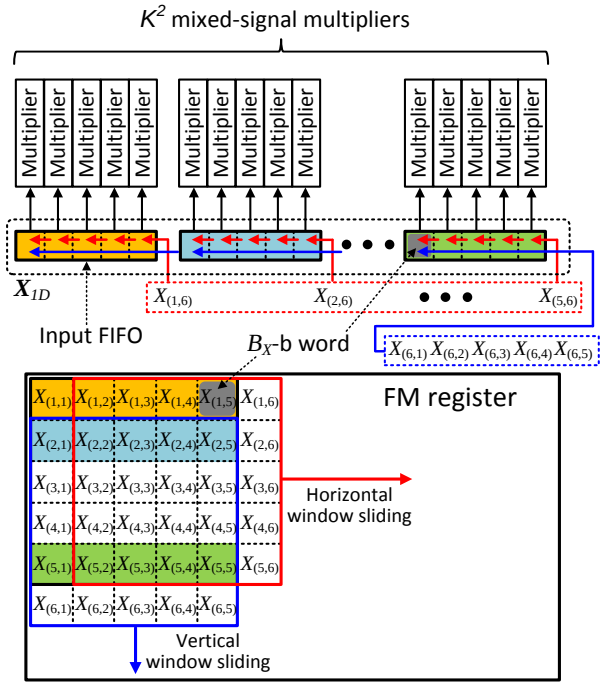


Figure 8: Sliding window FM register and input FIFOs with $K = 5$, e.g., the green-marked right most FIFO contents $\{X_{(5,1)}, X_{(5,2)}, ..., X_{(5,5)}\}$ are replaced by $\{X_{(5,2)}, X_{(5,3)}, ...X_{(5,6)}\}$ after horizontal window sliding or by $\{X_{(6,1)}, X_{(6,2)}, ..., X_{(6,5)}\}$ after vertical sliding.

$$\Delta V_m = \frac{T_{pulse}}{C_m} \sum_{i=0}^{B_X-1} x_i 2^i I_0(V_{in}) = a \sum_{i=0}^{B_X-1} x_i 2^i (bV_{in} + c) = abX(V_{in} + c/b) \quad (6)$$

where $a = T_{pulse}/C_m$. Thus, $\Delta V_m \propto XV_{in}$ with an offset c/b added to V_{in}. The impact of offset is analyzed and overcome by retraining [31] as shown in Section V.

Though the mixed-signal multiplier in [20], [27] does not suffer from offsets, it does not allow the reuse of the functional READ outputs. Thus, the W in the bitcell array needs to be accessed repeatedly for every kernel window sliding as shown in Fig. 7(a). On the other hand, the proposed multiplier (Fig. 6(a)) does not have such limitations enabling the result of functional READ to be reused up to 200 times as shown in Fig. 7(b), thereby minimizing the number of functional READ operations.

E. Sliding Window FM Register

Figure 8 describes data movements in the FM register and input FIFOs during the convolution. Moving the $\boldsymbol{W}_{nm}$s in the bitcell array via the standard SRAM interface (see Fig. 2(a)) is delay and energy inefficient. Alternatively, we provide matching portion of $\boldsymbol{X}_m$ at every move of input window. Each FIFO stores K words, and K such FIFOs align a total of K^2 kernels. When the processing window slides horizontally (shown in red), the FM register provides new words $(X_{(1,6)}, ..., X_{(5,6)})$ to the input of each FIFO by shifting all the FIFO contents to the left. On the other hand, when the window slides vertically (shown in blue), new words $(X_{(6,1)}, ..., X_{(6,5)})$ are input to the K-th (right most) FIFO by shifting it K times to the left while all the other FIFOs' contents are transferred to the left neighboring FIFOs, i.e., the K words in the i-th FIFO are transferred to (i-1)-th FIFO.

