Deep In-memory Architectures in SRAM: An Analog Approach to Approximate Computing

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

***Abstract*—This paper provides an overview of recently proposed deep in-memory architectures (DIMAs) in SRAM for energy and latency-efficient hardware realization of machine learning (ML) algorithms. DIMA tackles the data movement problem in von Neumann architectures head-on by deeply embedding mixed-signal computations into a conventional memory array. In doing so, it trades-off its computational signal-to-noise ratio (compute SNR) with energy and latency, and therefore represents an analog form of approximate computing. DIMA exploits the inherent error immunity of ML algorithms and SNR budgeting methods to operate its analog circuitry in a low-swing/low compute SNR regime thereby achieving $>100\times$ reduction in the energy-delay product (EDP) over an equivalent von Neumann architecture with no loss in inference accuracy.**

This paper describes DIMA's computational pipeline, provides a Shannon-inspired rationale for its robustness to process, temperature and voltage variations, and design guidelines to manage its analog non-idealities. DIMA's versatility, effectiveness and practicality, demonstrated via multiple silicon IC prototypes in a 65 nm CMOS process is described. A DIMA-based instruction set architecture (ISA) to realize an end-to-end application-to-architecture mapping for the accelerating diverse ML algorithms is also presented. Finally, DIMA's fundamental trade-off between energy and accuracy in the low compute SNR regime is analyzed to determine energy-optimum design parameters.

***Index Terms*—accelerator, in-memory computing, artificial intelligence, machine learning, energy efficiency, non von Neumann.**

I. Introduction

There is growing interest in employing decision making capabilities based on artificial intelligence (AI) algorithms into various sensor-rich platforms at the Edge such as healthcare, internet-of-things (IoT), robots, autonomous driving, and many others. Though deep neural network (DNN)-based machine learning (ML) algorithms have begun to exceed the human capabilities in complex decision-making tasks [1], [2], they require processing of large data volumes. On the other hand, such applications require inferences to be generated under stringent constraints on the form factor, latency, and energy. Therefore, implementing ML algorithms on resource-constrained Edge platforms is an important problem that needs to be addressed.

When subject to severe resource constraints, it is critical that the design of AI computational platforms comprehend the fundamental trade-off between three primary system-level metrics: 1) inference accuracy, 2) latency and throughput, and 3) energy-efficiency. Today, it is well-established that the energy and latency costs of decision-making are dominated by memory accesses, e.g., it takes roughly $20-100\,\mathrm{pJ}$ per

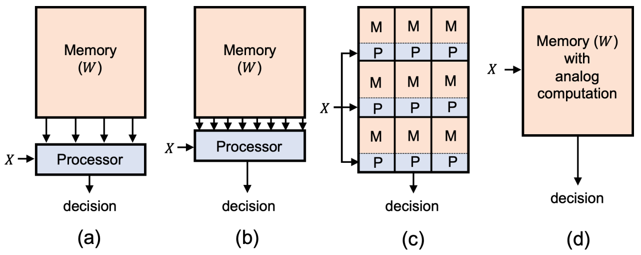


Fig. 1. Inference architectures: (a) the digital (von Neumann) architecture, (b) the near-memory architecture, (c) the logic-in-memory (LIM) architecture, and (d) the *deep* in-memory architecture (DIMA), where W is the weight parameter stored in memory and X is the input.

16-bit word access from a 32 kB - 1 MB SRAM versus $1\,\mathrm{pJ}$ per multiplication in a $45\,\mathrm{nm}$ process [3]. Recent hardware implementations of deep neural networks (DNN) [4], [5] support this assertion reporting 35% and 45% of the total energy cost due to memory accesses.

As a result, an emerging trend in compute architectures for AI is from the traditional von Neumann (digital) architecture [4]–[9] (Fig. 1(a)) towards *near-memory* [10]–[15] (Fig. 1(b)), *logic in-memory* (LIM) (Fig. 1(c)), and *deep in-memory architectures* (DIMAs) [16]–[18] (Fig. 1(d)), whereby the memory access costs are alleviated by bringing computation in close physical proximity to memory. While digital architectures (Fig. 1(a)), still the mainstream architecture today, strive to minimize memory accesses via techniques such as efficient data-flow, data reuse, and computation reduction, near-memory architectures (Fig. 1(b)) distribute computations around memory banks or employ 3D technologies that physically stack memory over traditional CMOS dies. However, like digital architectures, near-memory architectures preserve the intrinsic separation between the memory and processor. LIM architectures (Fig. 1(c)) [19]–[23] employ memory bitcells (BCs) with embedded digital logic to realize useful computations in memory. This approach requires specialized BCs (e.g., 16T in CMOS [19], [20] or 2T-2R [21] / 4T-2R [22], [23] structures in emerging technologies), which prevents exploiting the highly optimized design process for the conventional BC arrays, e.g., layout design rules, and mask layers. In contrast, DIMA (Fig. 1(d)) [16], [24]–[43] embeds analog computations close to the memory bitcell array (BCA) to overcome the memory-processor interface limitations. As compared to an equivalent von Neumann architecture, DIMA implementations have demonstrated more than $100\times$ reduction in energy-delay product (EDP) of decisions via multiple integrated circuit (IC) prototypes [16], [24]–[43].

ML algorithms offer a unique opportunity to trade-off accuracy with decision EDP due to their ability to tolerate computational errors and due to their use of statistical metrics, e.g., misclassification rate. As a result, a number of approximate computing techniques have been proposed [44]–[55] for realizing ML tasks. These include: 1) algorithmic level techniques such as pruning in DNN [44], [56], stochastic or probabilistic computation [51], [52], and incremental refinement [49], [50], 2) computations in reduced precision [45], [46] or dynamic precision adaptation [57], 3) approximate arithmetic components to save gate count [47], [48], [53], [58], and 4) approximate logic synthesis [54], [55]. Most of these techniques have been applied to digital architectures (Fig. 1(a)). In contrast, since DIMA reduces the energy-delay product (EDP) of inference by amortizing the latency and energy costs of a single (intrinsically analog) read cycle over massive numbers of computations, these architectures, much like the above mentioned approximate computing techniques, exhibit an intrinsic trade-off between the accuracy of their computations (via their compute SNR) and energy efficiency. Therefore, DIMA offers a unique approach to approximate computing in the context of AI applications.

While many DIMA architectures have been proposed, in this paper, we focus on DIMA [16], [24]–[43] which refers to in-memory architectures that realize useful computations on the bitlines (BLs) of the BCA in the *low* compute SNR domain, as an intrinsic part of the read cycle. Furthermore, embedding computations on the BLs makes DIMA a massively parallel architecture for realizing a matrix-vector multiply (MVM) compute kernel – a pervasive kernel in ML algorithms. In this paper, we present DIMA with its intrinsic SNR vs. energy trade-off as an analog approach to approximate computing for ML applications.

This paper is organized as follows: Section II provides an overview of DIMA including its various stages of DIMA computation. Design guidelines and a Shannon-inspired rationale for its intrinsic robustness to PVT variations are provided in Section III. DIMA's versatility in realizing a broad class of ML algorithms is described via two case studies in Section IV. In Section V, we demonstrate the use of algorithmic approaches that exploit the intrinsic accuracy vs. energy trade-off of DIMA to push the limits of its energy efficiency via case studies based on two prototype ICs. Finally, in Section VI, we discuss DIMA's fundamental trade-offs in EDP vs. accuracy and identify the most effective design parameters to maximize its EDP benefits over a digital architecture.

II. The Deep In-memory Architecture (DIMA)

This section provides an overview of DIMA. First, the attributes of an idealized DIMA are presented. Then, DIMA is compared with a conventional von Neumann (digital) architecture to identify the key differences. Next, DIMA's analog computational pipeline, is described along with a rationale for its energy and delay benefits over its digital counterpart.

A. Idealized DIMA

We define an *idealized* DIMA as one that satisfies the following properties:

1) **use of a standard BCA:** preserves memory density by using a standard memory BC architecture.
2) **row parallelism:** activates all rows in the BCA simultaneously.
3) **BL computations:** realizes useful analog computations on the BLs.
4) **delayed decision:** implements analog computations on the BL outputs to delay hard-decisions (i.e., digitization, comparison, sense amplification). This requires the design of column-pitch-matched analog circuits that operate in massively parallel fashion.

In practice, existing DIMAs [16], [24]–[43] strive to satisfy these properties to various degrees but none are able to do so fully. Difficulties in satisfying these properties arise due to the high compute SNR requirements often imposed by designers and sometimes by applications, stringent density constraints, device variability, technology limitations, and others. In this sense, DIMA is a full-stack technology that requires one to optimize across applications, systems/algorithms, architectures, circuits and devices.

In the following, we focus on a version of DIMA [16], [17], [24], [26]–[28], [59], [60] that embodies to the fullest extent three of the four properties of an ideal DIMA listed above (property 2 is partially realized). In doing do, we present DIMA's intrinsic SNR vs. energy trade-offs and design methods that exploit DIMA's full-stack nature to push the limits of energy, latency and accuracy.

B. DIMA vs. Digital Architecture

We assume that both conventional digital system (Fig. 2(a)) and DIMA (Fig. 2(b)) employ a $N_{\text{ROW}} \times N_{\text{COL}}$ SRAM BCA based on the standard 6T BC architecture. The read cycle in both architectures is initiated by precharging the BLs to V_{PRE}. What occurs next differs in each architecture.

In the digital architecture, *one row* of the BCA is activated so that each of the N_{COL} BL pairs develops a voltage discharge ΔV_{BL} in proportion to the bit value stored in the activated BC. Next, the $L:1$ (typically $L = 4, ..., 16$) column multiplexers in the periphery of the BCA routes one-of-L BL outputs to the sense amplifiers (SAs) which amplify ΔV_{BL} to full-rail (digital) swing. The column multiplexers allow pitch-matching of large-footprint SAs across multiple columns and the SAs are designed to achieve a very low bit error rate when converting BL discharge into a digital bit.

In contrast, DIMA activates *multiple* (M) rows so that the voltage discharge ΔV_{BL} on a specific BL is a *function* of M bits stored in that column. Furthermore, DIMA eliminates column multiplexers and the SAs altogether and instead processes the BL discharges on all N_{COL} BL pairs *in parallel* via mixed-signal circuits pitch-matched to the BCA columns. Analog-to-digital (A/D) conversion is delayed as long as possible by employing low-voltage energy-efficient mixed-signal computations to minimize the burden on the analog-to-digital converter (ADC). In this manner, DIMA, unlike a digital architecture, computes functions of stored data as an intrinsic part of its read cycle, in a massively parallel fashion, and in analog. Note: the write process in both architectures is identical therefore all the benefits from DIMA accrue from its unique read process.

The differences between DIMA and the conventional digital architecture are summarized in Fig. 2(c).

C. DIMA and its Variants

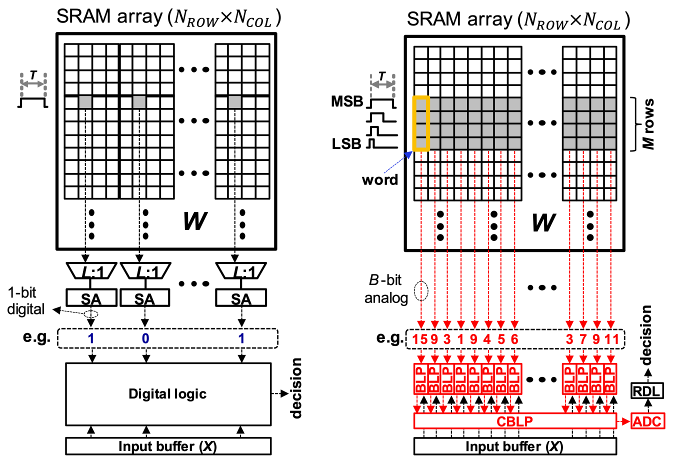
The DIMA (Fig. 2(b)) employs an $N_{\text{ROW}} \times N_{\text{COL}}$ SRAM BCA using a standard 6T [25], [26] BC to store the weights W while streaming in an external input X (see Fig. 1(c)). DIMA designs using 8T [34] or modified BCs [29], [61] have been designed but at a loss in storage density – a critical metric in memory designs. DIMA's compute mode involves the following sequentially executed processes:

- *Multi-row functional read (FR)*: computes weighted sums of the multiple ($M \leq N_{\text{ROW}}$) stored bits in each column. It does so by generating BL voltage discharges or BL currents proportional to the weighted sum on the N_{COL} BLs in parallel per read cycle.
- *BL processing (BLP)*: performs the scalar distance (SD) computation between W and X per column low-swing charge redistribution circuits to enhance energy efficiency [26]. The N_{COL} BLP blocks operate in parallel in a single-instruction multiple-data (SIMD) fashion.
- *Cross BL processing (CBLP)*: aggregates the N_{COL} BLP outputs via charge-sharing to obtain the vector distance (VD).
- *Analog-to-digital converter (ADC)* and *residual digital logic (RDL)*: converts the analog CBLP output into the digital domain for further processing. For simple ML kernels such as the support vector machine (SVM) [62], the ADC generates the final decision. The RDL implements simple functions such as ReLU, min, max, and majority-voting or processes intermediate results.

Thus, DIMA is a highly efficient, i.e., a low-EDP MVM engine and its architecture is matched well to the data-flow intrinsic to widely employed ML algorithms. DIMA processes inference (forward path) computations in DNNs most efficiently though it can also be employed for MVM computations in the back-propagation algorithm [63] employed in training.

There are many variants of the basic DIMA described above including:

1) setting $M = N_{\text{ROW}}$, constraining the bit precision of W $B_w = 1$-bit, in order to access all the rows at a time [25], [31].
2) implementing CBLP in the digital domain after the per-column ADCs [27], [35].
3) restricting all computations to the FR stage only [25].
4) performing specialized functions such as XOR or XNOR within the BCs [33], [65].
5) employing sense amplifiers (SAs) or comparators instead of ADCs [25], [27], [31], [34].
6) by dividing the array into sub-banks, where each sub-bank implements dot product (DP) computations [29].



Attribute	Conventional system	DIMA
word storage pattern	row major	column major
column mux ratio	$L : 1$	$1 : 1$
# of fetched words per access	$N_{\text{COL}}/(LB)$	N_{COL}
BL swing / LSB (ΔV_{BL})	250-300 mV	5-30 mV
# of rows per access	1	M
WL driver	fixed pulse width	pulse width / amp. modulated

(c)

Fig. 2. Comparison of: (a) the digital architecture with a $L : 1$ column mux and sense amplifiers (SAs), (b) DIMA [16], [24], [26] with a functional read (FR) with BL processors (BLPs), and a cross BLP (CBLP), and (c) a summary of the differences. The BCs marked in gray in (a) and (b) are accessed at the same time per precharge/read cycle. Analog domain is shown in red. Note: certain DIMA variants require per-column ADCs [64].

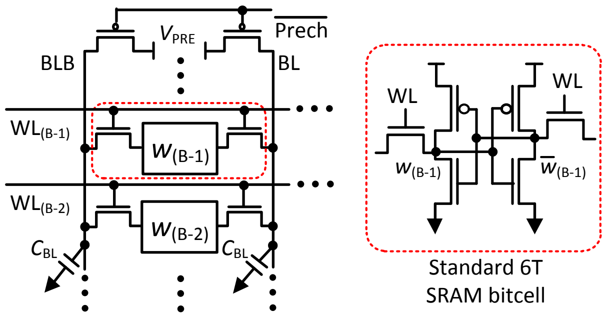
D. DIMA Processing Stages

A detailed description of each DIMA processing stage is presented next.

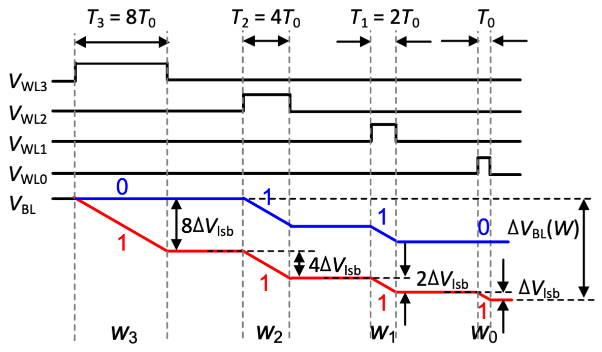
D.1 Functional Read (FR): This stage generates a BL voltage drop ΔV_{BL} (or current flow) proportional to the weighted sum of the column stored bits. Consider FR of M rows using pulse-width modulation (PWM) [16] of the wordline (WL) pulse widths T_i ($i \in [0, M-1]$) as shown in Fig. 3(b) [16]. The total BL voltage drop $\Delta V_{\text{BL}}(W)$ can be represented as follows:

$$\Delta V_{\text{BL}} = \frac{\sum_{i=0}^{M-1} \Delta Q_i}{C_{\text{BL}}} = \frac{V_{\text{PRE}}}{C_{\text{BL}}} \sum_{i=0}^{M-1} \frac{w_i T_i}{R_i} \qquad (1)$$

where ΔQ_i is the charge drawn by the pull-down transistor of the i-th BC in the column. The V_{PRE} is the BL precharge voltage, C_{BL} is the BL capacitance, w_i is the data stored in the i-th BC in the column, and R_i is its discharge path resistance. Equation (1) shows that the BL voltage discharge ΔV_{BL} is a DP between $\{w_i\}$ and $\{T_i\}$ if $R_i = R_{\text{BL}}$, i.e., the discharge currents in the M BCs are identical. Note that



(a)



(b)

Fig. 3. The functional read (FR) operation with $M = B$, which is the bit precision of W: (a) BC column structure, and (b) ideal waveform using pulse-width modulated (PWM) WL enabling signals during a $B = 4$-bit word $W = 1111b'$ (red) and $W = 0110b'$ (blue) read-out. The WL pulses are shown non-overlapped for clarity, but can overlap in practice [16].

pulse amplitude modulation (PAM) based FR has also been employed by modulating the WL pulse amplitudes V_{WL}s [25]. The use of PAM-based FR is limited by the available voltage headroom and read upset considerations.

