Algorithm Hardware Co-Design for ADC-Less Compute In-Memory Accelerator

Shubham Negi, *Student Member, IEEE*, Utkarsh Saxena, Deepika Sharma, *Graduate Student Member, IEEE*, Jeffry Victor, Imtiaz Ahmed, Sumeet Kumar Gupta, *Senior Member, IEEE*, and Kaushik Roy, *Fellow, IEEE*

***Abstract*—The increasing pervasiveness of artificial intelligence (AI), particularly deep learning demands high-performing yet efficient hardware resources at the edge. Analog compute-in-memory (CiM) architectures have tremendous potential to accelerate AI at the edge by reducing data movement between memory and compute units and exploiting parallelism. However, to fully reap the benefits of analog CiM, it is imperative to deal with the area, latency, and power overheads introduced by high-precision analog-to-digital converters (ADCs). In this work, we propose a hardware-algorithm co-design approach to reduce ADC overhead in analog CiM architectures. We designed a deep neural network (DNN) quantization framework tailored to analog CiM hardware architectures, integrating essential features such as tiling, bit-slicing, and layer mapping. Moreover, we also developed an ADC-Less hybrid analog-digital CiM hardware architecture HCiM that can efficiently process the DNNs trained using our framework. Additionally, we studied the effects of non-idealities in analog CiM on DNN accuracy. Using our hardware-aware training methodology, we can perform extremely low precision quantization and reduce the required ADC precision to binary (1-bit) or ternary (1.5-bit). Compared to an analog CiM baseline architecture using 7 and 4-bit ADC, HCiM achieves energy reductions up to 28× and 12×, respectively. Furthermore, in the presence of analog non-idealities, DNN mapped to HCiM exhibits a minimal drop in accuracy.**

***Index Terms*—Analog compute-in-memory (CiM), quantization-aware training, ADC-less design, binary and ternary quantization.**

I. Introduction

THE relentless pursuit of greater performance and functionality in artificial intelligence (AI) has fueled the development of increasingly complex neural network architectures [1], [2]. This growth in model size and complexity significantly increases the demand for powerful hardware resources for both cloud and edge applications [1]. However, it is challenging to meet this demand at the edge, where the hardware is constrained by a stringent power and area budget. Additionally, the increasing disparity between data movement and computation energy in modern technology nodes makes data movement a bottleneck operation in von Neumann architectures [3], [4]. Analog compute-in-memory (CiM) offers a compelling solution to this problem by directly embedding computation within memory elements [5], [6]. It reduces the data movement bottleneck and exploits the inherent parallelism of the memory array to efficiently perform the fundamental DNN operation – matrix-vector multiplication (MVM).

Although CiM accelerators exhibit excellent performance at the array level, these advantages often do not extend to the system level, primarily due to the peripheral overhead. Researchers have shown that peripheral circuits, particularly analog-to-digital converters (ADCs) consume significant area/power in CiM designs [6], [7], [8]. Moreover, to maximize the parallelism of matrix-vector multiplications in CiM, ideally, each column of the memory array requires a dedicated ADC; however, high ADC area creates area/layout challenges, necessitating column multiplexing which reduces overall throughput [6], [9], [10]. Therefore, there is a need to reduce the ADC overhead to leverage the full potential of analog CiM architectures.

A direct approach to reducing ADC overhead is to reduce its precision, leading to improvements in area, power, and latency. The required ADC precision depends on the desired precision of partial sums from the memory subarray, which is influenced by the number of simultaneously active rows and the precision of individual memory cells. While reducing the number of active rows in a crossbar array can lower the ADC precision requirements [11], such an approach sacrifices parallelism and ultimately impacts system throughput. Prior efforts to reduce ADC precision in CiM hardware have explored techniques such as exploiting input sparsity [9], [12] and quantizing partial sums using non-linear quantization [13], [14] or linear quantization with a smaller dynamic range [15]. However, these hardware-only approaches face limitations in how aggressively they can reduce ADC precision without compromising DNN accuracy.