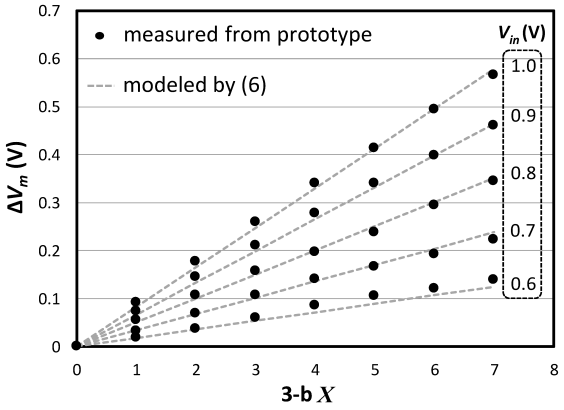


Figure 9: Measured accuracy of the mixed-signal multiplier fabricated in the prototype IC of [20] compared with the model in (6) with $\alpha = ab = 0.16$ and $\beta = c/b = -0.5$.

IV. Modeling and Validation

This section provides behavioral, energy, and delay models for DIMA to be employed in system simulations in Section V. Additionally, the parameters for the models are validated with the measurement of the prototype IC in [20], [26] for the analog blocks, and post-layout simulations for digital blocks, respectively.

A. Behavioral Models with Circuit Non-idealities

DIMA is analog-intensive and therefore subject to a number of circuit-level non-idealities. Dominant among these are:

B1) non-linearity of functional READ process due to V_{BL}-dependent discharge path resistance R_{BL},

B2) impact of process variations on ΔV_{BL} and ΔV_m due to local transistor threshold voltage V_t-mismatch caused primarily by random dopant fluctuations,

B3) input offset of the analog differential comparators,

B4) charge loss on the node V_s due to the leakage current, leading to the inaccuracy on the node V_{in},

B5) mixed-signal multiplier offset (c/b in (6)) due to the non-ideal transistor switching behavior.

The non-linearity of functional READ in *B1)* is modeled by a polynomial fit of measured results in [20], [26] as $\Delta\widetilde{V}_{BL}(W) = \sum_{k=0}^{5} c_k W^k$, where $\Delta\widetilde{V}_{BL}(W)$ is a distorted version of $\Delta V_{BL}(W)$, and c_ks are the fitting parameters. The impact of process variations in *B2)* is modeled as a Gaussian distributed random variable (RV) as shown below:

$$\Delta\widehat{V}_{BL}(W) \sim \mathcal{N}(\Delta\widetilde{V}_{BL}(W), \sigma^2_{BL}(W)) \quad (7)$$

where $\sigma^2_{BL}(W)$ is the variance of $\Delta\widehat{V}_{BL}(W)$. The same modeling method is applied for the analog multiplier with $\sigma^2_m(V_{in})$, the variance of ΔV_m in (6). Monte Carlo SPICE simulations show that $\sigma_{BL}(W)$ and $\sigma_m(V_{in})$ are less than 12.5% and 6.5% of their average values, respectively. Note that the noise due to the variation are well averaged out during the aggregation of 128 BL processing outputs in the cross BL processing stage. The measurement results of prototype IC in [20] confirms this as the standard deviation of cross BL processing output across 128 rows is less than 1.1% of the average value.

The comparator offset in *B3)* is modeled as:

$$V_{mux}(W) \sim max(V_{BL}(W), V_{BLB}(W) + \eta) \tag{8}$$

where η is a zero-mean Gaussian distributed RV with 10 mV standard deviation [18].

The impact of leakage in *B4)* on the node V_{in} as a function of the reuse number r is modeled as follows:

$$\widehat{V}_{in} = V_{in}e^{-\gamma(V_{in})r} \tag{9}$$

$$\simeq V_{in}(1 - \gamma(V_{in})r) \tag{10}$$

where $\gamma(V_{in})$ is the discharge rate of V_{in} after each reuse. Provided that $r \ll 1/\gamma(V_{in})$, the impact of leakage can be approximated as shown in (10). SPICE simulations show that $\gamma < 0.05\,\%$ at the FF corner and 85 °C, where leakage current is maximized. This also includes the kick-back noise from the toggling of switch $mult\_on$ to the node V_s. The parameter r is uniform distributed, i.e., $r \sim \mathrm{U}(1, R)$, where $R \ll 1/\gamma(V_{in})$ is the maximum allowed reuse number.

Figure 9 shows the measured results of the mixed-signal multiplier fabricated in a 65 nm process which is used to model the non-ideality *B5)*. The model parameters in (6) can be written as $\alpha = ab$ and $\beta = c/b$, and fitted to the curve in Fig. 9 with $\alpha = 0.16$, and $\beta = -0.5$. The maximum modeling error is less than 2% of ΔV_m's dynamic range. The offset β results in a deviation from the ideal product $\alpha X V_{in}$, but since it is deterministic, it can be compensated via offline retraining of $\boldsymbol{W}_{nm}$ and C_n in (1).