The FR stage can be employed to realize at least two classes of functions:

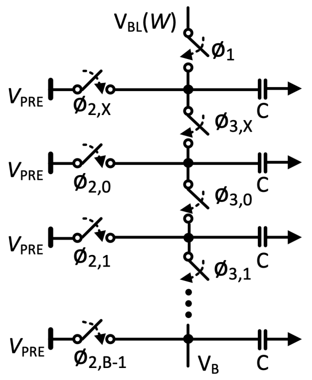
1) *Multi-bit digital-to-analog (D/A) conversion*: By employing the binary-weighted pulse widths $T_i = 2^i T_0$, where T_0 is the LSB, and storing a B-bit W ($W \equiv \{w_0, w_1, ..., w_{B-1}\}$) in a column-major format (see Fig. 3(a)), (1) is transformed into:

$$\Delta V_{BL}(W) = \Delta V_{lsb} \sum_{i=0}^{B-1} 2^i w_i = \Delta V_{lsb} W \qquad (2)$$

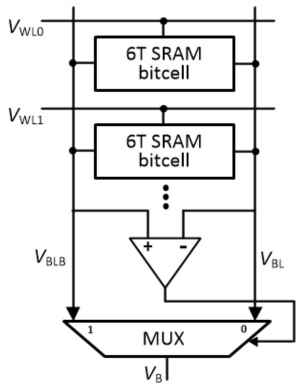
where $\Delta V_{lsb} = \frac{V_{PRE} T_0}{R_{BL} C_{BL}}$. Thus, the FR performs a highly efficient D/A conversion to provide a multi-bit data per BL column. It was shown that 5-bit of data can be D/A converted comfortably without losing accuracy [26]. In contrast, conventional SRAM reads fetch a single bit per BL in a read cycle.

2) *Integrated data read and SD computation*: Equation (2) shows that a simultaneous application of FR to two sets of B rows storing W and X, respectively, generates a BL discharge voltage ΔV_{BL} proportional to $X + W$ as shown below:

$$\Delta V_{BL}(W, X) = \Delta V_{lsb}(W + X) \qquad (3)$$



(a)



(b)

Fig. 4. BL processing (BLP): (a) charge redistribution-based multiplication with $B = 4$ [66], and (b) absolute difference ($|W - X|$), where W and $\overline{X}$ are pre-stored in the same BC column [16].

Similarly, a simultaneous reading of W and $\overline{X}$ generates a ΔV_{BL} and ΔV_{BLB} proportional to $W - X$ and $X - W$, respectively.

D.2 BL Processing (BLP): The BLP implements additional SD computations if required. It resides in the periphery of the BCA pitch-matched to its column (see Fig. 2(b)). The N_{COL} BLPs in Fig. 2(a) accept two input operands: 1) the BL voltage discharge $\Delta V_{BL}(W)$ generated by the FR stage; and 2) an externally provided word X, to generate an output voltage $V_B(W, X)$. The column pitch-matching of the BLP layout makes the BLP stage a massively parallel analog SIMD processor.

The BLP can realize the product $W \cdot X$ of two multi-bit operands W and X which is pervasively employed in the vector DP computation ($\sum_{i=1}^{N} W_i X_i$). A charge redistribution-based mixed-signal multiplier (Fig. 4(a)) accepts an analog inputs $\Delta V_{BL}(W)$ (FR stage output) and a digital input X, whose B bits x_is control the $\phi_{2,i}$ switches, to generate the output voltage $\Delta V_B(W, X) = V_{PRE} - V_B(W, X)$ as follows: [26]:

$$\Delta V_B(W, X) = (0.5)^B X \Delta V_{BL}(W) = (0.5)^B \Delta V_{lsb} W X \quad (4)$$

Therefore, $\Delta V_B(W, X)$ is proportional to the product WX. Note that the BLP multiplier needs to employ unit-sized capacitors rather than ones with binary-scaled sizes as in [67], due to the tight column pitch constraints.

D.3 Integrated FR and BLP Operation: It is also possible to integrate the FR and BLP operations to realize more sophisticated functions such as a multi-bit absolute difference $|W - X|$, which are widely employed for computing the Manhattan distance (MD) ($\sum_{i=1}^{N} |W_i - X_i|$) between two vectors. The absolute difference $|W - X|$ can be represented as [16]:

$$|W - X| = \max(W - X, X - W) \qquad (5)$$

Equation (3) suggests that storing W and the one's complement $\overline{X}$ in two sets of B rows within the same column and

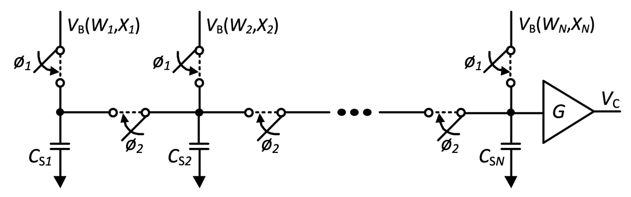


Fig. 5. Cross BL processing (CBLP) [26].

simultaneously applying FR to the $2B$ rows will generate a voltage discharge on the BL and BLB as follows:

$$\Delta V_{\text{BL}}(W, \overline{X}) = \Delta V_{\text{lsb}}(W + \overline{X}) \tag{6}$$

$$\Delta V_{\text{BLB}}(X, \overline{W}) = \Delta V_{\text{lsb}}(X + \overline{W}) \tag{7}$$

A differential comparator and a multiplexer per column processes these voltage discharges to realize a local compare-select operation to compute $\max(V_{\text{BL}}, V_{\text{BLB}})$ and thereby realize (5) (see Fig. 4(b)).

D.4 Cross BL Processing (CBLP): The CBLP (Fig. 5) samples the BLP output voltage ($V_{\text{B}}(W, X)$) on the sampling capacitors C_{S} in each BL column by pulsing the ϕ_1 switches. The sizes of capacitors C_{S} in each column can be either identical or different. For example, in [26], the capacitors are scaled in a ratio of $16 : 1$ in order to combine the BLP outputs of two adjacent columns to obtain an 8-bit BLP results from two 4-bit BLPs.

The CBLP output V_{C} is generated in one shot by closing the switches ϕ_2 with an (optional) amplifier with voltage gain G. Finally, the CBLP output V_{C} is digitized by the ADC for further processing, e.g., by the RDL or by the next DIMA bank. The RDL performs simple slicing/thresholding operations digitally such as sign, tanh, max, min, sigmoid, ReLU, and majority voting.

In a single-ADC DIMA [26], the ADC and RDL process a single scalar value (V_{C}) once after processing a large number (e.g., > 1024) of words via the application of FR, BLP and CBLP processing steps. Therefore, the energy overhead of ADC and RDL is negligible compared to the BCA precharge and computation energy. However, as mentioned earlier, certain DIMA variants employ per-column ADCs to implement the BLP and CBLP in the digital domain. The energy and delay requirements on such ADCs and the resulting overheads will naturally be higher than the single-ADC DIMAs but will remain much smaller, e.g., $< 8\%$ in [61], than that of the BCA.

E. Energy and Delay Benefits

In this section, we provide a justification for DIMA's delay and energy reduction over an equivalent digital architecture via the use of simple circuit models. Specifically, we estimate the normalized delay (τ_{d}) and the normalized energy (ϵ_{e}) which are defined as DIMA's delay and energy costs, respectively, normalized w.r.t. that of an equivalent digital architecture (Fig. 2(a)).

In order to estimate DIMA's normalized delay, we assume that both architectures need to read B bits in order to compute a function. Since the number of bits fetched per read cycle in a digital architecture is limited to N_{COL}/L compared to $N_{\text{COL}}B$ in DIMA, DIMA requires $LB\times$ (where LB is a product of column mux ratio L and the number of bits B fetched for computation) fewer read cycles to fetch the B bits. However, DIMA's cycle time (T_{DIMA}) can be greater than that (T_{digital}) of its digital counterpart since T_{DIMA} includes both data read and compute delays executed via the FR, BLP, and CBLP stages. Making the *worst-case assumption* that the DIMA processing stages are executed sequentially, but without considering the computational delay required by the digital architecture, DIMA's normalized delay τ_{d} to read B bits is as follows:

$$\tau_{\text{d}} = LB\left[\frac{T_{\text{digital}}}{T_{\text{DIMA}}}\right] = \frac{LB}{\gamma} \tag{8}$$

where T_{DIMA} and T_{digital} are the cycle times of DIMA and the digital system, respectively. The normalized delay $\gamma = T_{\text{DIMA}}/T_{\text{digital}}$ ranges from 3 to 6 in $65\,\text{nm}$ CMOS process depending on the type of BLP function. It has been shown that DIMA can enable $B \leq 6$ [26] with $B = 4$ being implemented reliably. Therefore, $\tau_{\text{d}} = 5\times$ to $21\times$ can be achieved in silicon [26] with typical values of $B = 4$, $L = 4$ to 16, $\gamma = 3$ and $\tau_{\text{d}} = 5.3$.

Next, we estimate the DIMA's normalized energy consumption ϵ_{d}. The precharge energy of the large BL capacitances C_{BL} and the leakage energy caused by subthreshold conduction dominate the dynamic and static energy consumption, respectively, in the SRAM BCA. Thus, the energy consumption of the digital system and DIMA to fetch B bits is given by:

$$E_{\text{digital}} = LBC_{\text{BL}}\Delta V_{\text{BL,max}}V_{\text{PRE}} + E_{\text{lk-digital}} \tag{9}$$

$$E_{\text{DIMA}} = \beta C_{\text{BL}}\Delta V_{\text{BL,max}}V_{\text{PRE}} + \frac{E_{\text{lk-digital}}}{\tau_{\text{d}}} \tag{10}$$

where $E_{\text{lk-digital}}$ is the leakage energy in the digital system, and $1 \leq \beta < 2$ is an empirical factor that accounts for DIMA's FR stage being either a B-bit or a $2B$-bit accesses per column when integrating FR with a SD computation [26]. DIMA can be placed into a standby mode earlier than the digital system due to its higher throughput, which in turn reduces the leakage energy by a factor of τ_{d}.

As the first term in (9) and (10) dominates during active (inference) mode, the ratio between these two indicates the energy saving factor ϵ_{e}, given by:

$$\epsilon_{\text{e}} = \frac{E_{\text{DIMA}}}{E_{\text{digital}}} = \frac{LB}{\beta} \tag{11}$$

where $\epsilon_{\text{e}} = 8\times$-to-$32\times$ can be achieved easily with typical values of $B = 4$, $L = 4$, 8, 16 and $\beta = 2$.

DIMA's EDP reduction over its digital counterpart can be represented as follows from (8) and (11):

$$\rho_{edp} = \epsilon_{\text{e}}\tau_{\text{d}} = \frac{(LB)^2}{\beta\gamma} \tag{12}$$

where ρ_{edp} lies in the range $21\times$-to-$1365\times$, of which $\rho_{edp} = 100\times$ has been demonstrated via silicon prototypes [28]. This indicates that EDP gains by DIMA can be further increased. In addition to the savings in memory read energy, DIMA

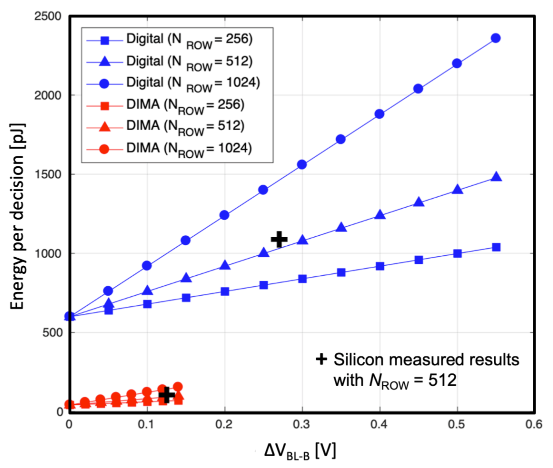


Fig. 6. Energy trends in DIMA vs. conventional digital system from (9) and (10) with respect to N_{ROW}, where $L = 4$, $B = 4$, $\beta = 1$, $N = 128$, and $\Delta V_{\text{BL-B}} = \Delta V_{\text{BL,max}}/B$ for DIMA and $\Delta V_{\text{BL,max}}$ for the digital system. [64].

also achieves roughly $10\times$ lower computational energy as compared to the digital architecture.

The energy models (9)-(10) correlate well with measured results from silicon prototype [26], as shown in Fig. 6, with a modeling error of 11% when $N_{\text{ROW}} = 512$. Figure 6 also indicates that the energy consumption for both DIMA and the digital architecture increases linearly with N_{ROW} (due to the increased BL capacitance C_{BL}) and with the BL swing ($\Delta V_{\text{BL-B}}$) due to the increased precharge energy. However, DIMA realizes its huge EDP gains by harnessing its precharge energy to compute functions of $B \times N_{\text{COL}}$ bits vs. simply accessing N_{COL}/L bits as done by the digital architecture. The price for DIMA's EDP gain is the accompanying loss in its compute SNR. The next section discusses design methods to manage this SNR loss in order to minimize its impact on its inference accuracy.

III. DIMA's Robustness

As DIMA exploits low-swing/low compute SNR analog processing under stringent pitch-matching constraints, it is subject to various circuit non-idealities. However, as compared to traditional analog circuits, the use of DIMA in inference applications combined with its unique architectural data-flow, affords it a certain degree of in-built robustness to circuit non-idealities. This section first presents design guidelines and techniques for minimizing the impact of circuit non-idealities on the compute SNR, and then provides a systems rationale for DIMA's in-built robustness to those non-idealities.

A. Design Guidelines for Functional Read (FR)

The BL voltage discharge expression in (2) assumes the following conditions:

- Condition 1: $T_i \ll R_i C_{\text{BL}}$.
- Condition 2: R_{BL} is constant over V_{BL}, i.e., bias-invariant discharge resistance.
- Condition 3: $R_i = R_{\text{BL}}$, i.e., spatially-invariant discharge resistance.
- Condition 4: $T_i = 2^i T_0$.

Though fully meeting all of the above conditions can be challenging, in practice, these can be approached quite easily. For example, Condition 1 can be achieved by lowering the amplitude V_{WL} of WL enabling pulse to increase R_i. Doing so also helps realize Condition 2 by guaranteeing the access transistors to operate in saturation region across a wide range of V_{BL}. However, an excessive lowering of V_{WL} increases the impact of spatial threshold voltage mismatch of BCs on the cell currents since the access transistors will be operating in the near-threshold voltage regime. For a typical 65 nm CMOS process with $V_{\text{DD}} = 1\,\text{V}$ and $V_{\text{th}} = 0.4\,\text{V}$, choosing $0.55\,\text{V} \leq V_{\text{WL}} \leq 0.65\,\text{V}$ has been found to be a reasonable compromise. Note that Condition 3 gets relaxed by the CBLP stage, where the aggregation process averages out the variations in ΔV_{BL} as shown in Section III-C. Similarly, by choosing T_0 to be large enough, e.g., 250 ps for a 512×256 BCA, makes it possible to meet Condition 4. This value of T_0 ensures that the finite rise and fall times of V_{WL} is a small fraction of T_i.

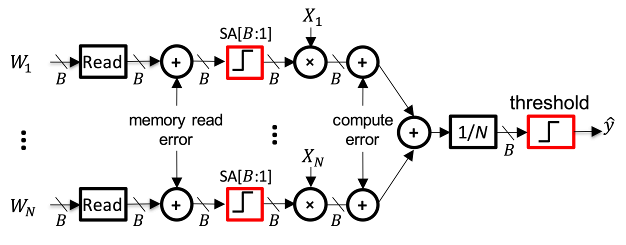
The above guidelines enable the realization of a sufficiently linear FR operation for $B \leq 5$. For $B > 5$, the method of sub-ranged read [26] can be employed whereby $B/2$ MSBs and the $B/2$ LSBs are stored in adjacent BCA columns. The generated BL discharge voltages of those columns by FR process are then merged by weighing the MSB column by factor of $2^{\frac{B}{2}}\times$ more than the LSB column.

B. Design Guidelines for BLP and CBLP

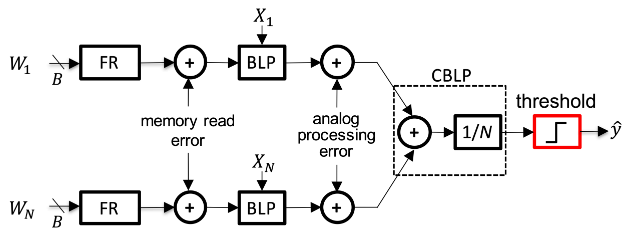
Computation of the BLP (Fig. 4(b)) and aggregation in the CBLP (Fig. 5) are processed by charge sharing mechanism. These circuits suffer from the following noise sources: (1) thermal noise, (2) charge-injection noise, and (3) coupling noise. The capacitors in those blocks need to be sized to guarantee that the error magnitude from thermal noise and charge-injection is smaller than ΔV_{lsb}, the BL voltage discharge corresponding to one LSB. For example, realizing an 8-bit precision with $\Delta V_{\text{BL,max}} = 300$ mV results in 1 mV resolution. Therefore, the thermal noise constraint ($\sqrt{KT/C} < 0.5V_{\text{res}}$) requires $C > 17$ fF at $T = 300$ K, e.g., $C = 25$ fF was used in [26] to provide a sufficient margin. The layouts of BLP and CBLP blocks also need to be carefully designed with proper shielding to minimize coupling noise between the full-swing digital signals and low-swing analog nodes in the BLP and CBLP.