Consequently, there has been a growing interest in training the neural network with partial sum quantization (PSQ) [16],

Received 20 May 2024; revised 10 October 2024; accepted 28 October 2024. Date of publication 11 November 2024; date of current version 26 November 2024. This work was supported by the Center for the CoDesign of Cognitive Systems (COCOSYS), by the DARPA sponsored JUMP center of Semiconductor Research Corporation (SRC), SRC AIHW, and by the National Science Foundation, Intel and Department of Energy (DoE). The review of this article was arranged by Associate Editor Mingoo Seok. *(Shubham Negi and Utkarsh Saxena contributed equally to this work.) (Corresponding author: Shubham Negi.)*

The authors are with Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: snegi@purdue.edu).

Digital Object Identifier 10.1109/TCASAI.2024.3495587

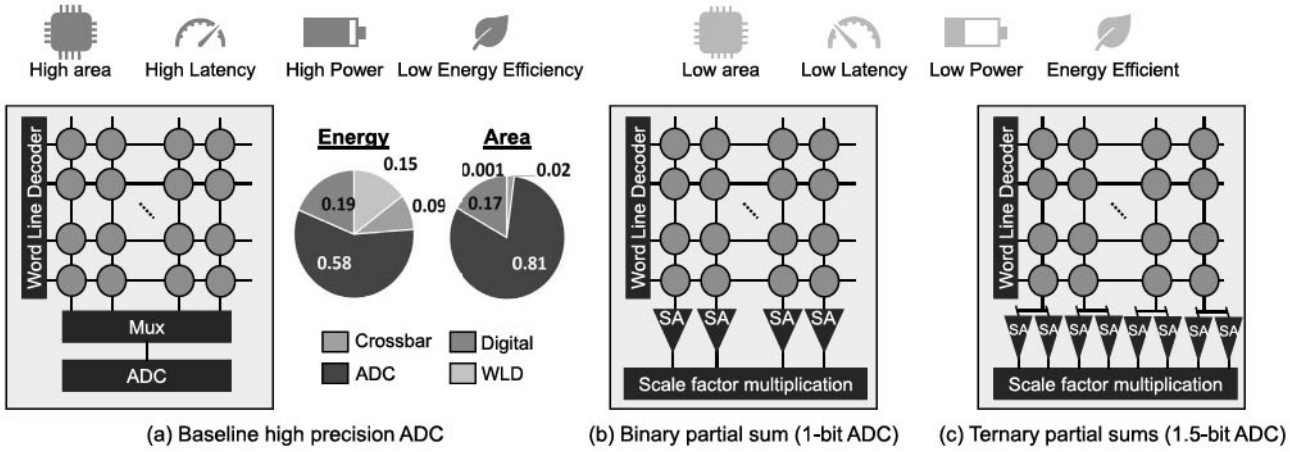


Fig. 1. Overview of ADC-Less CiM design. (a) Baseline with high precision ADC has low energy efficiency, latency, throughput and high power consumption. (b) ADC-Less design with binary partial sum quantization requires only a sense amplifier for analog to digital conversion and (c) ADC-Less design with ternary partial sums require two comparators for the conversion. ADC-Less CiM hardware alleviates the ADC bottleneck leading to improvements in hardware performance.

[17], [18], [19]. DNNs trained with partial sum quantization are inherently robust to the reduction in ADC precision in CiM hardware. We take this idea to the extreme level where we develop PSQ-aware training techniques for extremely low partial sum precision to Binary (1-bit) and Ternary (1.5-bit) levels [20]. ADC-Less[1] DNNs with binary partial sums require just a sense amplifier for analog to digital conversion in hardware while ADC-Less DNNs with ternary partial sums require two comparators for the conversion.

An artifact of partial sum quantization is the introduction of scale factors. The scaling factors are essential to align the range of quantized partial sums with the actual floating point data. While employing scale factors at a finer granularity results in better accuracy, it increases the number of scale factors required and their storage and management complexity. Additionally, these scale factors are typically floating-point values [16], [17], [18], [20], and processing them requires complex floating-point operations in hardware, which increases computational overhead. These considerations underscore the need for careful management and optimization of scale factors to balance accuracy, efficiency, and computational complexity in PSQ techniques.