B. Delay and Energy Models for DIMA-CNN

This section provides the delay and energy models for the conventional and proposed DIMA architectures. We focus on the C-layer though these models can also be applied to the F-layers with slight modification. The conventional system generates N output FMs with a delay given by:

$$T_{conv} \approx \left\lceil \frac{MNK^2}{(B_{IO}/B_W)N_{bank}} \right\rceil T_{read} + \left\lceil MNK^2/N_{mult} \right\rceil N_{mov} T_{mult} \tag{11}$$

where $N_{mov} = (L - K + 1)^2$ is the number of window slidings across input FM, B_{IO} is the bit width of SRAM IO so that B_{IO}/B_W (e.g., 2-to-8) words are fetched per read access per bank, N_{mult} is the number of digital multipliers, and T_{read} and T_{mult} are the delays per the memory access and multiplication, respectively. The delay and energy for the aggregation via an adder tree is assumed to be negligible. The first and second terms in (11) correspond to the delays of SRAM read and subsequent convolution operations. Similarly, DIMA-CNN generates N output FMs with a delay given by:

$$T_{DIMA} \approx \left\lceil \frac{MNK^2}{(N_{bank}N_{col}/2)} \right\rceil \times \left(\lceil N_{mov}/R \rceil T_{FR} + N_{mov} T_{BLP} \right) \tag{12}$$

where T_{FR} and T_{BLP} are the delays of functional READ and BL processing stages, respectively. DIMA can access and process $N_{col}/2$ (e.g., 128) words per bank in parallel, where factor 2 in the denominator arises from the use of sub-ranged read (see Section II-C). This parallelism creates a significant throughput enhancement as seen by comparing the first terms in (12) and (11).

The energy consumption of the conventional architecture to generate N output FMs is given by:

$$E_{conv} \approx MNK^2 E_{read} + MNN_{mov}E_{reg} + MNK^2 N_{mov} E_{mult} + P_{leak} T_{conv} \tag{13}$$

where E_{read}, E_{mult} and P_{leak} represent the single word SRAM read energy, digital multiplier energy, and SRAM leakage power consumption, respectively. It is assumed that a deep-sleep mode is enabled to minimize the leakage during standby using techniques such as power gating or lowering the supply voltage for the bitcell array [32]. Here, E_{reg} is the register energy to: 1) fetch five words per window sliding from the FM register, 2) shift the contents of input FIFOs per window sliding in Fig. 8, and 3) write back a word of the output FM into the FM register after four such window shifting and sub-sampling. Similarly, the energy consumption of DIMA is given by

$$E_{DIMA} \approx MNK^2 \lceil N_{mov}/R \rceil E_{FR} + MNN_{mov}E_{reg} + MNK^2 N_{mov} E_{BLP} + P_{leak} T_{DIMA} \tag{14}$$

where E_{FR} and E_{BLP} are the energy consumed for the functional READ and BL processing operations of single word, respectively. These are significantly smaller (roughly $10\times$) than E_{read} and E_{mult} due to the DIMA's read and processing with low-voltage swing. These models are employed in Section V to demonstrate the benefits of the proposed architecture.

Table II: Delay, energy, and behavioral model parameters.

Parameter	Values	Parameter	Values
T_{FR}, T_{read}	7 ns, 4 ns	T_{BLP}	17 ns
T_{pulse}	2 ns	T_{mult}	4 ns
E_{BLP}, E_{mult}	0.08 pJ, 0.9 pJ	E_{FR}, E_{read}	0.5 pJ, 5.2 pJ
E_{reg}, P_{leak}	4 pJ, 2.4 nW	C_s,C_c,C_m	25, 25, 100 fF
$\sigma_{BL}(W)$	7-12.5% of avg.	$\sigma_m(V_{in})$	2.5-6.5% of avg.
α, β	0.16, -0.5	γ	<0.05%
c_0,..., c_5	-0.04, 0.97, -0.14, 4.7×10^{-2}, -5.3×10^{-3}, 2.5×10^{-4}, -4.3×10^{-6}		

Table III: Design parameters.