C. DIMA's Algorithmic Source of Robustness

DIMA's intrinsic analog nature raises concerns regarding the impact of non-ideal circuit behavior, e.g., PVT variations, on its inference accuracy. Design guidelines provided in Sections III-A and III-B can help mitigate the impact of such non-ideal circuit behavior. Surprisingly, it turns out that DIMA displays an inherent robustness to process variations due to its high-dimensional vector processing combined with the intrinsic error-tolerance of ML algorithms. This section employs a Shannon-inspired perspective [68] to describe DIMA's algorithmic source of robustness whereby its highly integrated processes of memory read and inference computations are

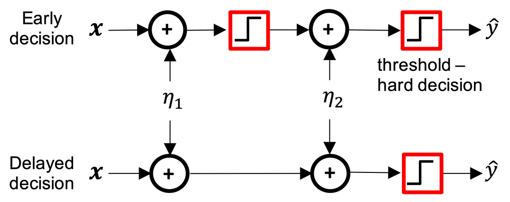


(a)

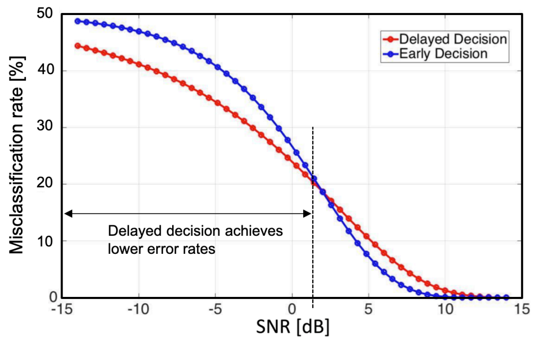


(b)

Fig. 7. A Shannon-inspired view of: (a) the digital architecture, and (b) DIMA, where red boxes are hard-decision blocks [26].



(a)



(b)

Fig. 8. Early vs. delayed decision scenarios: (a) functional flow, and (b) misclassification rate p_e vs. compute SNR [17].

viewed as being equivalent to the process of reliable data transfer over a (noisy) communication channel (see Fig. 7).

Under the Shannon-inspired perspective (Fig. 7), both DIMA and the conventional architectures read data from memory and compute inference functions with the fetched data. The realizable accuracy of the inference functions being computed, e.g., misclassification rate, will depend on how noise sources including PVT variations are controlled or compensated for. The digital architecture in Fig. 7(a) minimizes the impact of non-ideal circuit behavior on its accuracy for the inference task by separating the processes of data read and computation, and by ensuring that each of those processes is deterministic, i.e., high compute SNR. The cost of this strategy is a high EDP. In contrast, though DIMA (Fig. 7(b)) trades-off its compute SNR to obtain enormous EDP gains, it is able to preserve its decision accuracy using three principles: 1) *delayed decision*, 2) *significance-based swing allocation*, and 3) *massive aggregation* as described next.

C.1 Delayed Decision: Delayed decision making implies that no hard decisions are made until it is absolutely necessary, i.e., mandated by the ML algorithm, and occurs at the final stage (Fig. 7(b)). DIMA realizes delayed decision by replacing SAs, which make early hard binary decisions, with analog BLPs implementing scalar operations. In this way, DIMA avoids the loss of information associated with making a hard decision right after noisy BL discharge. Instead of converting BLP outputs into the digital domain via a bank of ADCs, DIMA instead aggregates them via a switched capacitor network in the CBLP stage followed by an ADC. In the ideal scenario, this ADC would generate the final decision based on a scalar CBLP output, e.g., a 3-b output for an 8-way classifier. Thus, DIMA exploits the data compression inherent in the feature extraction functionality of ML algorithms to alleviate both the precision requirements and the sampling rate of the ADC.

The difference between early and delayed decision-making can be observed in Fig. 8(a) for a binary classifier where an input $x \in \{1, -1\}$ and its priors $P(X = 1) = P(X = -1) = 0.5$ with additive noise sources $\eta_1, \eta_2 \sim \mathcal{N}(0, \sigma_n^2)$, which are independent and identically distributed. Assuming zero thresholds, the misclassification rate $p_e = \Pr\{x \neq \hat{x}\}$ for early (digital architecture) and delayed decision (DIMA) scenarios can be derived as [17]:

$$p_e = \begin{cases} 2Q(\frac{1}{\sigma_n})[1 - Q(\frac{1}{\sigma_n})] & \text{(early decision)} \\ Q(\frac{1}{\sqrt{2}\sigma_n}) & \text{(delayed decision)} \end{cases} \tag{13}$$

where $Q()$ is the tail integral of the standard normal $\mathcal{N}(0, \sigma_n^2)$. The plot of (13) in Fig. 8(b) shows that delayed decision incurs a lower p_e achieving higher accuracy than early decision, but at low compute SNRs. This observation indicates that DIMA compensates for its low compute SNR by avoiding hard decisions early in the processing chain.

C.2 Significance-based Swing Allocation: This principle states that energy, and hence signal (e.g., voltage or current) swing, needs to be allocated in proportion to the information content in the data being read/processed. For example, DIMA's FR stage (Fig. 3(b)) assigns BL voltage swings based on the significance of the bit, e.g., with MSBs allocated more swing than the LSBs. When B bits need to be read within a fixed value of the total swing ΔV_{BL}, the swing ΔV_n for the n^{th} bit position is represented as follows:

$$\Delta V_n = \begin{cases} \frac{\Delta V_{\text{BL}}}{B} & \text{(significance-based)} \\ \frac{\Delta V_{\text{BL}} 2^{n-1}}{(2^B - 1)} & \text{(conventional)} \end{cases} \tag{14}$$

It can be shown [17] that significance-based swing allocation assigns approximately 2× higher and 3.8× less swing when $B = 4$ for the MSB and LSB, respectively, compared to the conventional uniform swing assignment. In that manner, DIMA budgets a limited resource (the BL voltage swing) more efficiently with minimum accuracy degradation.

TABLE I
TYPICAL DATA-FLOW OF COMMONLY USED ML ALGORITHMS [69].

$f(D(\mathbf{w},\mathbf{x}))$	Inner loop kernel $D(\mathbf{w},\mathbf{x}) = \sum_{i=1}^{N} d(W_i, X_i)$	$f(\,)$
Support vector machine	$\sum_{i=1}^{N} W_i X_i$	sign
Template matching (L1)	$\sum_{i=1}^{N} \lvert W_i - X_i \rvert$	min
Template matching (L2)	$\sum_{i=1}^{N} (W_i - X_i)^2$	min
Deep neural network	$\sum_{i=1}^{N} W_i X_i$	sigmoid
Feature extraction (PCA)	$\sum_{i=1}^{N} W_i X_i$	$--$
k-Nearest Neighbor (L1)	$\sum_{i=1}^{N} \lvert W_i - X_i \rvert$	majority voting
k-Nearest Neighbor (L2)	$\sum_{i=1}^{N} (W_i - X_i)^2$	majority voting
Matched filter	$\sum_{i=1}^{N} W_i X_i$	min
Linear regression	$\sum_{i=1}^{N} W_i$	accumulate
	$\sum_{i=1}^{N} W_i^2$	accumulate
	$\sum_{i=1}^{N} W_i X_i$	accumulate

C.3 Massive Aggregation: The CBLP stage (Fig. 7(b)) averages out the noise contributions from the BLs via the charge-sharing mechanism by reducing the standard deviation of the noise at the output by a factor of $\sqrt{N}$, assuming that the N noise sources are uncorrelated zero-mean random variables. Both η_1 and η_2 are averaged out by the CBLP stage, boosting the SNR right before the final hard decision (thresholding) when it matters most. There is no such effect in a digital system which aggregates right after the first hard decision in the SA.

Therefore, we see that DIMA's unique architectural data-flow combined with the intrinsic robustness of inference algorithms to computational errors provides for an in-built algorithmic robustness to noise and PVT variations.

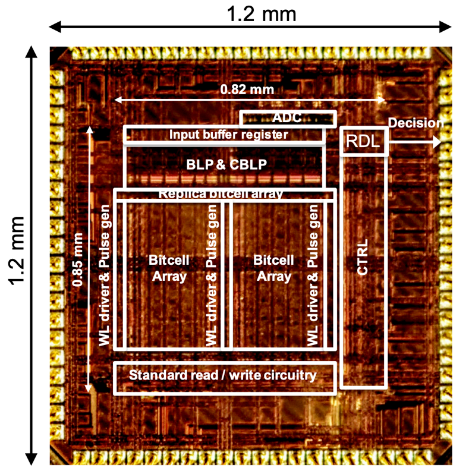
IV. MULTI-FUNCTION AND PROGRAMMABLE DIMA

In spite of its analog nature, DIMA is able to realize a variety of ML algorithms on the same architecture while achieve significant reduction in the system-level EDP. A key reason for DIMA's functional versatility lies in the excellent match between its architectural data-flow and the algorithmic data-flow of various ML algorithms. This section first identifies the commonalities in a variety of ML algorithms and relates it to DIMA's architectural data-flow. Then, two case studies are presented to demonstrate DIMA's functional versatility: (1) the design of a multi-functional DIMA IC prototype in a 65 nm process [26], [70], and (2) PROMISE: a DIMA-based accelerator for ML algorithms [69].

A. Data-flow of ML Algorithms and DIMA

ML algorithms require massive vector distance (VD) computations $D(\mathbf{w},\mathbf{x})$ between a weight vector $\mathbf{w}$ and an input vector $\mathbf{x}$ [71]. Widely employed VD computations include the Hamming distance, L1 distance (Manhattan distance), L2 distance (Euclidean distance), and dot product for many ML algorithms such as the deep neural network (DNN), template matching, k-nearest neighbor (k-NN), matched filter (MF), and support vector machine (SVM) as listed in Table I.

The VD computation dominates the latency and energy consumption in those ML algorithms. As it turns out, much of



(a)

Technology	65 nm CMOS
Die dimension	1.2 mm×1.2 mm
SRAM Capacity	16 kB (512 × 256 BCs)
BC dimension	2.1μm×0.9 μm
CTRL operating freq	1 GHz
Supply voltage (V)	CTRL: 0.85, CORE: 1

(b)

Fig. 9. The multi-functional DIMA IC [26]: (a) silicon prototype micrograph, and (b) chip summary.

DIMA's compute pipeline (FR→BLP→CBLP→ADC+RDL) is well-matched to the data-flow of VD computation, e.g., the VD is computed by first processing N SD computations $d(W_i, X_i)$ (FR→BLP) and then aggregating (→CBLP) to generate the final scalar VD $D(\mathbf{w},\mathbf{x}) = \sum_{i=1}^{N} d(W_i, X_i)$. Finally, the VD passes through a simple thresholding/slicing function $f()$ (→ADC+RDL), such as ReLu, tanh, or sigmoid, to generate the final decision y. There are cases where VDs between a single input vector $\mathbf{x}$ and many weight vectors $\mathbf{w}$ needs to be computed. In such scenarios, DIMA's advantage over the digital architecture will proportionally increase.

B. The Multi-Functional DIMA IC

The multi-functional DIMA prototype (Fig. 9) [26] in a 65 nm CMOS process realizes four tasks: 1) handwritten digit recognition using k-NN; 2) face recognition using TM; 3) face detection using SVM; and 4) gun shot detection using MF [26]. The prototype enables two VD computation modes: 1) dot product (DP) mode for SVM and MF based on the charge-redistribution multiplier in Fig. 4(a), and 2) Mahattan distance (MD) mode for k-NN and TM between the stored data (W) and the input data (X) with the absolute difference computation in Fig. 4(b).

B.1 Architecture and Operation: The multi-functional DIMA architecture (Fig. 10) comprises a CORE, an input register to stream in the input pattern X, and a digital controller (CTRL). The CORE consists of the conventional read/write circuitry, a 512×256 BCA, four 8-bit single-ramp ADCs [72], BLP, and CBLP blocks. The RDL is embedded in the CTRL. A 4 (=

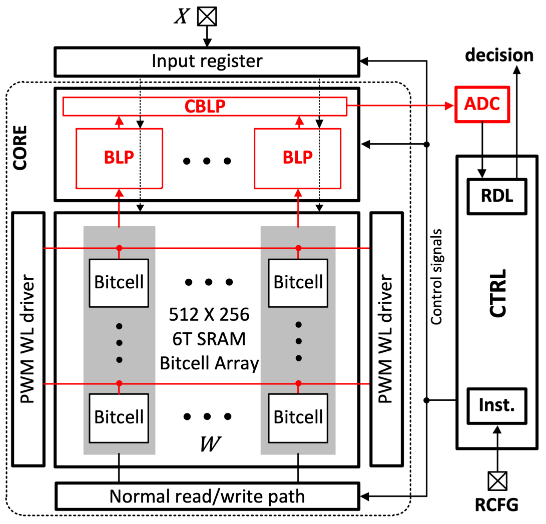


Fig. 10. The multi-functional DIMA IC architecture [26].

L) : 1 column-mux is employed to maximize the throughput of the standard SRAM read. An 8-bit W and X are used to achieve inference accuracy close ($< 1\%$ loss) to that of 8-bit fixed-point digital accuracy [1], [57], [73]. A reconfiguration word (*RCFG*) configures the operation mode of the chip. The CTRL provides the CORE with control signals with a 1 ns time resolution via a 1 GHz master clock (CLK).

The DIMA circuitry takes 19% of the CORE area, with 10% and 9% for the DP and MD modes, respectively. Realizing FR functionality does not incur an area penalty, as pre-existing WL drivers are repurposed.

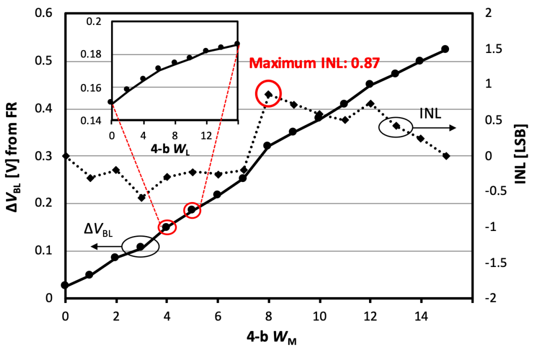
The FR, BLP, and CBLP stages are executed sequentially to generate the outputs V_{BL}, V_{B}, and V_{C}, respectively. Then, the CBLP output V_{C} is sampled and provided as input to the single-ramp ADC. Next, the RDL block further processes the ADC output for thresholding operation. This four-stage processing is iterated until the ML algorithm is fully executed.

B.2 Measured Results: This section describes the measured energy, delay, and accuracy from the silicon prototype.

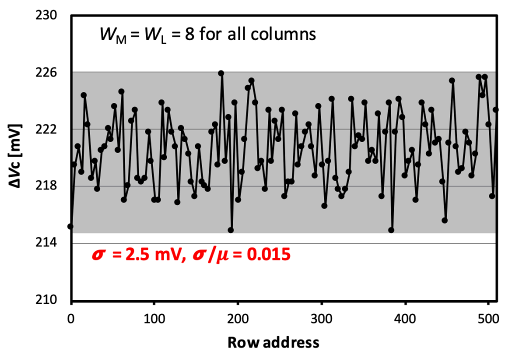
• **Accuracy of FR**: Figure 11(a) shows the measured accuracy of FR operation for 8-bit word W obtained via the sub-ranged read operation achieving an integral non-linearity (INL) $<$ 0.87 LSB. The variation in ΔV_{BL} was also measured (see Fig. 11(b)) by accessing the stored data with the same value from multiple locations in the BCA. It was found that the variation in ΔV_C has a σ/μ of only 1.1% because of the averaging effect of the CBLP stage.

• **Accuracy of CORE output**: Figure 12 shows the CORE output including all the analog processing chain in the FR, BLP and CBLP stages. The measured error magnitude from an ideal linear plot was found to be less than 18 mV with a mean of 4 mV over all combinations of (W,X). These deterministic errors can be compensated via an off-line re-training process in the presence of these errors.

• **Task-level Energy, Delay, and Accuracy**: The CORE decision energy and decision accuracy were measured for two tasks: 1) face detection (binary class) with SVM, and 2) face recognition (64-class) with TM. The inference accuracy



(a)



(b)

Fig. 11. FR accuracy of 8-bit W measured from silicon prototype: (a) BL voltage drop ΔV_{BL} with sub-ranged read, where $W_{\mathrm{M}} = 2^3 w_7 + 2^2 w_6 + 2 w_5 + w_4$ and $W_{\mathrm{L}} = 2^3 w_3 + 2^2 w_2 + 2 w_1 + w_0$, and (b) spatial variations on ΔV_{C} across the entire BCA, accessing via FR, and subsequently aggregating via CBLP [26].

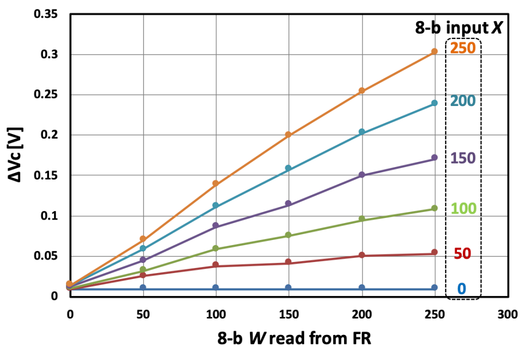
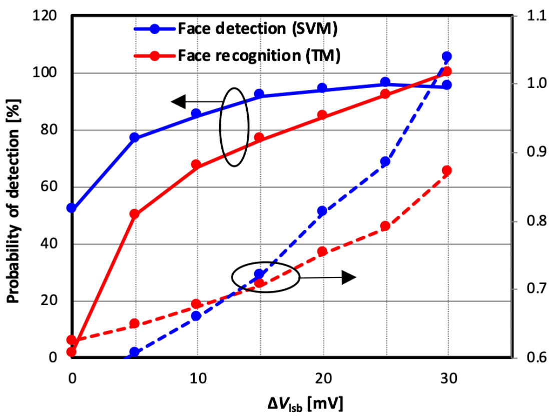


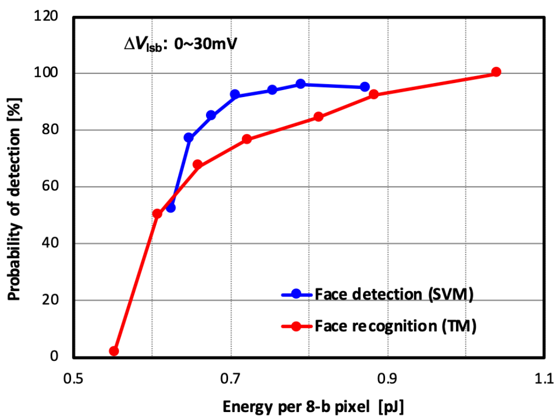
Fig. 12. Measured CORE analog output in the DP mode ΔV_{C} ($\propto \sum W_i X_i$ with 8-bit operands W and X, where the same data W and X are stored in all the columns [26].