We adopted a two-part strategy to address the challenges of scale factor processing. First, we quantize the scale factors to a few bits during training. Second, we propose a digital CiM (DCiM) array to process these quantized scale factors efficiently. Moreover, we developed an ADC-Less hybrid analog-digital CiM accelerator architecture HCiM [21]. This design integrates an analog CiM crossbar that performs the MVM operation between inputs and weights, generating quantized partial sums that are then processed by the DCiM array for scale factor multiplication.

In this work, we propose a full-stack approach to ADC-Less DNN implementation, spanning from quantization-aware training to customized hardware architecture design, ensuring accuracy and efficiency. The overview of our hardware-software co-design approach is depicted in Fig. 1. We make the following contributions in this work:

- We developed a DNN quantization framework tailored to analog CiM hardware architectures, integrating essential features like tiling, bit-slicing, and layer mapping.
- We proposed scale factor quantization-aware training and integrated it with our framework to reduce the complexity of processing scale factors.
- We achieve state-of-the-art accuracy at extreme (1-bit and 1.5-bit) partial sum quantization using our proposed training framework.
- We designed an ADC-Less hybrid analog-digital CiM accelerator HCiM which uses an analog CiM array for MVM operations and a digital-CiM array with a novel in-memory subtraction technique for scale factor multiplication.
- We achieve a 15% reduction in energy through clock gating in the proposed DCiM peripherals by utilizing the inherent sparsity in ternary quantized partial sums.
- We evaluate HCiM using a cycle-accurate DNN simulator across multiple workloads, demonstrating its efficacy against baselines.
- We evaluate the impact of crossbar non-idealities on our proposed ADC-Less hardware architecture.

The remainder of this paper is organized as follows. Section II provides the necessary background information. Section III presents a comprehensive review of related works. Section IV details our proposed hardware and algorithm co-design approach for ADC-Less CiM designs. Evaluation results are discussed in Section V, and we conclude in Section VI.

II. Background

A. Matrix Multiplication Using Analog Crossbar Array

Matrix multiplication serves as a fundamental operation in DNNs. An effective and common method of accelerating matrix

[1]ADC-Less term means that our approach relies on sense amplifiers instead of conventional ADCs.

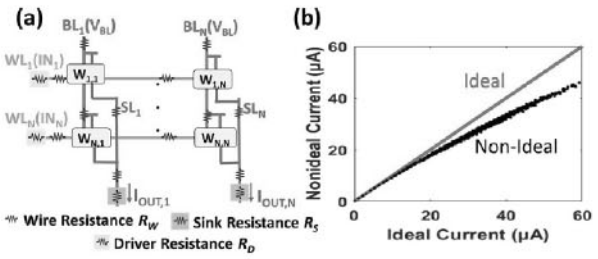


Fig. 2. (a) Typical structure of analog CiM crossbar with non-idealities (b) Output currents at the column of the crossbar obtained for 32 nm SRAM technology.

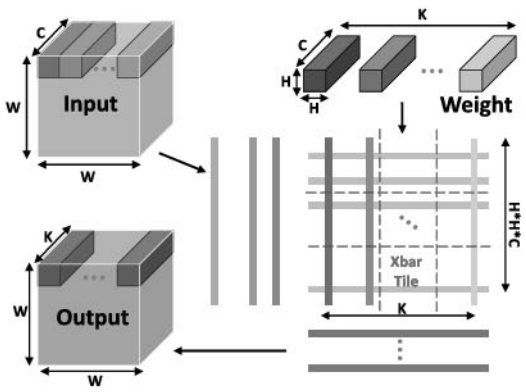


Fig. 3. Convolution layer mapping for crossbar based CiM accelerators.

multiplication is to utilize analog crossbar arrays. Weights are stored within the memory array as the conductance of memristive devices or the voltage of the SRAM cells. Input vectors are then applied as voltages across the rows of the crossbar following a digital-to-analog converter (DAC). The resulting current passing through each crossbar element depends on the stored weight value. These currents are accumulated at the bit lines or crossbar columns, and the value is sensed using ADCs.