Parameter	Values	Parameter	Values
V_{DD}	1 V	V_{PRE}	1 V
input L	32	K	5
B_W, B_X	8, 6	N_{mult}	175
R	0 - 200	B_{IO}	16 - 64
$(N)_{bank,row,col}$	4, 512, 256		
N	C1: 6 , C3: 16, F5: 120, F6: 10		

The parameter values T_{FR} = 7 ns and T_{BLP} = 17 ns in (11) - (14) are validated via the prototype IC in [20] by measuring the functional READ and BL processing accuracies within the given delay. The conventional SRAM read of the prototype IC in [20] takes 9 ns (=T_{read}) per single read access. However, it could be improved if a self-timed controller is employed. Thus, for the conventional system, T_{read} = 4 ns is assumed to be conservative. The digital blocks in the conventional system are also synthesized to have the same delay (T_{mult} = 4 ns) as T_{read} to enable two-stage pipeline between the dot product processing and the SRAM access in F-layers. The values of E_{FR}, E_{read}, E_{BLP}, and P_{leak} are also validated by comparing the measured total energy of prototype IC in [20] and that from the post-layout simulations. Other parameters for digital blocks such as E_{mult} and E_{reg} are estimated from post-layout SPICE simulations of synthesized blocks. The delay, energy, and behavioral model parameters are summarized in Table II.

The F-layers are implemented similarly to the C-layers except that neither the sliding window nor the reuse of functional READ outputs is required. Therefore, the parameters of the F-layer model need to be set accordingly, e.g., $r = 0$ in (10), and $N_{mov} = 1$, $R = 1$ in (11) - (14).

V. Simulation and Results

This section describes the system configuration and simulated results of the proposed DIMA-CNN architecture in terms of its energy, delay, and inference accuracy. As shown in Section IV-A, the accuracy analysis needs to incorporate the stochastic nature of DIMA, thereby requiring a sufficiently large number of simulations. Therefore, we employ LeNet-5 [28] for hand written digit recognition with MNIST database [33].

A. Architecture Configurations

LeNet-5 [28] has six layers: C-layers C1 and C3, F-layers F5 and F6, and S-layers S2 and S4. The design parameters of DIMA-CNN are summarized in Table III. Bit precision of B_W = 8 and B_X = 6 are chosen to obtain negligible accuracy degradation compared to a floating point realization. The DIMA-CNN with 64 KB SRAM ($N_{bank}N_{row}N_{col} = 4 \times 512 \times 256$) and a total of 2 KB registers are employed, where 20 $\boldsymbol{W}_{mn}$s ($N_{col}N_{bank}/(2K^2) \approx 20$ with $K = 5$) are stored per word-row and processed per read access cycle. The FM registers for C-layers are designed to support the vertical and horizontal window movements with kernel size K = 5. The SRAM, BL processing/cross BL processing blocks, and registers take 79.8%, 8.6%, and 11.6% of overall layout area, respectively.

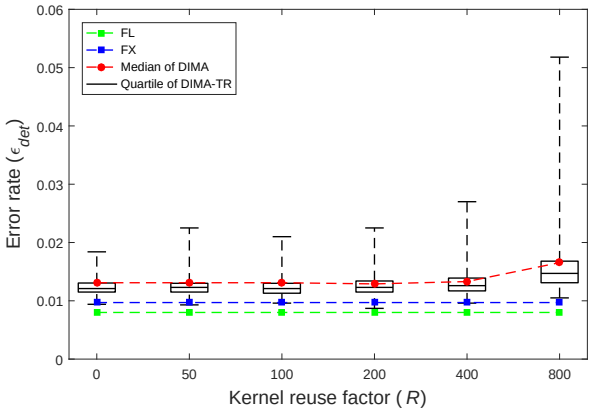


Figure 10: Error rate (ϵ_{det}) vs. reuse factor (R).