(P_{det}) is measured as the ratio of the number of correctly classified queries to the total number of queries. Figure 13(a) indicates that CORE energy is reduced with smaller voltage swing ΔV_{lsb}, but at the degraded detection accuracy (P_{det}). Figure 13(b) clearly shows the trade-off between detection accuracy vs. CORE energy.

• **Benefits over digital architecture**: The benefit of DIMA over the conventional digital system (Conv) is validated by measuring the silicon prototype. The reference system consists of an SRAM with the same BCA as the one in the silicon pro-



(a)



(b)

Fig. 13. Trade-offs between: (a) BL swing per LSB (ΔV_{lsb}) vs. energy and probability of correct detection (P_{det}), and (b) energy vs. P_{det}, obtained from silicon prototype [26] measurements.

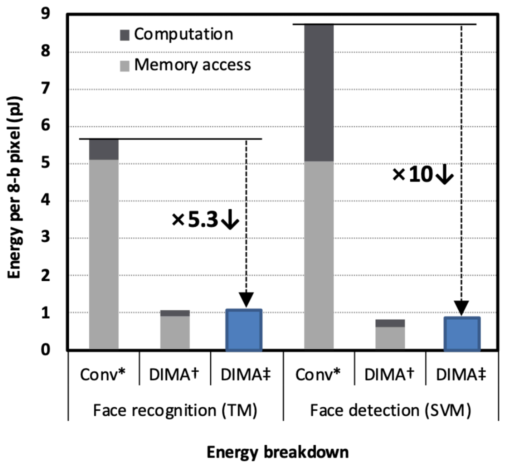


Fig. 14. CORE energy per pixel for DIMA (†post-layout simulations, ‡measured) and a conventional digital system (Conv) (*SRAM energy measured from the prototype and digital computation energy estimated from post-layout simulations) [26].

totype and digital logic blocks synthesized for the DP and TM modes separately. The energy and delay costs of the SRAM read operation in the reference architecture are estimated by measuring the DIMA prototype during the normal read mode whereas those of the synthesized logic block were estimated from post-layout simulations. Figure 14 shows the energy breakdowns for the DIMA and the conventional digital system estimated from post layout simulations, and measured from prototype IC. DIMA achieves 10× and 5.3× energy savings in the DP and MD modes, respectively, which are close to the estimated benefits from the post layout simulations. DIMA also achieves 5.8× and 5.3× throughput improvement in the MD mode (TM and k-NN) and DP mode tasks (SVM and MF), respectively, due to massive parallelism (128 8-bit words per access) compared to the normal SRAM mode (8 8-bit words per access). Therefore, DIMA achieves 32× and 53× EDP gains in the MD and DP modes, respectively.

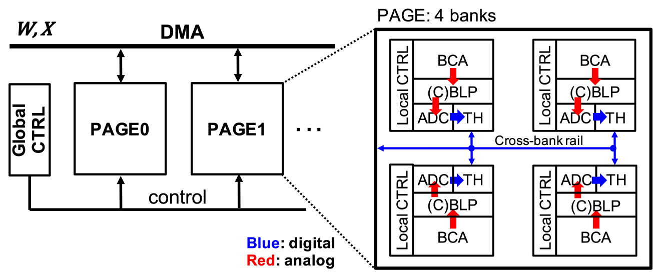


Fig. 15. The PROMISE multi-bank configuration to enable scalability [69].

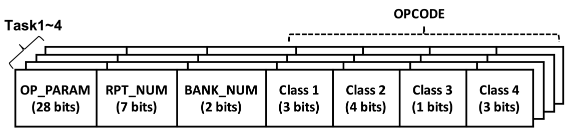
C. PROMISE: A DIMA-based Accelerator

While it is clear that DIMA is highly effective in reducing the system-level EDP of ML algorithms, it does beg the question: *Can DIMA be made programmable while preserving its enormous EDP gains?* To answer this question, this section describes a DIMA-based accelerator called PROMISE [69] which realizes a high degree of programmability without a noticeable loss in efficiency. PROMISE also enables software control over DIMA's intrinsic energy-vs-accuracy tradeoff via its instruction set.

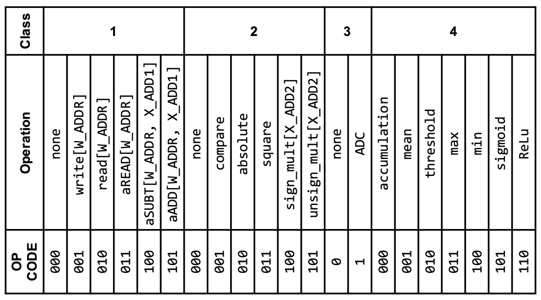
C.1 PROMISE Architecture: PROMISE is built on the DIMA core (Fig. 10 [26]) with four processing stages **(S1)** `aREAD`, **(S2)** `aSD`, **(S3)** `aVD`, and **(S4)** ADC and TH, which correspond to the FR, BLP & CBLP, ADC, and RDL stages described in Section II-D.

PROMISE includes digital blocks `CTRL` and `X-REG` to provide the programmability. A PROMISE bank consists of 512 × 256 (= $N_{\text{ROW}} \times N_{\text{COL}}$) BCA similar to the multi-functional DIMA in Fig. 10, which reads out a 128-element vector at a time and seamlessly converts it to corresponding analog values. The `aREAD` stage also supports the optional element-wise addition or subtraction with an 128-element input operand vector X.

The `aSD` and `aVD` stages process the 128 analog values from the `aREAD` stage at a time, followed by the `ADC` stage. The digitization in the `ADC` stage also prevents the noise



(a)



Class	Operation	OP CODE
1	none	000
	write[W_ADDR]	001
	read[W_ADDR]	010
	aREAD[W_ADDR]	011
	aSUBT[W_ADDR, X_ADD1]	100
	aADD[W_ADDR, X_ADD1]	101
2	none	000
	compare	001
	absolute	010
	square	011
	sign_mult[X_ADD2]	100
	unsign_mult[X_ADD2]	101
3	none	0
	ADC	1
4	accumulation	000
	mean	001
	threshold	010
	max	011
	min	100
	sigmoid	101
	ReLu	110

(b)

Fig. 16. PROMISE instruction set: (a) instruction format, and (b) operations in each `Class` [69].

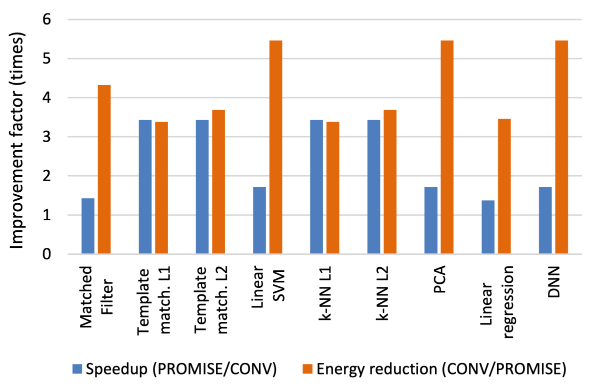


Fig. 17. PROMISE speed-up and energy savings (with `SWING` = 111) over CONV [69].

accumulation from excessively long analog processing chain. `TH` supports not only the decision functions $f()$ in Table I, but also aggregates partial sums when the vector length $N > 128$. `CTRL` generates the enable signals for all the other blocks based on a given *instruction* to make DIMA a programmable mixed-signal accelerator. Finally, `X-REG` holds streamed-in eight 128-element input vectors Xs.

Figure 15 shows an 8-bank PROMISE with two `PAGEs`, where four banks are employed per page. Therefore, long (>128) vectors can be processed in parallel fashion by distributing the vector across multiple banks. After in-memory processing, digitization via ADCs is followed by aggregation of the resulting partial sums in the digital domain.

C.2 PROMISE Instruction Set: The PROMISE ISA to enable programmable mixed-signal accelerator is described next:

• **Instruction Format:** PROMISE ISA is based on a wide-word macro instruction format, `Task`, which consists of multiple operations. This is similar to a very-large instruction word (VLIW) ISAs except that the operations are sequentially processed due to DIMA's analog compute pipeline rather than parallel as in VLIW architectures. PROMISE ISA (Fig. 16(a)) includes four `Class` fields, which correspond to four pipelined stages of PROMISE with three other fields, `OP_PARAM`, `RPT_NUM`, and `MULTI_BANK` for further configurations.

• **Operating Parameter Field:** `OP_PARAM` (33 bits) configures operating parameters, e.g., address, to support flexible programmability. The `SWING` parameter also controls BL swing ΔV_{BL}, e.g., 5 mV/LSB - 30 mV/LSB with eight different levels to exploit the trade-off between accuracy vs. energy efficiency.

• **Class Fields:** `Class-1` includes six possible memory operations including `READ`, `WRITE`, or `aREAD` for a standard SRAM read, and write, or analog read operation, respectively. `aADD` (`aSUB`) processes one `aREAD` and an element-wise addition (subtraction) in the analog domain with operand W read from the BCA and input vector X from `X-REG`. `Class-2` chooses one of five possible `aSD` operations such as `compare`, `absolute`, `square`, `sign_mult` and `unsign_mult` and the following optional `aVD` operation for the aggregation. `Class-3` controls whether an `ADC` operation is performed or not, and `Class-4` specifies one of seven `TH` operations such as `ReLu`, `mean`, `max`, `min`, `sigmoid`, `accumulation`, and `threshold`.

• **Loop Control Field:** `RPT_NUM` controls how many times the `Task` is iterated to process the distance computation $D(\mathbf{w}_j, \mathbf{x})$ with multiple $\mathbf{w}_j$s.

• **Multiple Bank Control Field:** `MULTI_BANK` defines the number of banks used to distribute long (>128) vectors for parallel processing. These vector elements need to be distributed within the same row across multiple banks to be processed in parallel. The output of a `Class-4` operation can be transferred to the `X-REG` or `TH` block of any bank by controlling the destination *DES* in `OP_PARAM`. For large-scale applications, where the vectors are too long to fit into multiple banks, those vectors can be sequentially processed by setting `RPT_NUM` and other parameters.

C.3 Evaluation: Commonly employed ML algorithms such as SVM, linear regression, principle component analysis (PCA), TM with L1 and L2 distances, and k-NN [69] were mapped to PROMISE including three different DNN models with differing levels of network size and complexity. The 8-bit precision is employed for PROMISE to maintain the accuracy close to floating-point implementations [1], [57], [73], [74]. On the other hand, the baseline digital architecture (CONV) includes conventional SRAM and fixed-point computational logic synthesized for the specific algorithm with the minimum bit precision required per benchmark.

Figure 17 shows that PROMISE achieves $1.4 - 3.4\times$ throughput benefits as compared to CONV across the benchmarks. It is also shown that PROMISE achieves energy savings of $3.4 - 5.5\times$ compared to CONV, leading to $4.7 - 12.6\times$ EDP reductions. These benefits are mainly achieved from aREAD (`Class-1`) and aSD/aVD (`Class-2`) stages with low-voltage swing mixed-signal processing. Despite the increased CTRL complexity for the programmability, its energy is <10% of the total energy demonstrating that the programmability overhead of PROMISE is negligible.

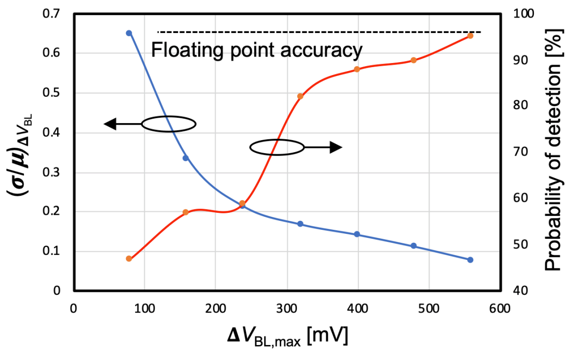


Fig. 18. Spatial variations $(\sigma/\mu)_{\Delta V_{\rm BLB}}$ and decision accuracy with respect to $\Delta V_{\rm BL,max}$ measured from silicon prototype across 30 randomly chosen 4-row groups on [28].

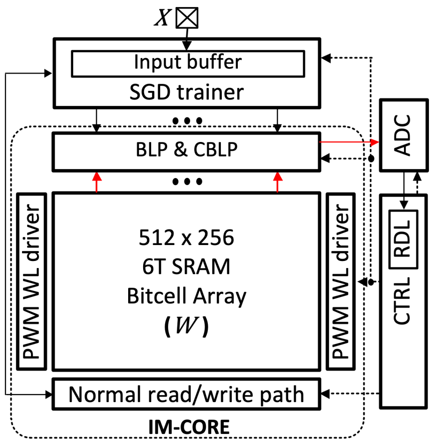


Fig. 19. SGD-SVM DIMA IC chip architecture showing the IM-CORE, trainer and CTRL, where analog signals are marked in RED [28].

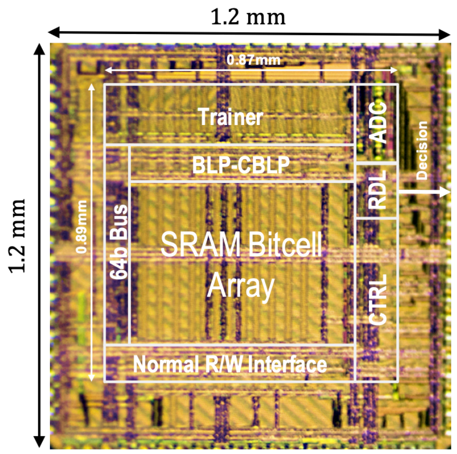


Fig. 20. Chip micrograph of DIMA IC with an on-chip trainer [28].

V. Enhancing DIMA's Efficiency via Error Compensation

Sections III and IV have shown that the impact of DIMA's non-ideal analog behavior can be overcome to a great extent via a combination of: 1) good circuit design guidelines; 2) its unique Shannon-inspired architectural attributes such as a) delayed-decision making, b) significance-based swing allocation, and c) SNR boosting via aggregation; and 3) the intrinsic tolerance of ML algorithms to computational errors. In this process, DIMA realizations [26] have demonstrated both excellent robustness to PVT variations, and substantial EDP gains over their digital counterparts.

Nevertheless, if DIMA's EDP gains are to be enhanced even further, one will need to lower the compute SNR more aggressively, e.g., per (10) DIMA's energy consumption reduces with the BL voltage discharge $\Delta V_{\rm BL}$. However, doing so leads to an increase in the variations observed in $\Delta V_{\rm BL}$ and hence to a loss in accuracy as seen in Fig. 18, where $\Delta V_{\rm BL}$ is controlled by WL enabling voltage $V_{\rm WL}$. Indeed, reducing the maximum BL swing $\Delta V_{\rm BL,max}$ increases the normalized variation (σ/μ) in $\Delta V_{\rm BL}$ and a corresponding increase in the misclassification rate p_e of a SVM algorithm implemented in DIMA [28].

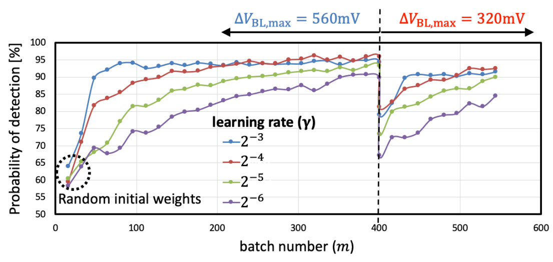
In order to address this loss in inference accuracy, one will need to employ algorithmic techniques since we can safely assume that the design guidelines and the intrinsic error tolerance of ML algorithms would have been exhausted at this point. This section describes two such techniques: a) the use of on-chip learning [28], and b) the use of ensemble methods [27].

A. Error Compensation via On-chip Learning

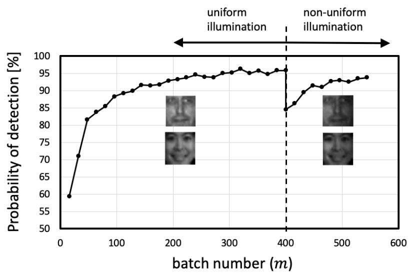
Since ML algorithms employ data-driven training methods to learn sufficient statistics for accurate inference, it is possible to harness the power of such methods to realize an on-chip learning set-up whereby the training method adapts to both data statistics and the statistics of non-ideal circuit behavior such as those due to process variations. Such on-device training infrastructure would be critical for other reasons as well: 1) to enable always-ON Edge devices to adapt to changing environmental characteristics that impact the input data statistics and noise, 2) for privacy reasons related to the sensitivity of locally sensed data, e.g., wearable biomedical devices, and 3) to track the temporal changes in electrical device characteristics (with temperature, and aging).