However, the precision of memory devices and DACs is usually lower than the precision required by weights and activations in DNNs. To accommodate the higher precision requirements, techniques like bit-slicing and bit-streaming are employed [10], [22]. The high-precision MVM operation is decomposed into multiple bit-wise MVM operations that the crossbar can support. Weight values are divided into smaller bits and distributed across different memory subarrays or crossbars, whereas input activation bit-slices are streamed in at the crossbar rows over multiple cycles. The required high-precision output values are accumulated by appropriately shifting and adding the low-precision partial outputs. The ADC precision required to capture the entire dynamic range of analog signals using bit-slicing and bit-streaming is determined by the equation:

$$B_{ADC} = log(N) + s_w + s_a - 1, \tag{1}$$

Here, s_w and s_a represent the bit-slice precision of weights and activations, respectively, and N denotes the crossbar size. Although bit-slicing reduces the required precision for ADCs, further reducing ADC precision below B_{ADC} (Eq. 1) results in inaccuracies in capturing the entire dynamic range of analog signals, leading to quantization of partial sums.

B. Non-Idealities in Analog CiM

Analog non-idealities present in analog CiM hardware impact the fidelity of compute operation leading to degradation in workload accuracy [23]. Fig. 2(a) shows a 32nm SRAM-based crossbar array including parasitic resistances: wire resistance (RW), driver resistance (RD), and sink resistance (RS). The presence of these non-idealities causes *IR* drops along the current-carrying lines (BL and SL) leading to deviations in output currents from its ideal values as shown in Fig. 2(b). These deviations cause the ADC to sense incorrect analog current values, thereby injecting errors into the partial sums. These errors accumulate across each column of every crossbar, ultimately leading to a degradation in the DNN inference accuracy. The severity of degradation in inference accuracy is impacted by several factors including, crossbar size, device technology, bit-slice, and bit-stream precision.

C. Mapping Convolution Layer to CiM Hardware

Convolution (Conv) operation between layer inputs and weights is converted to MVM operation for execution on crossbar-based CiM accelerators as shown in Fig. 3. It involves flattening the 4-dimensional weight kernel along height, width, and channel dimensions to generate the weight matrix. Similarly, each activation window is flattened to form the activation vector. Since flattened tensors are much larger than a single crossbar, the activation vectors and weight matrix are tiled according to the crossbar size. Further, each tiled tensor is bit-sliced according to the precision supported by the hardware. Each crossbar generates a partial sum for bitwise MVM operation which is scaled and accumulated spatially and temporally to generate the layer output. While we show results for the layer mapping described here, our methodology is flexible and can support other layer mappings.

III. Related Works

Finding optimal bit precision for the various data structures in deep learning is an active area of research. For instance, authors in [24] use reinforcement learning to reduce the precision of weights, activations, and partial sums while maintaining high accuracy. However, they do not incorporate quantization during training and thus require high precision for partial sums, which in turn requires high precision ADCs. Authors in BNN-RRAM [25] and BinaryResnet [26] reduce the required ADC precision to just 1-bit, but they only show results for Binary Neural Networks (BNNs). Moreover, BNN-RRAM [25] also requires batch normalization in the analog domain, potentially increasing non-idealities. The authors in [19] extend the training method proposed in BinaryResnet [26] to maintain 1-bit partial sums while supporting high precision for weights and activations. However, the results in [19], [25], [26] are only limited to small-scale datasets (CIFAR-10).