We assume the conventional architecture comprises a conventional SRAM instead of functional READ, and digital dot-product computations instead of BL processing and cross BL processing blocks. For the conventional architecture, the IO bit-widths (B_{IO}) per SRAM bank of 16, 32, and 64 bits are considered corresponding to a column mux ratio L = 4, 8, and 16, respectively. A K^2 (= 25) dimensional digital dot product block is synthesized and post-layout simulated in the same (65 nm) process. For fair comparison, we assume that the conventional architecture has seven digital dot product blocks as the area occupied by these blocks is approximately equal to the BLP and CBLP blocks in DIMA-CNN. The DIMA-CNN achieves higher parallelism, e.g., the number of BL processing blocks $N_{col}N_{bank}/2 = 512$, than the conventional architecture with N_{mult} = 175 (= 25×7) within the same area due to the pitch-matched BL processing layout as shown in Fig. 6(c). The energy and delay of the BL processing blocks are estimated via post-layout simulations.

B. Recognition Accuracy

The error rates (ϵ_{det}) are measured on MNIST test dataset [33] via system simulations for the following architectures:

- FL: floating-point simulations.
- FX: the conventional (SRAM+digital processor) architecture with error-free fixed-point computations (B_W and B_X).
- DIMA: DIMA-CNN with fixed-point computations using behavioral models (6) - (9) without resorting to retraining.
- DIMA-TR: DIMA-CNN with fixed-point coefficients retrained accounting for the non-linearity of functional READ process and the offset of charge-recycling mixed-signal multiplier.

Retraining (< 10 iterations) employs the behavioral model of charge-recycling multiplier in (6), and is applied after the standard training process (100 iterations). The ϵ_{det}s of DIMA and DIMA-TR are obtained via Monte Carlo simulations with 400 iterations at each value of R. DIMA and DIMA-TR have the same statistical behavior as DIMA-TR is retrained to compensate the only deterministic non-idealities. Figure 10 plots the ϵ_{det}s of above architectures for different values of

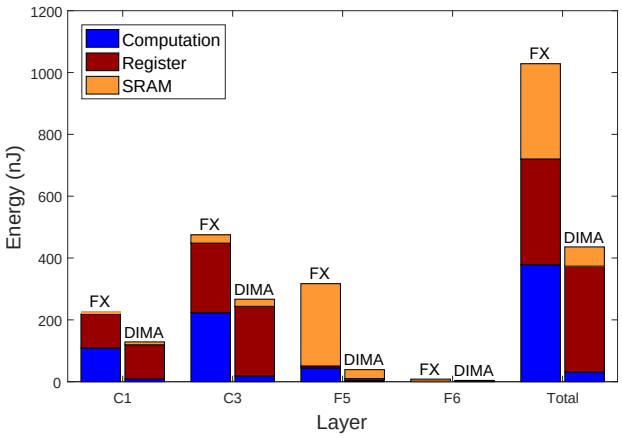


Figure 11: Per layer energy breakdowns with B_{IO} = 16 and R = 50.

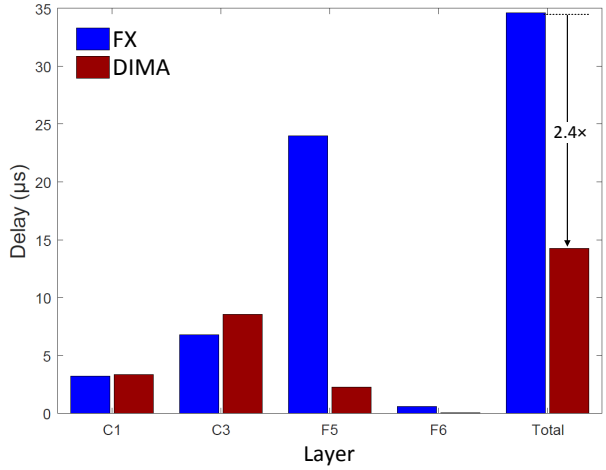


Figure 12: Delay per layer with B_{IO} = 16 and R = 50 for the conventional architecture (FX) and DIMA.

the reuse factors R. The floating-point simulations (FL) and the conventional architecture (FX) achieve a constant $\epsilon_{det} = 0.8\%$ and 0.97% across R, respectively. In contrast, the ϵ_{det} of DIMA increases with R because of the leakage current as described in (9) resulting in a median ϵ_{det} of 1.7% with $R = 800$. However, the median ϵ_{det} of DIMA-TR reduces by up to 0.2% due to retraining. The ϵ_{det} of DIMA-TR increases negligibly with a median $\epsilon_{det} < 1.3\%$ (worst case $\epsilon_{det} < 2.3\%$) for $R < 200$.