We illustrate the efficacy of on-chip training via the case study of a DIMA IC prototype in a 65 nm CMOS process [59] that employs an stochastic gradient descent (SGD)-based on-chip training to implement a robust SVM classifier. The prototype IC implements the feedforward computations in DIMA and feedback computations using on-chip digital circuitry. This prototype demonstrates that on-chip training not only adapts to chip-specific spatial variations in the BCA but also to data statistics, thereby further enhancing DIMA's energy efficiency.

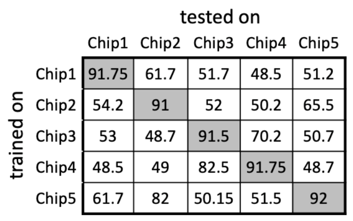
A.1 Systems Rationale: The non-idealities in DIMA are dominated by spatial transistor threshold voltage variations, discharge path non-linearity, and the finite transition times of the WL pulses. The effects of these non-idealities do not change over time. As a result, even though the weights $\mathbf{w}$s are stored in the BCA, the FR outputs $\mathbf{w}' = H(\mathbf{w})$, where $H(\cdot)$ is a function that accounts for the impact of spatial variations, and is chip-specific, unknown, and static after fabrication. It can be shown [28] that the SGD algorithm will converge to an optimal solution when the feed-forward path implemented on



(a)



(b)



tested on / trained on

	Chip1	Chip2	Chip3	Chip4	Chip5
Chip1	91.75	61.7	51.7	48.5	51.2
Chip2	54.2	91	52	50.2	65.5
Chip3	53	48.7	91.5	70.2	50.7
Chip4	48.5	49	82.5	91.75	48.7
Chip5	61.7	82	50.15	51.5	92

(c)

Fig. 21. Learning curves measured from silicon prototype shows robustness to: (a) process variations, (b) variations in input statistics with $\Delta V_{\mathrm{BL,max}} = 560\,\mathrm{mV}$, and (c) classification accuracy with the weights trained on a different die. A batch size $N = 64$ and a regularization factor $\lambda = 2^{-4}$ is used [28].

DIMA computes $\mathbf{w}' = H(\mathbf{w})$ provided $H(\cdot)$ is monotonic – a condition that is easily satisfied by DIMA since it reads a weighted function of the bits of W instead of the bits directly. Therefore, on-chip learning can enable accurate inference in the presence of die-specific variations by learning both the optimal weights ($\mathbf{w}$) as well as the die-specific variations ($H(\cdot)$).

The on-chip training can be also employed in a conventional digital system to compensate PVT variations with reduced ΔV_{BL}. But, this approach is less effective for the digital system because the variations can cause SA bit errors, including MSB errors, which will be catastrophic, i.e., $H(\cdot)$ is non-monotonic in $\mathbf{w}$. This observation has been confirmed via measurements in [59].

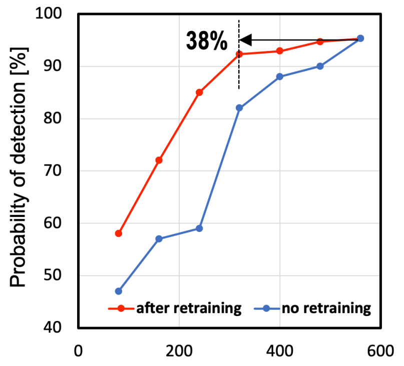
A.2 Architecture and Circuit Implementation: The architecture of the prototype IC in Fig. 19 is based on [26] (see Fig. 9) and includes following blocks: (a) the in-memory CORE (IM-CORE) for in-memory computation; (b) the SRAM peripheral circuitry for standard read/write operations; (c) decision block; (d) the digital trainer; and (e) a digital controller (CTRL). There are following operation modes: 1) standard SRAM READ/WRITE mode; 2) in-memory inference mode; and 3) on-chip learning mode. The in-memory inference function is given by:

$$\mathbf{w}^{\mathsf{T}}\mathbf{x}_k + b \underset{\hat{y}_k=-1}{\overset{\hat{y}_k=+1}{\gtrless}} 0, \tag{15}$$

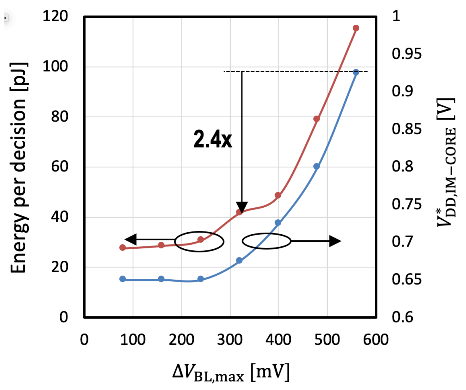
where $\hat{y}_k$ is the decision (label) for the input vector $\mathbf{x}_k$. The weight vector $\mathbf{w}$, and a scalar bias b are parameters of the SVM algorithm. The IM-CORE block implements the dot-product operations, and a set of comparators and ADC in the decision block generates the estimated class label $\hat{y}_k$.

The training algorithm is a batch SGD-SVM implemented in the digital domain. The trainer includes two sub-blocks: 1) an input buffer to store the streamed-in input $\mathbf{x}$, and 2) a gradient buffer to store intermediate gradients. The updated weights are written once per batch into the BCA to amortize the energy and latency costs from SRAM write operation. The trainer allows the learning rate γ to be configured in powers-of-2.

A.3 Measurement Setup: We evaluate the prototype IC (see Fig. 20) for a face detection task on MIT CBCL dataset [75]. It is a binary classification problem where the dataset comprising



(a)



(b)

Fig. 22. Measured: (a) classification accuracy as a function of $\Delta V_{\mathrm{BL,max}}$, where 38% reduction in $\Delta V_{\mathrm{BL,max}}$ is achieved by on-chip learning, and (b) IM-CORE energy as a function of $\Delta V_{\mathrm{BL,max}}$ showing a 2.4× energy saving ($V^*_{\mathrm{DD,IM\text{-}CORE}}$: minimum supply voltage to prevent destructive read) [28].

4000 training and 858 test images. The images are re-sized to 11×11 pixels such that it fits into the 128 word row width of the DIMA BCA. During training, input batch was generated by sampling the training set randomly with replacement. During convergence, the misclassification rate p_e was measured at the end of every 8^{th} batch.

A.4 Robustness Enhancement: Starting from random initial weights in the BCA, on-chip learning converges to a misclassification error rate of $p_e \leq 7\%$ as shown in Fig. 21. At the end of 400 batch updates with $\gamma = 2^{-3}$ and 2^{-4},

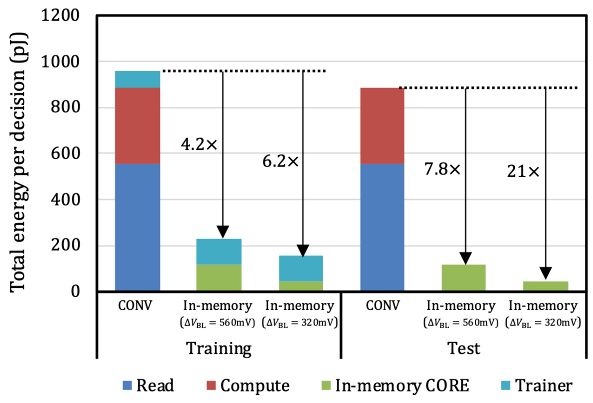


Fig. 23. Measured on-chip energy compared to a conventional digital reference architecture (CONV) for training (at $N = 64$) and inference, showing a simultaneous reduction in energy consumption and delay by 21× and 4.7×, respectively. The supply voltage $V_{\text{DD,IM-CORE}}$ and ΔV_{BL}: (1 V, 560 mV) and (675 mV, 320 mV) are tested [28].

on-chip learning results in the p_e within 1% of floating-point accuracy. When $\Delta V_{\text{BL,max}}$ is reduced to 320 mV from 560 mV at the batch number $m = 400$, the misclassification rate p_e increases drastically to 18% due to the increased impact of spatial variations. The misclassification rate p_e recovers to <8% with $\gamma \leq 2^{-3}$ due to the continual on-chip learning. Similarly, p_e increases to 16% with an abrupt change of the illumination in the input images at $m = 400$ (see Fig. 21(b)), but further training eventually reduces it down to 6%. The on-chip learned weights are found to be chip specific, e.g., when weights learned from specific chip are tested on other chip instances, the average p_e increases from 8.4% to 43% (see Fig. 21(c)). Figure 21 indicates the enhanced robustness to chip-specific variations in the process parameters and the input data statistics due to the on-chip training.

A.5 Reduction in System-level EDP: The minimum $\Delta V_{\text{BL,max}}$ required to maintain $p_e \leq 8\%$ is 520 mV without on-chip training (Fig. 22(a)). On the other hand, the target misclassification rate can be achieved with a 38% lower $\Delta V_{\text{BL,max}} = 320$ mV with on-chip learning. Operating with $\Delta V_{\text{BL,max}} = 320$ mV enables the reduction of $V_{\text{DD,IM-CORE}}$ from 0.875 V to 0.675 V without encountering destructive reads (see Fig. 22(b)). Thus, on-chip learning enables a 2.4× reduction in IM-CORE energy without accuracy degradation. The prototype achieves energy reduction of 1.5×-to-2.6× for p_e in the range 5%-to-8% over the multi-functional DIMA IC in Section IV due to on-chip learning.

The SRAM write operations during the weight update operation at the end of each batch dominates the training energy costs. However, this cost is amortized over the batch size N. For $N = 128$, the contribution of SRAM writes is < 26% of the total training energy, and 60% of the total energy can be attributed to the controller which is amortized over the BCA size.

The measured result shows a 7.8× energy savings over CONV, which is a conventional digital architecture including an SRAM with the same BCA size as the one in DIMA prototype, when operating with *pre-trained* weights at $\Delta V_{\text{BL}} =$ 560 mV (see Fig. 23). With on-chip training, this energy reduction increases to 21×. Combined with a 4.7× reduction in delay, DIMA achieves an overall 100× reduction in EDP over an equivalent digital architecture.

SRAM writes in DIMA during training are no different than in the conventional architecture and such writes are energy expensive. In order to alleviate the cost of SRAM writes, training is done in batch mode and write are executed once per batch. With batch mode updates and writes, the energy reduction factor during training reduces to 6.2× with batch size $N = 64$ from its value of 21× during inference mode. Since practical environments are quasi-static or slowly varying, it is expected that the training mode will be activated very infrequently, and hence the energy savings in the inference mode will manifest themselves.

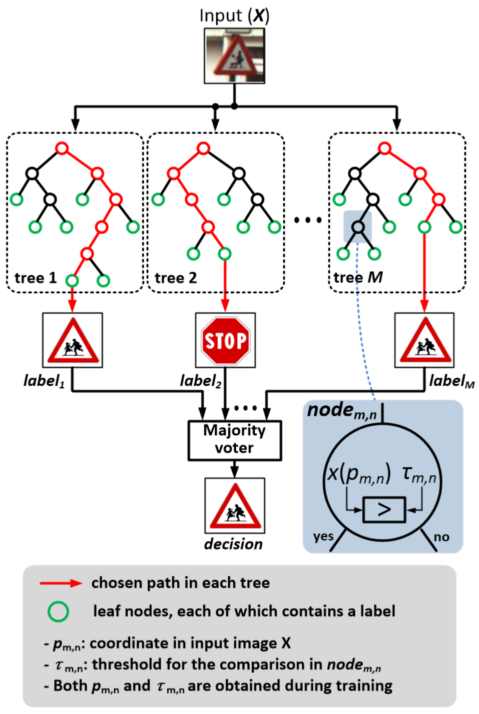

Input (X)
tree 1
tree 2
tree M
label1
label2
labelM
STOP
Majority voter
decision
node m,n
x(p m,n)
τ m,n
>
yes
no
chosen path in each tree
leaf nodes, each of which contains a label
- p m,n: coordinate in input image X
- τ m,n: threshold for the comparison in node m,n
- Both p m,n and τ m,n are obtained during training

Fig. 24. The random forest (RF) algorithm (RSS: random sub-sampling) [27].

B. Error Compensation via the Random Forest (RF) Algorithm

Another algorithmic approach to enhance DIMA's robustness to PVT variations is to employ ensemble algorithms such as *Boosting* [76] and *Bagging* [77]. Such algorithms employ multiple weak (low-accuracy) classifiers whose decisions are aggregated to obtain a strong (high-accuracy) classification at the final output. For example, [25] employed the AdaBoost algorithm whereby binary-weighted SVMs are realized on the BLs and whose outputs were weighted, summed, and sliced to obtain the final (strong) decision.

The RF algorithm [78] is another option. This algorithm employs an ensemble of decision trees as weak classifiers and employs a simple majority-vote to obtain a strong classifier decision. Furthermore, the RF algorithm scales easily to multi-class problems and involves simple operation, e.g., comparisons, which are suitable for being mapped to DIMA.

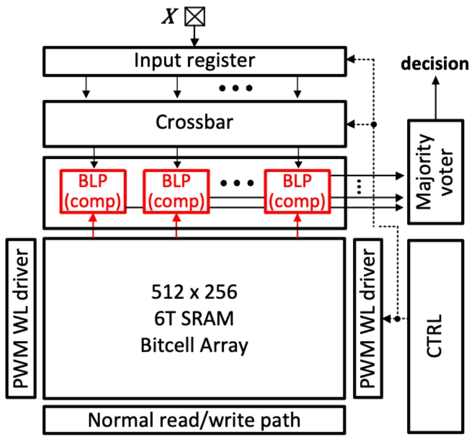


Fig. 25. DIMA implementing the RF algorithm. Analog computations are marked in RED [27].

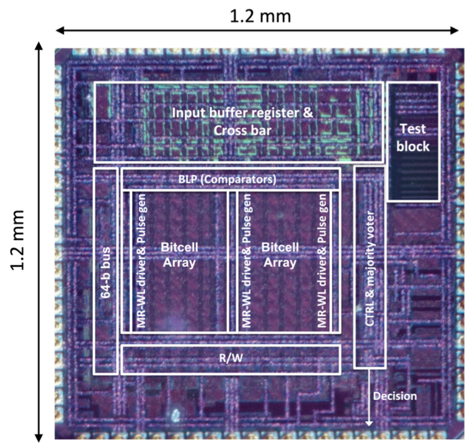


Fig. 26. Micrograph of the DIMA-based RF classifier IC [27].

We illustrate the use of the RF algorithm via a case study of a DIMA IC prototype in a 65 nm CMOS process [27], [79]. This prototype demonstrates that the RF algorithm is effective in generating accurate inferences in spite of the increased variations in the BL voltage discharge ΔV_{BL} that would occur if the maximum BL discharge were to be reduced to save energy (see Fig. 18).

B.1 Random Forest Algorithm: Figure 24 describes the RF algorithm, which consists of M decision trees with a maximum of N nodes per tree. Each tree processes randomly sub-sampled (RSS) pixels input image $\mathbf{x}$ based on a pseudo-random pattern vector that is obtained during the training stage. For example, given a pattern vector $\mathbf{p}_m \equiv \{p_{m,1}, p_{m,2}, .., p_{m,N}\}$ associated with the m-th tree, the n^{th} node of this tree compares the pixel (or feature) $x(p_{m,n})$ indexed by $p_{m,n}$ with a threshold $\tau_{m,n}$ to obtain a node-level binary decision $q_{m,n}$. Each node decision $q_{m,n}$ dictates if the left or the right branch needs to be taken. This process is iterated until the binary search reaches one (l^{th}) of leaf nodes with the label $c_{m,l}$, which corresponds to the tree-level decision. The final ensemble decision is obtained by majority-voting the decisions of all the M trees.

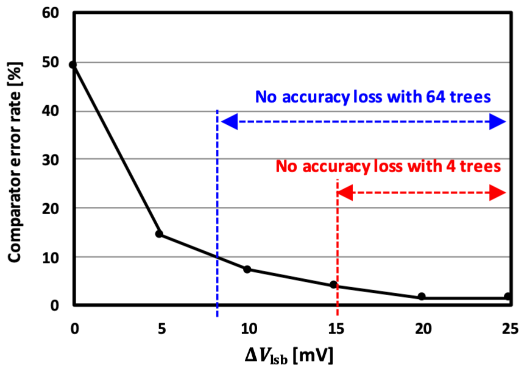


Fig. 27. Comparator error rates measured from prototype IC. Note: ΔV_{lsb} is fine-tuned by controlling the WL access pulse V_{WL} (see Fig. 3), and estimated from (16) by employing measured values of ΔV_{BL} with $X = 15$ and $T = 0$ in the test mode of silicon prototype [27].

B.2 Decision-trees in DIMA: The decision tree comparisons in DIMA are implemented in highly parallel fashion by unrolling the tree across the memory array (see Fig. 25). This unrolling eliminates the need to conditionally fetch the B-b thresholds $\tau_{\text{m,n}}$. The comparison begins with the simultaneous application of WL access pulses to all the rows storing a threshold $T = \tau_{m,n}$ and a pixel $\overline{X} = x(p_{m,n})$ as in the functional read (FR) process similar to the MD mode in Section II. Therefore, ΔV_{BL} is proportional to $(T - X)$ given as:

$$\Delta V_{\text{BL}}(T, \overline{X}) = \frac{V_{\text{PRE}} W_0}{R_{\text{BL}} C_{\text{BL}}} \sum_{i=0}^{B-1} 2^i (t_i + \overline{x_i}) \\ = \Delta V_{\text{lsb}}(T - X - 1) \qquad (16)$$

where x_i and t_i are the i^{th} bit of X and T, respectively. Due to the complementary nature between BL and BLB, ΔV_{BLB} is also proportional to $(X - T)$. Then, analog comparators [80] generate node-level decisions $(q_{m,n})$ by comparing ΔV_{BL} and ΔV_{BLB} in parallel per each column.

B.3 Architecture: Figure 25 shows the overall architecture of silicon prototype, which includes a CORE with a 512×256 SRAM BCA, FR WL drivers, peripherals for standard read/write operations, an input buffer to store the streamed-in 256-b $\mathbf{x}$, a digital CTRL, a 64-b I/O ($B_{\text{IO}} = 64$) with a $4:1$ ($L = 4$) column mux, a majority voter, and crossbars. Other features are the same as those listed in Fig. 9(b).

The prototype IC processes a group of four trees in parallel, requiring 171 clock cycles. A total of $M/4$ such groups are processed sequentially, where $M \leq 168$. The processing begins by storing the four copies of 4:1 sub-sampled image $\mathbf{x}$s in the input buffer. Then, the pixel indices $p_{m,n}$s are fetched from the BCA into index registers inside the cross bar through 12 normal SRAM read accesses. Next, according to the index $p_{m,n}$, the B = 8-b pixels $x(p_{m,n})$s are fetched from input register and subsequently placed into the RSS registers in the cross bar. Finally, the $x(p_{m,n})$s are written into the replica BCA so that in-memory comparison between $x(p_{m,n})$ and thresholds $\tau_{m,n}$ is prepared. The FR operation begins by applying PWM WL enabling signals, and then BL and BLB are fed to analog comparators. The 128 pitch-

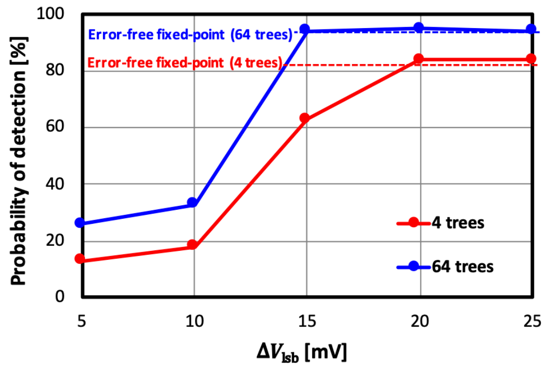


Fig. 28. BL voltage swing ΔV_{lsb} vs. decision accuracy (P_{det}) with the number of trees $(M) = 4$ and 64 [27].

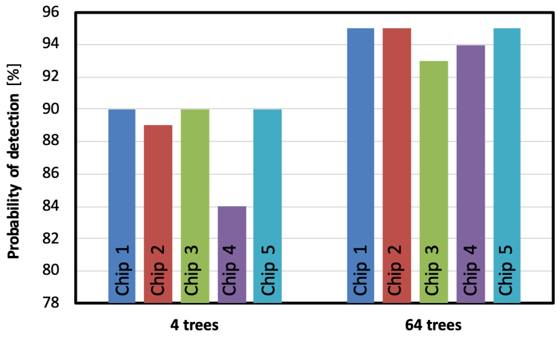


Fig. 29. Classification accuracy for traffic sign recognition across chips with numbers of trees (M) = 4 and 64 at $\Delta V_{\text{lsb}} = 25\,\text{mV}$ [27].

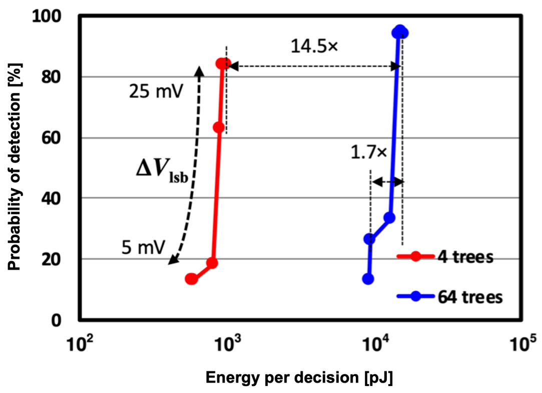


Fig. 30. Trade-off in energy vs. accuracy for traffic sign recognition task with respect to the number of trees (M) at $\Delta V_{\text{lsb}} = 5$ to $25\,\text{mV}$ [27].

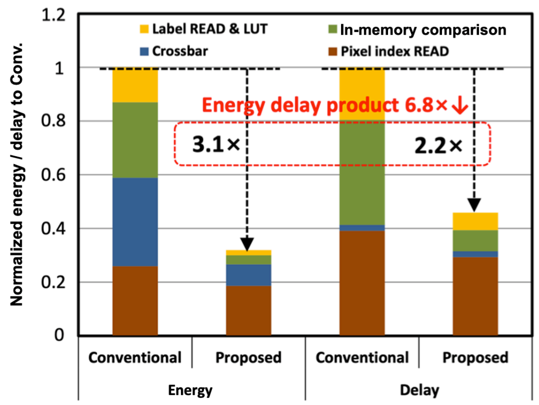


Fig. 31. Energy and delay breakdown at $\Delta V_{\text{lsb}} = 25\,\text{mV}$ ($\Delta V_{\text{BL}} = 200\,\text{mV}$) obtained via post-layout simulations [27].

matched analog comparators generate 128 comparison outputs $q_{m,n}$s per precharge cycle in parallel. The controller uses the address generated by the in-memory comparison outputs $q_{m,n}$, to fetch the four tree-level labels $c_{m,l}$s from the BCA. The final decision is generated by majority voting the M tree-level labels $c_{m,l}$ after processing $M/4$ such groups.

B.4 Measurement setup: An 8-class traffic sign recognition task on the KUL Belgium traffic sign dataset [81] was employed to test the IC. The tree depth was chosen to be six, which is the minimum needed to minimize the accuracy degradation in the target application [82]. The impact of the number of trees on accuracy was evaluated by testing the IC with $M = 4$ and $M = 64$. We used 200 randomly chosen test images were used to determine the classification accuracy (P_{det}).

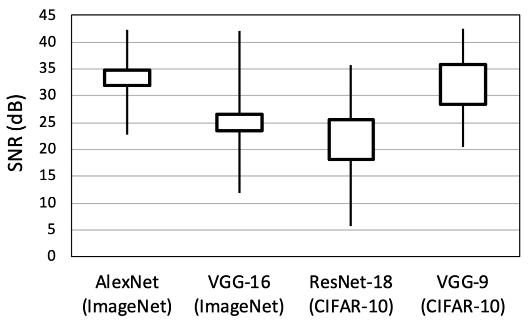
The DIMA RF IC is compared with a conventional digital implementation that employs the same architecture as in Fig. 25, but with a 256 : 1 crossbar and digital logic for the comparison operations. Eight digital comparisons are performed in parallel for each read cycle that fetches eight $\tau_{m,n}$s (64-b) from the SRAM. The memory and the digital comparators are pipelined to improve throughput. The energy and delay costs of the conventional digital system are estimated by combining the measured results of normal SRAM read access from the prototype IC and the post-layout simulation results of the crossbar and digital comparators.

B.5 Measured Robustness: The BL swing ΔV_{BL} has maximum 25 mV deviations due to the multiple sources of process variations. The measured error rate of in-memory comparison increases from 1.6% to 14.5% as a consequence of increased impact of process variations when ΔV_{lsb} reduces from 25 mV to 5 mV (see Fig. 27). We measured the comparator errors at each ΔV_{lsb} during the classification with the KUL Belgium dataset.

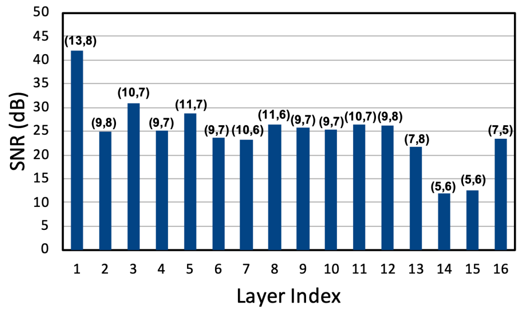
System simulations show that $M = 64$ trees tolerates a comparator error rate of 9.5% with undiscernible 8-class classification accuracy loss, whereas only 4% comparison error rate can be tolerated with $M = 4$. Therefore, ΔV_{lsb} = 15 mV and 8 mV are required to prevent accuracy degradation when $M = 4$ and 64, respectively (see Fig. 27). When operating with $M = 64$ on the prototype IC, ΔV_{lsb} can be reduced to 15 mV with minimal loss in accuracy while ΔV_{lsb} can only bre reduced to 20 mV when operating with $M = 4$ (see Fig. 28). Thus, the RF algorithm's inherent error tolerance enables its DIMA-based realization to achieve negligible accuracy degradation in spite of comparator errors.

The impact of process variations can be observed by measuring P_{det} over multiple (5) dies. Minor differences in P_{det} are observed, e.g., <2% with M = 64 in Fig. 29. This indicates that the ensemble nature of the RF algorithm provides inherent error tolerance to compensate process variations.

B.6 Energy Efficiency: The energy vs. accuracy trade-off with respect to 1) the number of trees (M), and 2) the BL voltage



(a)



(b)

Fig. 32. SNR requirements of dot-product in popular DNNs [83]: (a) box plots indicating layer-wise SNR requirements variability, and (b) per-layer SNR requirements and precision (B_w, B_x) of VGG-16 deployed on ImageNet. Precision requirements were obtained by a 1-step quantization of a floating-point network and hence are conservative.

swing ΔV_{lsb} are analyzed in Fig. 30. The energy savings in the prototype IC can be improved by reducing ΔV_{lsb} or M. However, controlling M allows a graceful trade-off between the energy consumption and P_{det}. For example, we can achieve 14.5× energy savings with less than 10% drop in P_{det} by tuning M, whereas reducing ΔV_{lsb} results in 68% degradation in P_{det} with only 1.7× reduction in energy. Therefore, choosing a smaller M always achieves better energy efficiency than reducing ΔV_{lsb} for a fixed classification accuracy P_{det}.

The prototype IC achieves the misclassification rate of 6% with an energy savings of 3.1× over the conventional architecture. One can expect the energy efficiency to improve in real-world tasks with a few hundreds trees, where ΔV_{lsb} can be reduced further. As expected, the face detection task, which is a binary classification, is more tolerant to the ΔV_{lsb} reduction than 8-class traffic sign recognition. This indicates that the prototype IC can achieve a smooth energy vs. accuracy trade-off via ΔV_{lsb} scaling based on the number of trees M, target accuracy, and the number of classes.

Figure 31 shows the energy and delay breakdowns estimated from post-layout simulations. The estimated results show 3.1× energy saving and 2.2× delay reduction leading to 6.8× EDP reduction over the digital system.

VI. Fundamental Energy-Delay-Accuracy Trade-offs

Previous sections have shown that DIMA can achieve large EDP reductions (see Figs. 14, 23, 31) over an equivalent digital architecture with a negligible loss in inference accuracy – a conclusion supported by measured results from multiple prototype ICs [25]–[28]. We have also seen that DIMA is a flexible architecture demonstrating an intrinsic energy vs. accuracy trade-off (see Figs. 13(c), 30). This trade-off is affected by the choices of several of DIMA design parameters: 1) the BL discharge voltage ΔV_{BL}, 2) the operand and ADC precision, 3) the BCA size ($N_{\text{ROW}} \times N_{\text{COL}}$), and 4) the ML workload, e.g., DNN size, number of classes, statistics of the dataset and others. Thus, the DIMA design space is complex and, while existing DIMA IC prototypes have demonstrated good choices for those design parameters, it is not clear what are the optimal choices and, therefore, the limits of energy-delay-accuracy for DIMA are.

In this section, we present a system-level framework to understand this fundamental trade-off and, therefore, the limits of energy-delay and accuracy. This framework begins by defining DIMA's compute SNR, followed by the development of silicon-validated models of the dot-product SNR, and finally by employing these models to find a favorable design space for DIMA to maximize its EDP gain for a given target accuracy.

TABLE II
Distortion and noise in DIMA [26].

Error type	FR (η_{F})	BLP (η_{B})	CBLP (η_{C})
% mean distortion	$2.6^{(1)}$	$2.1^{(1)}$	$0.8^{(3)}$
% noise (σ/μ)	(6 -to- 29)$^{(2)}$	$2.8^{(2)}$	$0.2^{(3)}$

- Row 1: average distortion over all 16 4-bit data values.
- Row 2: noise of maximum discharge $\Delta V_{\text{BL,max}}$.
- (1) measured from silicon prototype; (2) estimated from Monte Carlo simulations at $0.4\,\text{V} \leq V_{\text{WL}} \leq 0.8\,\text{V}$; (3) estimated from the size of capacitor in [26].

A. Modeling DIMA's Compute SNR

Consider the following N-dimensional dot-product:

$$y = \mathbf{w}^{\mathsf{T}}\mathbf{x} = \sum_{j=1}^{N} W_j X_j \tag{17}$$

where y is the ideal DP of two vectors $\mathbf{w}$ and $\mathbf{x}$ with N elements. Recall that DIMA's EDP gain is obtained at the expense of a loss in its compute SNR caused by its intrinsic analog nature. We explore the source of this SNR loss by analyzing the impact of various noise sources on the computation of a DP within DIMA.

Assuming a fixed total discharge time T, the DP computation (17) in the presence of circuit non-idealities is transformed as follows:

$$\widehat{y} = \frac{1}{N}\sum_{i=1}^{N}(W_i + g(W_i,T) + \eta_{\text{F},i})X_i + \eta_{\text{B}} + \eta_{\text{C}} \tag{18}$$

$$= y + \eta_{\text{y}} \tag{19}$$

where $g(W,T)$ is an additive term representing distortion and η_{F} is the spatial noise (variance σ^2_{wF}) in W due to FR. Here, η_{B} (variance σ^2_{B}) and η_{C} (variance σ^2_{C}) represent the additive noise terms due to non-ideal behavior in the BLP and CBLP, respectively. Finally, η_{y} is the composite noise as seen in y at the CBLP output.

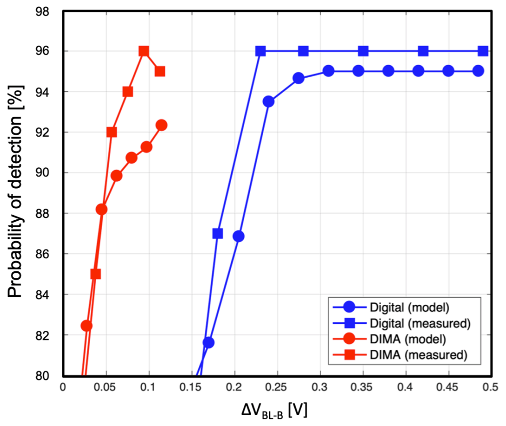


Fig. 33. Accuracy prediction model vs. silicon measured results [26] on probability of detection (P_{det}) with $N_{\text{ROW}} = 512$ for SVM. The MIT-CBCL dataset was used [64].

Table II quantifies noise contributions and distortion η_{F}, η_{B}, and η_{C} at the outputs of the FR, BLP, and CBLP stages, whose noise variances are $\sigma_{\text{F}}^2 \gg \sigma_{\text{B}}^2 \gg \sigma_{\text{C}}^2$. As shown in Fig. 18, the noise in FR dominates due to the minimum-sized transistors in the BC and near-threshold voltage operation caused by a low V_{WL} leading to a significant spatial mismatch. Using these models, the total compute SNR (SNR_{T}) can be estimated using:

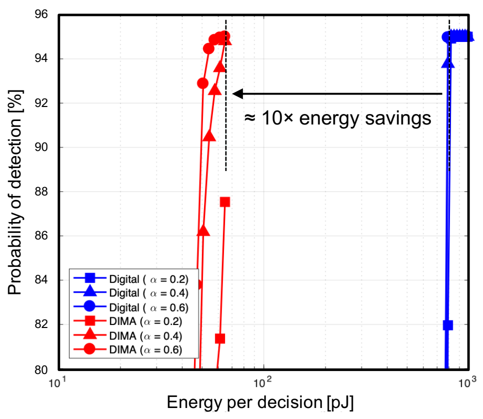
$$\text{SNR}_{\text{T}} = \frac{\mathbb{E}[y^2]}{\mathbb{E}[\eta_y^2]} = \frac{\mathbb{E}[y^2]}{\mathbb{E}[X^2]\sigma_{\text{wF}}^2/N + \mathbb{E}[D^2] + \sigma_{\text{B}}^2 + \sigma_{\text{C}}^2} \quad (20)$$

where $D = N^{-1}\sum_i^N g(W_i, T)X_i$ is the distortion reflected at the output.

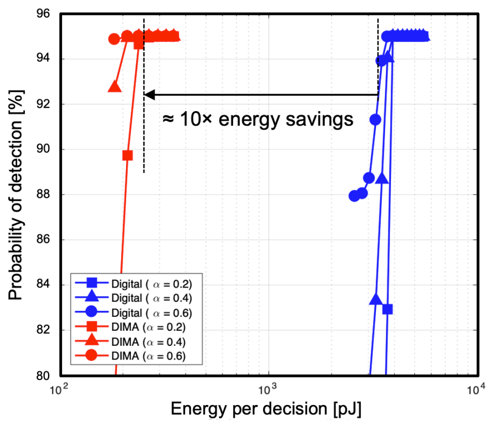
B. Compute SNR Requirements of DNNs

In order to estimate typical SNR requirements of DPs in a DNN, we leverage the analysis in [83], [84] that establishes theoretical guarantees on the worst-case degradation in the inference accuracy due to a 1-step quantization of activations and weights of a floating-point network. This method in [83] computes the noise gain from a quantization noise source from an internal node of the network to its output in order to estimate its impact on network accuracy without employing retraining.

Employing the method outlined above, the DP-level SNR requirements of four popular DNNs deployed on ImageNet and CIFAR-10 dataset were obtained under an accuracy loss budget of $\leq 1\%$ as shown in Fig. 32(a). This plot shows that the typical SNR requirements of DNN at the DP-level ranges between 10 dB-to-40 dB. To understand the distribution of the SNR requirements across individual layers in a network, we study the per-layer SNR requirements of VGG-16 deployed on ImageNet. Figure 32(b) shows that the DP SNR requirements decrease with layer depth and the precision requirements conservatively lie in the range as $5 \leq B_w \leq 13$ (weights) and $5 \leq B_x \leq 8$ (activations). These precision requirements can be reduced further, e.g., $\approx$ 4-bits, via retraining methods that search for better solutions in the quantized domain [85].