C. Energy and Delay

The energy and delay of the conventional architecture and DIMA-CNN are estimated based on (11) - (14). Figure 11 shows the energy breakdowns across the C- and F-layers except the S-layers, which has a negligible contribution to the total energy and delay. The DIMA-CNN architecture saves energy in every layer due to the low-swing computation in functional READ, BL processing, and cross BL processing. Specifically, DIMA significantly reduces the energy in F5 layer, where the highest data access volume is required compared to the other layers. The overall system energy is reduced

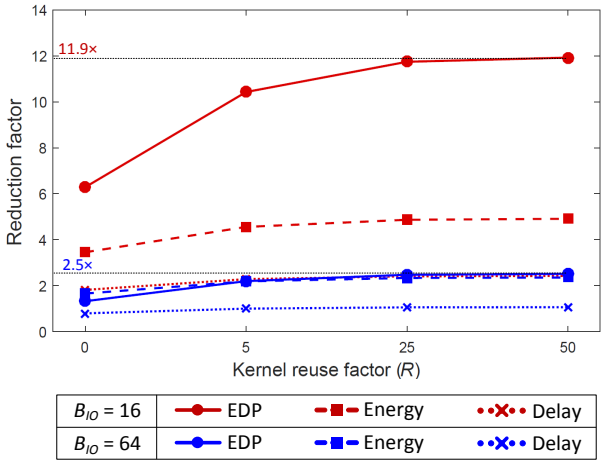


Figure 13: Energy, delay, and EDP reductions over the conventional architecture (FX).

by 4.9× when compared to the conventional architecture, achieving an energy efficiency of 436 nJ/decision.

As shown in Fig. 12, DIMA-CNN has similar or slightly higher delay in C-layers than that of the conventional architecture since the proposed mixed-signal multiplier is slower than digital multiplication ($T_{BLP} > T_{mult}$). However, DIMA-CNN achieves significant throughput improvement in the memory-intensive F5 layer due to the highly parallel functional READ process. Overall, DIMA-CNN achieves approximately a 2.4× delay reduction at a decision latency of 14.3 μs/decision.

Figure 13 shows that both energy and delay benefits over the conventional architecture increase with the R achieving up to 11.9× EDP reduction. The improvement in EDP saturates at R = 50 as the contribution of functional READ to the total energy and delay becomes negligible when $R > 50$. As B_{IO} increases, the energy and delay of the conventional architecture are reduced due to the enhanced memory bandwidth and smaller column mux ratio. Although the delay reduction is negligible with B_{IO} = 64, DIMA-CNN still achieves a 2.4× energy improvement. Overall results indicate that DIMA-CNN achieves a 2.5×-to-11.9× EDP reduction with $R = 50$ as compared to the conventional architecture with less than 1.3% degradation in recognition accuracy. This EDP improvement is less than the reported benefit 24.5× in [27] as practical hardware implementation constraints have been considered here.

As listed in a Table IV, DIMA-CNN achieves at least 4× smaller EDP compared to prior DNN implementations with comparable accuracy, e.g., >97%, for the MNIST dataset.

VI. Conclusion and Future Work

This paper employs the DIMA [18]–[20], an analog deep in-memory computing architecture, to implement an energy-efficient and high throughput VLSI architecture for the CNN algorithm. Simulation results with energy and delay models validated from the prototype IC measurements in [20] demonstrate up to 11.9× EDP reduction with 4.9× energy and 2.4× delay reductions, respectively, with negligible accuracy loss

Table IV: Comparison with other DNN implementations on the MNIST dataset.

Related works	Process (nm)	Network type	Throughput (K decisions/s)	Energy (nJ/decision)	EDP (pJ·s/decision)	Decision accuracy
[10]	28	5-layer perceptron	15	360	24	98%
[16]	65	LeNet-5	13	2463	184	>98%
[34]	65	Spiking Neural Network	1650	162	0.1	90%
This work	65	LeNet-5	70	450	6	>97% (median: 99%)

<1.3% as compared to a conventional (SRAM + fixed-point digital processor) architecture.

Mapping to larger networks such as AlexNet [1] and VGGNet [29] is a natural future direction. As an example, Eyeriss [3], [15], which is a VLSI implementation of AlexNet [1] in a 65 nm CMOS process, reduces the DRAM access energy to only 25% of the overall energy via extensive data reuse techniques, leaving 60% and 15% for on-chip data movement and computation, respectively. By replacing the 36 processing elements (PEs) of Eyeriss with a bank of DIMA-CNNs of equivalent SRAM and register capacity, we estimate that the reported EDP in [15] can be improved further by approximately 2.8×.