(a)



(b)

Fig. 34. Energy vs. decision accuracy trade-offs from accuracy prediction models (21) by sweeping $\Delta V_{\text{BL-B}}$ for SVM with $\alpha_{\text{SVM}} \in [0.2, 0.6]$ and vector length $N = 128$, and (b) $N = 512$ [64].

These trends provide another justification for the use of DIMA to realize DNNs since it has been shown that the analog computations are more energy efficient than the digital at lower SNRs (SNR $<$ 60 dB) [86], and because the deeper layers tend to be memory bound as the opportunities for weight reuse decrease.

C. Fundamental Trade-offs and Limits

The SNR modeling of DIMA operations in section VI-A allows us to study the fundamental trade-offs between accuracy vs. energy efficiency as a function of various algorithmic, architectural and circuit parameters such as the vector dimension N and the decision margins the array size (N_{ROW}). Furthermore, this allows us to determine conditions under which DIMA's EDP vs. accuracy trade-off is favorable. In this section, we present the general trends and trade-offs in DIMA, and our conclusions supported by analysis and simulations of the SVM algorithm implemented on DIMA.

Numerical values for the parameters of the energy and SNR models were obtained from circuit-level simulations on a 65 nm CMOS process and the measurements from the silicon prototype in [26] when possible. The total estimated energy

TABLE III
MODEL AND DESIGN PARAMETERS [64].

Parameter	Values	Parameter	Values
V_{DD} / V_{PRE}	1 V	V_{WL}	0.4 – 0.9 V
bit precision B	8	col. mux ratio L	4
# of rows N_{ROW}	256 - 1024	# of col.s N_{COL}	256
WL pulse width T_0	300 ps	vector length N	128 - 1024

from (9) and (10) are validated by comparing with the measurements from the silicon prototype [26]. In this section, the design parameters listed in Table III and the MIT CBCL dataset are considered.

C.1 Accuracy Trade-offs vs. Energy Efficiency: The maximum classification accuracy loss (p_{loss}) in the SVM algorithm as a function of the DP SNR (SNR_T) is governed by [64]:

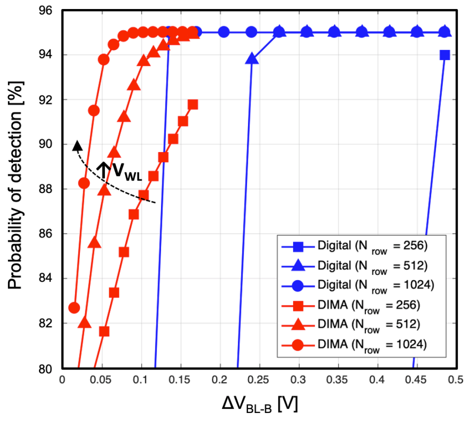
$$p_{\text{loss}} < Q\left(N\alpha_{\text{SVM}}\sqrt{\text{SNR}_\text{T}}\right) \tag{21}$$

where α_{SVM} is the margin in the SVM algorithm. We can use (21) along with the SNR analysis methodology in Section VI-A to estimate the accuracy of SVM algorithm implemented on DIMA. The SVM accuracy prediction model (21) applied on DIMA is validated by comparing its prediction with silicon measured results in [26] (see Fig. 33).

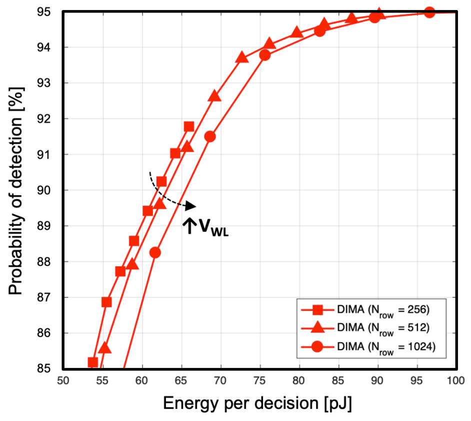
The SVM accuracy prediction model (21) used along side (9) and (10) allows us to relate the energy of SVM implementation on DIMA and digital architectures and to its system level accuracy. Figure 34 indicates that DIMA achieves approximately $10\times$ energy savings per decision at the same accuracy as compared to the digital architecture. Therefore, a $50\times$-to-$200\times$ EDP reduction is achieved over a digital system along with a $5\times$-to-$20\times$ delay reduction (see Section II-E). This observation correlates well with the $100\times$ EDP reduction from silicon prototypes in [28]. It is also seen that the accuracy P_{det} improves with the vector length N and decision margin α_{SVM}.

C.2 Impact of Array Size: The number of columns N_{COL} of the BCA is upper bounded by the need to keep rising/falling times of the WL enabling signals much smaller than the LSB pulse width T_0. This can be made difficult due to the increased WL capacitance and the row pitch-matching constraints. On the other hand, the upper bound on the number of rows N_{ROW}, impacts the energy and inference accuracy. This is because the BL capacitance C_{BL} increases with N_{ROW} leading to a higher energy consumption for the same BL discharge voltage ΔV_{BL}. Additionally, a higher WL enabling voltage V_{WL} is needed to increase the BC currents and hence maintain the same discharge time. This higher V_{WL} helps to alleviate the impact of transistor threshold voltage variations in the access transistors. These trends lead to improved accuracy (Fig. 35(a)) but with higher energy consumption (Fig. 35(b)) for both architectures. Figure 35 also indicates that DIMA requires sufficiently large N_{ROW}, e.g., $N_{\text{ROW}} = 256$, to achieve an accuracy equivalent to that of the digital system.

C.3 DIMA Design Space: The conditions under which DIMA's energy benefits can be maximized are summarized below:



(a)



(b)

Fig. 35. Inference accuracy vs. the number of rows N_{ROW} in the BC array with respect to: (a) BL swing per bit, and (b) decision energy for SVM with $\alpha_{\text{SVM}} = 0.4$ and $N = 128$ [64].

- For both DIMA and the digital architecture, there is a lower bound on the BL swing ΔV_{BL} beyond which the inference accuracy drops significantly. Figure 33 shows this lower bound to be $\approx 100\,\text{mV}$ for DIMA and $\approx 250\,\text{mV}$ for the digital architecture when $N_{\text{ROW}} = 512$. In addition, this lower bound is inversely proportional to N_{ROW}.
- DIMA's inference accuracy improves with a higher decision margin α_{SVM} due to the intrinsic error tolerance of the ML algorithms. Additionally, DIMA consumes $\approx 10\times$ less energy than the digital system at the same inference accuracy (Fig. 34).
- DIMA's inference accuracy improves with higher vector dimension N, but at a cost of higher energy consumption, e.g., the energy cost increases by $4\times$ when N increases from 128 to 512.
- DIMA cannot achieve an accuracy equivalent to that of a digital system when α_{SVM}, N or/and N_{ROW} are small.

To summarize, DIMA's efficiency is enhanced when the number of rows N_{ROW} and the vector length N are large, and for inference tasks with high decision margin α_{SVM}. As ML algorithms have inherent error tolerance, i.e., α_{SVM} is large, perform better with large vector lengths N, and have high storage requirements, i.e., large values of N_{ROW}, we conclude

TABLE IV
EXTENT TO WHICH IDEALIZED DIMA ATTRIBUTES ARE SATISFIED BY EXISTING CMOS DIMA ICS

	Ideal DIMA attributes				Remarks
	BCs	Row Parallelism/N_{ROW}	BL Compute	Delayed Decision	
Ideal	S-6T	$N_{\text{ROW}}/N_{\text{ROW}}$	QA / IA / QS	Yes	
[26]	S-6T	4/512	QA & QS	Yes	multi-bit, reconfigurable
[29]	10T	16/256	QS	Yes	binary-weighted
[25]	S-6T	81/128	QA	No	binary-weighted, AdaBoost
[32]	8T+Cap	512/512	QS	Yes	binarized
[59]	S-6T	4/512	QA & QS	Yes	multi-bit, on-chip learning
[27]	S-6T	4/512	QS	No	multi-bit, random forest
[31]	M-6T	64/64	QS	Yes	binarized
[33]	12T	256/256	IA	Yes	binarized
[34]	8T	64/64	QA & QS	Yes	multi-bit
[35]	8T+Cap	2304/2304	QS	No	multi-bit via digital combining
[36]	17T	148/148	QA	No	multi-bit via digital combining
[37]	M-6T	60/64	IA	Yes	binarized
[38]	M-6T	64/64	QA	Yes	binarized
[39]	8T	6/64	QA & QS	Yes	multi-bit
[40]	M-6T	32/32	QA	No	multi-bit via digital combining
[41]	8T	64/64	QA & QS	Yes	multi-bit
[42]	M-6T	32/512	QA	No	multi-bit via digital combining

S-6T: Standard 6T SRAM BC, M-6T: Modified 6T BC.
QA: Charge accumulation on a single capacitor, e.g., BL capacitor.
QS: Charge sharing across multiple capacitors.
IA: Current accumulation via multiple current sources.
Binary-weighted indicates that weights use 1-b precision.
Binarized indicates that both inputs and weights use 1-b precision.

that DIMA is well-suited for inference tasks.

VII. Discussion and Future Prospects

Since the publication of the DIMA concept paper [16], in-memory computing as a promising technology for bringing the power of AI into everyday lives. This section provides a comparison of various DIMA topologies and trends in DIMA design techniques, discusses the scalability of DIMA over process technology, and concludes with future prospects.

A. *Recent DIMA Trends and Comparison*

The attributes of an idealized DIMA (see Section II-A) include: (1) the use of a standard BCA, (2) row parallelism, (3) BL computations, and (4) delayed decision. While existing in-memory architectures strive to satisfy these principles, they fall short in order to either enhance scalability and/or robustness. The version of DIMA presented in this paper preserves: a) memory density by employing conventional 6T BC memory architecture, and b) generality by enabling multi-bit vector operations. Other variants of DIMA make alternative design choices such as: 1) restricting one or both operands to binary (1-bit) precision and combining them in digital [35], 2) using larger (8T or 10T) BCs [29], 3) using specialized BCs that enable computations [32], 4) partitioning the array into sub-banks [29], [42], and 5) using separate right and left WLs [31]. Table IV compares existing DIMAs in terms of the attributes of an idealized DIMA. One can see that achieving maximum row-parallelism is specially challenging if BL processing of multi-bit weights needs to be done.

B. *Scalability of DIMA over Process Technology*

As CMOS process scales, we expect improved energy efficiency and throughput due to lower capacitance and lower supply voltage. For example, post layout simulations show $4\times$ smaller BL capacitance in a 28 nm technology as compared to a 65 nm technology. This leads to $4\times$ smaller BL precharge energy (assuming the same V_{DD} and $\Delta V_{\text{BL,max}}$), which dominates the energy consumption in DIMA. The smaller BL capacitance also leads to lower delay.

However, DIMA's mixed-signal nature makes it vulnerable to multiple noise sources in advanced process nodes. For example, increased coupling noise due to strict layout constraints can lead to reduced accuracy in charge-domain computations. Thus, the area overhead of shielding in layouts is expected to increase as shown in [26]. The advanced nodes also suffer from the increased process variations which incur higher σ/μ of $\Delta V_{\text{BL,max}}$ leading to the decision accuracy degradation. In fact, one can emulate operation in an advanced node by reducing the BL voltage swing. For example, σ/μ in Fig. 18 is around 9% at the default $\Delta V_{\text{BL max}}$ = 550 mV, but it reaches 18% by scaling down $\Delta V_{\text{BL max}}$ to 320 mV. One can expect an equivalent impact of increased variations in a highly scaled process technology.

The equation (21) indicates that the loss of inference accuracy can be recovered by increasing $\Delta V_{\text{BL,max}}$ and/or N at the cost of increased energy consumption. Alternatively, the impact of increased variations can be overcome via on-chip training as shown in Fig. 22(b), where the inference accuracy at $\Delta V_{\text{BL max}}$ = 320 mV is is equal to that at $V_{\text{BL max}} = 550$ mV. The inference accuracy under low BL swing can also be enhanced by using a classifier ensemble as shown in Fig. 28 with ΔV_{lsb} = 20 mV.

These observations indicate that there are multiple algorithmic, architectural and circuit design approaches that can help to overcome the impact of increased process variations on DIMA's accuracy. Therefore, there exists an interesting trade-off between the EDP gains from the technology scaling and the loss in accuracy due to increased variations. This trade-off needs further study.

C. *Future Prospects*

Though DIMA provides significant energy and delay benefits, it has a number of challenges and drawbacks including: a) an intrinsically limited compute SNR due to its mixed-signal nature; b) mismatch between the array size and problem size due to the need for processing data within the BCA; c) difficulty in exploiting data reuse opportunities due the difficulty in realizing reliable analog storage; and d) the need to realize non-MVM computations. Each of these challenges point to research directions that can be explored.

1) Scalable Deep In-memory Architectures: To map large-scale applications onto a DIMA-based platform, one can explore methods to integrate DIMA into large-scale heterogeneous architectures such as systolic ML accelerators. Designing software infrastructure to augment such DIMA-based platforms will be also essential in order to map high-level application metrics, e.g., inference accuracy, to low-level hardware design parameters. Some preliminary work has already been demonstrated in [61].

2) Error Resilient DIMA: In this paper, we introduced two approaches for enhancing DIMA's resiliency to process variations by: 1) employing on-chip training in Section V-A, and 2) exploiting the ensemble classifier in Section V-B to overcome the various sources of non-idealities, e.g., process variation, voltage or temperature fluctuations.

However, there are a number of opportunities in both algorithmic, architectural and circuit domains, that can be explored including: 1) employing error compensation techniques such as error correcting codes (ECCs) as well as Shannon-inspired statistical error compensation (SEC) [68] to achieve greater reductions in EDP; 2) accuracy-optimal resource assignment [87], where more resources such as voltage swing, execution time, and area are assigned to critical components of inference computations so that accuracy is maximized within an energy budget; 3) foreground calibration techniques [32] for chip-specific tuning of circuit parameters before its use for inference; and 4) robust ML model training [88] so that the trained network itself becomes robust to variations and errors.

3) DIMA in Emerging Memory Technologies: There are multiple emerging memory device technologies beyond flash, DRAM, and SRAM, e.g., PCM, RRAM, MRAM, and others. These new devices provide fundamentally new design trade-offs for in-memory techniques and create opportunities for device-architecture co-design.

In summary, DIMA is a high-density, low-latency, energy efficient computational platform for realizing AI systems. DIMA's enormous EDP gains ($> 100\times$) over the von Neumann architecture arise primarily from its ability to deeply integrate read and compute functions in analog in the memory core. However, in doing so, DIMA exhibits an intrinsic compute SNR vs. EDP trade-off which manifests itself as a system-level trade-off between inference accuracy, energy and latency. In this process, DIMA represents an analog approach to approximate computing encompassing applications, systems/algorithms, architectures, circuits and devices. This full-stack nature of DIMA provides numerous collaboration opportunities for application developers, system/algorithm designers, architects, circuit designers, and researchers in semiconductor devices to work together in addressing challenges in the design of AI systems of the future.

Acknowledgement

This work was supported by the Systems On Nanoscale Information fabriCs (SONIC) and the Center for Brain-Inspired Computing (C-BRIC) funded by the Semiconductor Research Corporation (SRC) and the Defense Advanced Research Projects Agency (DARPA). We thank Charbel Sakr for the analysis of SNR and bit precision requirement in deep neural networks in Section VI-B.

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

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

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

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

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

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

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

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

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

[10] S. Aga, S. Jeloka, A. Subramaniyan, S. Narayanasamy, D. Blaauw, and R. Das, "Compute caches," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2017, pp. 481–492.

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

[12] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, "An always-on 3.8uJ/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 158–172, 2018.

[13] J. Jeddeloh and B. Keeth, "Hybrid memory cube new DRAM architecture increases density and performance," in *2012 symposium on VLSI technology (VLSIT)*. IEEE, 2012, pp. 87–88.

[14] M. M. Shulaker, T. F. Wu, A. Pal, L. Zhao, Y. Nishi, K. Saraswat, H.-S. P. Wong, and S. Mitra, "Monolithic 3D integration of logic and memory: Carbon nanotube FETs, resistive RAM, and silicon FETs," in *2014 IEEE International Electron Devices Meeting*. IEEE, 2014, pp. 27–4.

[15] S. Jeloka, N. B. Akesh, D. Sylvester, and D. Blaauw, "A 28 nm configurable memory (TCAM/BCAM/SRAM) using push-rule 6T bit cell enabling logic-in-memory," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 4, pp. 1009–1021, 2016.

[16] 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.

[17] M. Kang, S. K. Gonugondla, and N. Shanbhag R., *Deep In-memory Architectures for Machine Learning*. Springer, 2020.

[18] N. Verma, H. Jia, H. Valavi, Y. Tang, M. Ozatay, L.-Y. Chen, B. Zhang, and P. Deaville, "In-memory computing: Advances and prospects," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 3, pp. 43–55, 2019.

[19] P.-T. Huang and W. Hwang, "A 65 nm 0.165 fj/bit/search 256$\times$144 TCAM macro design for IPv6 lookup tables," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 2, pp. 507–519, 2010.

[20] K. Nii, T. Amano, N. Watanabe, M. Yamawaki, K. Yoshinaga, M. Wada, and I. Hayashi, "A 28nm 400MHz 4-parallel 1.6 Gsearch/s 80Mb ternary CAM," in *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE, 2014, pp. 240–241.