Though this paper focuses on ML inferences, an on-chip trainer will be a potential solution to handle both inter- and intra-chip variations of DIMA with aggressively low-SNR operations. Extensions to programmable accelerators and architectures based on emerging memory topologies can be also considered.

ACKNOWLEDGMENTS

This work was supported by Systems On Nanoscale Information fabriCs (SONIC), one of the six SRC STARnet Centers, sponsored by SRC and DARPA.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, Dec. 2012, pp. 1097–1105.

[2] D. Silver, A. Huang, and et al., "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016.

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

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

[5] S. Park, K. Bong, D. Shin, J. Lee, S. Choi, and H.-J. Yoo, "A 1.93 TOPS/W scalable deep learning/inference processor with tetra-parallel MIMD architecture for big-data applications," in *IEEE International Solid-State Circuits Conference-(ISSCC) Digest of Technical Papers*, Feb. 2015, pp. 1–3.

[6] K. Kim, S. Lee, J.-Y. Kim, M. Kim, and H.-J. Yoo, "A 125 GOPS 583 mW network-on-chip based parallel processor with bio-inspired visual attention engine," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 136–147, 2009.

[7] J.-Y. Kim, M. Kim, S. Lee, J. Oh, K. Kim, and H.-J. Yoo, "A 201.4 GOPS 496 mW real-time multi-object recognition processor with bio-inspired neural perception engine," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 1, pp. 32–45, 2010.

[8] J. Oh, G. Kim, B.-G. Nam, and H.-J. Yoo, "A 57 mW 12.5 μJ/Epoch embedded mixed-mode neuro-fuzzy processor for mobile real-time object recognition," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 11, pp. 2894–2907, 2013.

[9] M. Price, J. Glass, and A. P. Chandrakasan, "A scalable speech recognizer with deep-neural-network acoustic models and voice-activated power gating," in *Solid-State Circuits Conference (ISSCC), 2017 IEEE International*, 2017, pp. 244–245.

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

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

[12] K. Bong, S. Choi, C. Kim, S. Kang, Y. Kim, and H.-J. Yoo, "A 0.62 mw ultra-low-power convolutional-neural-network face-recognition processor and a CIS integrated with always-on haar-like face detector," in *Solid-State Circuits Conference (ISSCC), 2017 IEEE International*, 2017, pp. 248–249.

[13] B. Murmann, D. Bankman, E. Chai, D. Miyashita, and L. Yang, "Mixed-signal circuits for embedded machine-learning applications," in *Asilomar Conference on Signals, Systems and Computers*, 2015, pp. 1341–1345.

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

[15] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *International Symposium on Computer Architecture (ISCA)*, 2016, pp. 367–379.

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

[17] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner *et al.*, "RAZOR: A low-power pipeline based on circuit-level timing speculation," in *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, 2003, pp. 7–18.

[18] M. Kang, M.-S. Keel, N. R. Shanbhag, S. Eilert, and K. Curewitz, "An energy-efficient VLSI architecture for pattern recognition via deep embedding of computation in SRAM," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 8326–8330.

[19] N. Shanbhag, M. Kang, and M.-S. Keel, "Compute memory," Jul. 4 2017, US Patent 9,697,877 B2.

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

[21] M. Kang, E. P. Kim, M.-S. Keel, and N. R. Shanbhag, "Energy-efficient and high throughput sparse distributed memory architecture," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, June 2015.

[22] M. Kang and N. R. Shanbhag, "In-memory computing architectures for sparse distributed memory," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 4, pp. 855–863, Aug. 2016.

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

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

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

[26] M. Kang, S. Gonugondla, A. Patil, and N. Shanbhag, "A 481pJ/decision 3.4 M decision/s multifunctional deep in-memory inference processor using standard 6T SRAM array," *arXiv preprint arXiv:1610.07501*, 2016.

[27] M. Kang, S. K. Gonugondla, M.-S. Keel, and N. R. Shanbhag, "An energy-efficient memory-based high-throughput VLSI architecture for convolutional networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2015.

[28] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger *et al.*, "Comparison of learning algorithms for handwritten digit recognition," in *International Conference on Artificial Neural Networks*, vol. 60, 1995, pp. 53–60.

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[30] C. Farabet, C. Poulet, J. Y. Han, and Y. LeCun, "CNP: An FPGA-based processor for convolutional networks," in *IEEE International Conference on Field Programmable Logic and Applications (FPL)*, August 2009, pp. 32–37.

[31] W. Rieutort-Louis, T. Moy, Z. Wang, S. Wagner, J. C. Sturm, and N. Verma, "A large-area image sensing and detection system based on embedded thin-film classifiers," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 1, pp. 281–290, 2016.

[32] M. Yamaoka, Y. Shinozaki, N. Maeda, Y. Shimazaki, K. Kato, S. Shimada, K. Yanagisawa, and K. Osada, "A 300-MHz 25-μA/Mb-leakage on-chip SRAM module featuring process-variation immunity and low-leakage-active mode for mobile-phone application processor," *IEEE J. Solid-State Circuits*, vol. 40, no. 1, pp. 186–194, 2005.

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

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

Mingu Kang (M'13) received the B.S. and M.S. degrees in Electrical and Electronic Engineering from Yonsei University, Seoul, Korea, in 2007 and 2009, respectively, and the Ph.D. degree in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 2017.

From 2009 to 2012, He was with the Memory Division, Samsung Electronics, Hwaseong, South Korea, where he was involved in the circuit and architecture design of Phase Change Memory (PRAM). Since 2017, he has been with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA, where he designs machine learning accelerator architecture. His research interests include low-power integrated circuits, architecture, and system for machine learning, signal processing, and neuromorphic computing.

Sungmin Lim received the B.S. and M.S. degrees in Electrical Engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2014 and 2016, respectively. He is currently pursuing the Ph.D. degree in Electrical and Computer Engineering at University of Illinois at Urbana-Champaign, Champaign, IL, USA. His current research interests include design of energy-efficient architectures and implementation on integrated circuits for machine learning in resource constrained platforms.

Sujan K. Gonugondla (S'16) received the B.Tech. and M.Tech. degrees in Electrical Engineering from Indian Institute of Technology Madras, Chennai, India, in 2014. He is currently pursuing the Ph.D. degree in Electrical and Computer Engineering at the University of Illinois at Urbana- Champaign, Champaign, IL, USA. His current research interest includes low-power integrated circuits specifically algorithm hardware co-design for machine learning systems on resource constrained environments. He is a recipient of the Dr. Ok Kyun Kim Fellowship 2018-19 from the ECE department at UIUC and the ADI Outstanding Student Designer Award 2018.

Naresh R. Shanbhag (F'06) received the Ph.D. degree in Electrical Engineering from the University of Minnesota, Minneapolis, MN, USA, in 1993. From 1993 to 1995, he was with the AT&T Bell Laboratories, Murray Hill, NJ, USA, where he led the design of high-speed transceiver chip-sets for very high-speed digital subscriber line. In 1995, he joined the University of Illinois at Urbana-Champaign, Champaign, IL, USA. He has held visiting faculty appointments at the National Taiwan University, Taipei, Taiwan, in 2007, and at Stanford University, Stanford, CA, USA, in 2014. He is currently the Jack Kilby Professor of Electrical and Computer Engineering with the University of Illinois at Urbana-Champaign. His current research interests include the design of energy-efficient integrated circuits and systems for communications, signal processing, and machine learning. He has authored or co-authored more than 200 publications in this area and holds 13 U.S. patents.

Dr. Shanbhag was a recipient of the National Science Foundation CAREER Award in 1996, the IEEE Circuits and Systems Society Distinguished Lecturership in 1997, the 2010 Richard Newton GSRC Industrial Impact Award, 826 and multiple Best Paper Awards. In 2000, he co-founded and served as the Chief Technology Officer of Intersymbol Communications, Inc., (acquired in 2007 by Finisar Corporation) a semiconductor startup that provided DSP-enhanced mixed-signal ICs for electronic dispersion compensation of OC-192 830 optical links. From 2013 to 2017, he was the founding Director of the Systems On Nanoscale Information fabriCs Center, a 5-year multi-university center funded by DARPA and SRC under the STARnet program.