[21] J. Li, R. Montoye, M. Ishii, K. Stawiasz, T. Nishida, K. Maloney, G. Ditlow, S. Lewis, T. Maffitt, R. Jordan *et al.*, "1Mb 0.41 μm 2 2T-2R cell nonvolatile TCAM with two-bit encoding and clocked self-referenced sensing," in *Symposium on VLSI Technology (VLSI-T)*. IEEE, 2013, pp. C104–C105.

[22] S. Matsunaga, S. Miura, H. Honjou, K. Kinoshita, S. Ikeda, T. Endoh, H. Ohno, and T. Hanyu, "A 3.14 um 2 4T-2MTJ-cell fully parallel

TCAM based on nonvolatile logic-in-memory architecture," in *Symposium on VLSI Circuits (VLSI-C)*. IEEE, 2012, pp. 44–45.

[23] L.-Y. Huang, M.-F. Chang, C.-H. Chuang, C.-C. Kuo, C.-F. Chen, G.-H. Yang, H.-J. Tsai, T.-F. Chen, S.-S. Sheu, K.-L. Su *et al.*, "ReRAM-based 4T2R nonvolatile TCAM with 7x NVM-stress reduction, and 4x improvement in speed-wordlength-capacity for normally-off instant-on filter-based search engines used in big-data processing," in *Symposium on VLSI Circuits (VLSI-C)*. IEEE, 2014, pp. 1–2.

[24] N. Shanbhag, M. Kang, and M.-S. Keel, *Compute Memory*. Issued July 4 2017, US Patent 9,697,877 B2.

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

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

[27] M. Kang, S. K. Gonugondla, S. Lim, and N. R. Shanbhag, "A 19.4-nJ/decision, 364-K decisions/s, in-memory random forest multi-class inference accelerator," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 53, no. 7, pp. 2126–2135, July 2018.

[28] S. K. Gonugondla, M. Kang, and N. R. Shanbhag, "A variation-tolerant in-memory machine learning classifier via on-chip training," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 53, no. 11, pp. 3163–3173, Nov. 2018.

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

[30] W.-H. Chen, K.-X. Li, W.-Y. Lin, K.-H. Hsu, P.-Y. Li, C.-H. Yang, C.-X. Xue, E.-Y. Yang, Y.-K. Chen, Y.-S. Chang *et al.*, "A 65nm 1Mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2018, pp. 494–496.

[31] W.-S. Khwa, J.-J. Chen, J.-F. Li, X. Si, E.-Y. Yang, X. Sun, R. Liu, P.-Y. Chen, Q. Li, S. Yu *et al.*, "A 65nm 4kb algorithm-dependent computing-in-memory SRAM unit-macro with 2.3 ns and 55.8 TOPS/W fully parallel product-sum operation for binary DNN edge processors," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2018, pp. 496–498.

[32] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-tile 2.4-Mb in-memory-computing CNN accelerator employing charge-domain compute," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 6, pp. 1789–1799, 2019.

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

[34] X. Si, J.-J. Chen, Y.-N. Tu, W.-H. Huang, J.-H. Wang, Y.-C. Chiu, W.-C. Wei, S.-Y. Wu, X. Sun, R. Liu *et al.*, "A twin-8T SRAM computation-in-memory macro for multiple-bit CNN-based machine learning," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2019, pp. 396–398.

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

[36] S. Okumura, M. Yabuuchi, K. Hijioka, and K. Nose, "A ternary based bit scalable, 8.80 TOPS/W CNN accelerator with many-core processing-in-memory architecture with 896k synapses/mm2," in *2019 IEEE Symposium on VLSI Circuits*. IEEE, 2019, pp. 248–249.

[37] J. Kim, J. Koo, T. Kim, Y. Kim, H. Kim, S. Yoo, and J.-J. Kim, "Area-efficient and variation-tolerant in-memory BNN computing using 6T SRAM array," in *IEEE Symposium on VLSI Circuits (VLSI-C)*. IEEE, 2019, pp. 118–119.

[38] R. Guo, Y. Liu, S. Zheng, S.-Y. Wu, P. Ouyang, W.-S. Khwa, X. Chen, J.-J. Chen, X. Li, L. Liu, M.-F. Chang, S. Wei, and S. Yin, "A 5.1pJ/neuron 127.3us/inference RNN-based speech recognition processor using 16 computing-in-memory SRAM macros in 65nm CMOS," in *2019 IEEE Symposium on VLSI Circuits*. IEEE, 2019, pp. 120–121.

[39] J. Yue, Z. Yuan, X. Feng, Y. He, Z. Zhang, X. Si, R. Liu, M.-F. Chang, X. Li, H. Yang, and Y. Liu, "A 65nm computing-in-memory-based CNN processor with 2.9-to-35.8TOPS/W system energy efficiency using dynamic-sparsity performance-scaling architecture and energy-efficient inter/intra-macro data reuse," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 234–235.

[40] J.-W. Su, X. Si, Y.-C. Chou, T.-W. Chang, W.-H. Huang, Y.-N. Tu, R. Liu, T.-W. Lu, Pei-Jungand Liu, J.-H. Wang, Z. Zhang, H. Jiang, S. Huang, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, S.-S. Sheu, S.-H. Li, H.-Y. Lee, S.-C. Chang, S. Yu, and M.-F. Chang, "A 28nm 64Kb inference-training two-way transpose multibit 6T SRAM compute-in-memory macro for AI edge chips," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 240–241.

[41] Q. Dong, M. E. Sinangil, B. Erbagci, D. Sun, W.-S. Khwa, H.-J. Liao, Y. Wang, and J. Chang, "A 351 TOPS/W and 372.4 GOPS compute-in-memory sram macro in 7nm FinFET CMOS for machine learning applications," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 242–243.

[42] X. Si, Y.-N. Tu, W.-H. Huang, J.-W. Su, P.-J. Lu, J.-H. Wang, T.-W. Liu, S.-Y. Wu, R. Liu, Y.-C. Chou, Z. Zhang, S.-H. Sie, W.-C. Wei, Y.-C. Lo, T.-H. Wen, T.-H. Hsu, Y.-K. Chen, W. Shih, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, N.-C. Lien, W.-C. Shih, Y. He, Q. Li, and M.-F. Chang, "A 28nm 64Kb 6T SRAM computing-in- memory macro with 8b MAC operation for AI edge chips," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 246–247.

[43] C.-X. Xue, T.-Y. Huang, J.-S. Liu, T.-W. Chang, H.-Y. Kao, J.-H. Wang, T.-W. Liu, S.-Y. Wei, S.-P. Huang, W.-C. Wei, Y.-R. Chen, T.-H. Hsu, Y.-K. Chen, Y.-C. Lo, T.-H. Wen, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, and M.-F. Chang, "A 22nm 2Mb ReRAM compute-in-memory macro with 121-28TOPS/W for multibit MAC computing for tiny AI edge devices," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 244–245.

[44] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[45] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, "Training deep neural networks with 8-bit floating point numbers," in *Advances in neural information processing systems*, 2018, pp. 7675–7684.

[46] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in neural information processing systems*, 2016, pp. 4107–4115.

[47] V. Mrazek, S. S. Sarwar, L. Sekanina, Z. Vasicek, and K. Roy, "Design of power-efficient approximate multipliers for approximate artificial neural networks," in *Proceedings of the 35th International Conference on Computer-Aided Design*, 2016, pp. 1–7.

[48] S. S. Sarwar, S. Venkataramani, A. Raghunathan, and K. Roy, "Multiplier-less artificial neurons exploiting error resiliency for energy-efficient neural computing," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2016, pp. 145–150.

[49] S. H. Nawab, A. V. Oppenheim, A. P. Chandrakasan, J. M. Winograd, and J. T. Ludwig, "Approximate signal processing," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 15, no. 1-2, pp. 177–200, 1997.

[50] J. T. Ludwig, S. H. Nawab, and A. P. Chandrakasan, "Low-power digital filtering using approximate processing," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 3, pp. 395–400, 1996.

[51] N. R. Shanbhag, R. A. Abdallah, R. Kumar, and D. L. Jones, "Stochastic computation," in *Proceedings of the 47th Design Automation Conference*, 2010, pp. 859–864.

[52] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM Transactions on Embedded computing systems (TECS)*, vol. 12, no. 2s, pp. 1–19, 2013.

[53] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, "Approximate XOR/XNOR-based adders for inexact computing," in *IEEE International Conference on Nanotechnology (IEEE-NANO 2013)*, 2013, pp. 690–693.

[54] J. Miao, A. Gerstlauer, and M. Orshansky, "Multi-level approximate logic synthesis under general error constraints," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2014, pp. 504–510.

[55] ——, "Approximate logic synthesis under general error magnitude and frequency constraints," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2013, pp. 779–786.

[56] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in neural information processing systems*, 2016, pp. 2074–2082.

[57] 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)*, 2016, pp. 260–261.

[58] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Transactions*

on Computer-Aided Design of Integrated Circuits and Systems, vol. 32, no. 1, pp. 124–137, 2012.

[59] S. K. Gonugondla, M. Kang, and N. Shanbhag, “A 42pJ/decision 3.12 TOPS/W robust in-memory machine learning classifier with on-chip training,” in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2018, pp. 490–492.

[60] J. Zhang, Z. Wang, and N. Verma, “A machine-learning classifier implemented in a standard 6T SRAM array,” in *IEEE Symposium on VLSI Circuits (VLSI Circuits)*, 2016, pp. 1–2.

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

[62] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[63] Y. LeCun, D. Touresky, G. Hinton, and T. Sejnowski, “A theoretical framework for back-propagation,” in *Proceedings of the 1988 connectionist models summer school*, vol. 1. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988, pp. 21–28.

[64] M. Kang, Y. Kim, A. D. Patil, and N. R. Shanbhag, “Deep in-memory architectures for machine learning–accuracy versus efficiency trade-offs,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 5, pp. 1627–1639, 2020.

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

[66] 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.

[67] D. Bankman and B. Murmann, “An 8-bit, 16 input, 3.2 pJ/op switched-capacitor dot product circuit in 28-nm FDSOI CMOS,” in *IEEE Asian Solid-State Circuits Conference (A-SSCC)*, 2016, pp. 21–24.

[68] N. R. Shanbhag, N. Verma, Y. Kim, A. D. Patil, and L. R. Varshney, “Shannon-inspired statistical computing for the nanoscale era,” *Proceedings of the IEEE*, vol. 107, no. 1, pp. 90–107, 2019.

[69] P. Srivastava, M. Kang, S. K. Gonugondla, S. Lim, J. Choi, V. Adve, N. S. Kim, and N. Shanbhag, “PROMISE: An end-to-end design of a programmable mixed-signal accelerator for machine-learning algorithms,” in *Proceedings of the 45th Annual International Symposium on Computer Architecture*. IEEE Press, 2018, pp. 43–56.

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

[71] D. Liu, T. Chen, S. Liu, J. Zhou, S. Zhou, O. Teman, X. Feng, X. Zhou, and Y. Chen, “PuDianNao: A polyvalent machine learning accelerator,” in *ACM SIGARCH Computer Architecture News*, vol. 43, no. 1, 2015, pp. 369–381.

[72] Z. Zhou, B. Pain, and E. R. Fossum, “CMOS active pixel sensor with on-chip successive approximation analog-to-digital converter,” *IEEE Transactions on Electron Devices*, vol. 44, no. 10, pp. 1759–1763, 1997.

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

[74] B. Murmann, D. Bankman, E. Chai, D. Miyashita, and L. Yang, “Mixed-signal circuits for embedded machine-learning applications,” in *IEEE 49th Asilomar Conference on Signals, Systems and Computers*, 2015, pp. 1341–1345.

[75] “Center for biologicaland computationallearning (CBCL) at MIT,” 2000, http://cbcl.mit.edu/software-datasets/index.html.

[76] Y. Freund and R. E. Schapire, “A desicion-theoretic generalization of on-line learning and an application to boosting,” in *European conference on computational learning theory*. Springer, 1995, pp. 23–37.

[77] L. Breiman, “Bagging predictors,” *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[78] ——, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[79] M. Kang, S. K. Gonugondla, and N. R. Shanbhag, “A 19.4 nJ/decision 364K decisions/s in-memory random forest classifier in 6T SRAM array,” in *IEEE European Solid-State Circuits Conference (ESSCIRC)*, 2017, pp. 263–266.

[80] T. Kobayashi, K. Nogami, T. Shirotori, and Y. Fujimoto, “A current-controlled latch sense amplifier and a static power-saving input buffer for low-power architecture,” *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 76, no. 5, pp. 863–867, 1993.

[81] V. A. Prisacariu, R. Timofte, K. Zimmermann, I. Reid, and L. Van Gool, “Integrating object detection with 3D tracking towards a better driver assistance system,” in *IEEE International Conference on Pattern Recognition (ICPR)*, 2010, pp. 3344–3347.

[82] B. Van Essen, C. Macaraeg, M. Gokhale, and R. Prenger, “Accelerating a random forest classifier: Multi-core, GP-GPU, or FPGA?” in *IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2012, pp. 232–239.

[83] C. Sakr, Y. Kim, and N. Shanbhag, “Analytical guarantees on numerical precision of deep neural networks,” in *International Conference on Machine Learning (ICML)*, 2017, pp. 3007–3016.

[84] C. Sakr and N. Shanbhag, “An analytical method to determine minimum per-layer precision of deep neural networks,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 1090–1094.

[85] J. Choi, S. Venkataramani, V. Srinivasan, K. Gopalakrishnan, Z. Wang, and P. Chuang, “Accurate and efficient 2-bit quantized neural networks,” in *Proceedings of the SysML Conference*, 2019.

[86] R. Sarpeshkar, “Analog versus digital: extrapolating from electronics to neurobiology,” *Neural computation*, vol. 10, no. 7, pp. 1601–1638, 1998.

[87] F. Frustaci, M. Khayatzadeh, D. Blaauw, D. Sylvester, and M. Alioto, “SRAM for error-tolerant applications with dynamic energy-quality management in 28 nm CMOS,” *IEEE Journal of Solid-state circuits*, vol. 50, no. 5, pp. 1310–1323, 2015.

[88] B. Zhang, L.-Y. Chen, and N. Verma, “Stochastic data-driven hardware resilience to efficiently train inference models for stochastic hardware implementations,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 1388–1392.

Mingu Kang (M’13) is an assistant professor of Electrical and Computer Engineering at the University of California at San Diego (UCSD), La Jolla, CA, USA since 2020. He received the B.S. and M.S. degrees in Electrical and Electronic Engineering from Yonsei University, Seoul, South Korea, in 2007 and 2009, respectively, and the Ph.D. degree in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign, Urbana, 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. From 2017 to 2020, he was with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA, where he worked for machine learning accelerator architectures. He is a recipient of MICRO TOP Pick Honorable Mention 2019, IEEE International Symposium on Circuits and Systems (ISCAS) best Paper Awards in 2016 and 2018, University of Illinois Coordinated Science Laboratory (CSL) best thesis award in 2018, and Kwanjeong Scholarship from 2012 to 2016. His current research interests include low-power integrated circuits, architecture, and system for machine learning and signal processing by leveraging non-von Neumann approaches including in-memory, in-sensor, and neuromorphic computing with both CMOS and emerging devices.

Sujan Gonugondla (S'16) received the Bachelor's and Master's in Technology degrees in Electrical Engineering from the Indian Institute of Technology Madras, Chennai, India, in 2014 and the Ph.D. degree in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign, Urbana, IL, USA in 2020. Since June 2020, he has been with Amazon where he works as a Research Scientist. His research interests are in energy-efficient integrated circuits, and low complexity algorithms for machine learning systems, specifically algorithm hardware co-design for inference under resource-constraints.

Sujan K. Gonugondla is a recipient of the Dr. Ok Kyun Kim Fellowship 2018-19 and the M. E. Van Valkenburg Graduate Research Award 2019-20 from the ECE department at the University of Illinois at Urbana-Champaign, the ADI Outstanding Student Designer Award 2018 and the SSCS Predoctoral Achievement award in 2020. He has received Best Student Paper Awards in International Conference on Acoustics, Speech and Signal Processing (ICASSP) in 2016, and International conference in Circuits and Systems (ISCAS) in 2018.

Naresh R. Shanbhag (F'06) is the Jack Kilby Professor of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. He received his Ph.D. degree from the University of Minnesota (1993) in Electrical Engineering. From 1993 to 1995, he worked at AT&T Bell Laboratories at Murray Hill where he led the design of high-speed transceiver chip-sets for very high-speed digital subscriber line (VDSL), before joining the University of Illinois at Urbana-Champaign in August 1995. He has held visiting faculty appointments at the National Taiwan University (Aug.-Dec. 2007) and Stanford University (Aug.-Dec. 2014). His research focuses on the design of energy-efficient systems for machine learning, communications, and signal processing spanning algorithms, architectures and integrated circuits. He has more than 200 publications in this area, holds thirteen US patents, and is a co-author of two books and multiple book chapters.

Dr. Shanbhag received the 2018 SIA/SRC University Researcher Award, became an IEEE Fellow in 2006, received the 2010 Richard Newton GSRC Industrial Impact Award, the IEEE Circuits and Systems Society Distinguished Lecturership in 1997, the National Science Foundation CAREER Award in 1996, and multiple best paper awards including the 2006 IEEE Solid-State Society Best Paper of the Year Award. In 2000, Dr. Shanbhag co-founded and served as the Chief Technology Officer of the Intersymbol Communications, Inc., which introduced mixed-signal ICs for electronic dispersion compensation of OC-192 optical links, and became a part of Finisar Corporation in 2007. From 2013-17, he was the founding Director of the Systems On Nanoscale Information fabriCs (SONIC) Center, a 5-year multi-university center funded by DARPA and SRC under the STARnet program.