Several works have demonstrated the efficacy of ADC-Less architectures for large-scale datasets such as ImageNet [16],

[17], [18]. Although these works offer valuable insights, several key limitations remain. BitSplitNet [16] suffers a significant accuracy drop (6%) with 1-bit partial sum quantization and its modified bit-wise MVM execution model demands specialized hardware for deployment. Quarry [17], while achieving impressive ImageNet results with low ADC precision, lacks bit-slicing, a crucial technique to adapt the methodology to different memory devices with variable DAC and storage precision. EPSQ [18] incorporates bit-slicing, but both quarry [17] and EPSQ [18] suffer accuracy degradation under very aggressive partial sum quantization. In contrast, our work achieves the highest ImageNet accuracy with the lowest ADC precision (1.5-bits) reported to date.

Authors in [27] use a weight/input sparsity-aware CiM macro to reduce the ADC precision to 2 or 4 bits. However, they use an over-parameterized ResNet-18 model to achieve comparable accuracy with our approach. The work in [28] introduces a hierarchical in-memory ADC to reduce area overhead in eDRAM-based CiM macros by reconfiguring eDRAM cells as unit capacitors for a SAR ADC. However, this approach is limited to eDRAM and not applicable to other CiM technologies like 8T-SRAM or ReRAM. In contrast, our technology-agnostic hardware-software co-design methodology is applicable across various analog CiM technologies. Other works [29], [30] have proposed low power time domain ADCs. However, some of these works accumulate outputs in analog domain [30] or only show results on small scale datasets like CIFAR-10 [29].

Another way to eliminate ADC overhead in analog CiM designs is to perform maximum computations in the analog domain [31], [32], [33], [34]. In addition to using analog crossbars to perform MVM operations, these designs perform other operations such as accumulating partial sums from different crossbars [34] or caching data [31] in the analog domain. These analog domain computations are more susceptible to errors due to non-idealities. Moreover, Neuro-CIM [33] and [32] are specifically targeted at Spiking Neural Networks (SNNs) and BNNs, respectively.

Compared to the prior works discussed above, our approach provides a hardware algorithm co-design approach to mitigate the ADC overhead in CiM designs. We develop a quantization framework that is scalable to any weight, activation, and ADC precision. Additionally, we develop an optimized hardware macro for processing scale factors induced by partial sum quantization.

Table I summarizes the above discussion by showing a qualitative comparison of our work with previous works in the field of ADC-Less analog CiM designs. Our work is the first to provide high accuracy with ADC-Less CiM design (binary and ternary partial sums) complete with an optimized hardware implementation.

TABLE I
RELATED WORKS ON PARTIAL SUM QUANTIZATION

	Multibit W/A	Bit-slicing	Large Datasets	High accuracy with low ADC precision	Hardware implementation
BNN RRAM [25]	✗	-	✗	✓	✗
Saxena et al. [19]	✓	✓	✗	✓	✗
Kim et al. [26]	✗	-	✗	✗	✗
Bit-SplitNet [16]	✓	✓	✓	✗	✓
Quarry [17]	✓	✗	✓	✗	✗
EPSQ [18]	✓	✓	✓	✗	✗
This work	✓	✓	✓	✓	✓

IV. ALGORITHM HARDWARE CO-DESIGN

In this section, we introduce our algorithm hardware co-design approach for developing ADC-Less CiM hardware. From the algorithm side, we develop a quantization framework tailored for ADC-Less CiM design with binary and ternary partial sums. DNNs trained with this quantization technique require partial sums to be multiplied by scale factors. To efficiently process the scaling factors introduced by partial sum quantization, we propose an optimized hybrid analog digital CiM macro.

A. Quantization-Aware Training

Quantization focuses on optimizing DNNs for efficient deployment on resource-constrained hardware. Weight and activation quantization are among the most widely employed techniques to increase DNN efficiency. Quantized DNNs require fewer bits to represent these data structures, leading to a reduction in the compute and memory access cost during execution. While weight and activation quantization is highly popular for conventional DNN accelerators, CiM hardware architectures have an additional data structure for quantization: partial sums. Quantizing partial sums to low precision directly translates to a reduction in ADC precision, enabling a considerable improvement in the efficiency of analog CiM hardware.

To accurately simulate partial sum quantization effects during training, it is necessary to emulate the computational characteristics of analog CiM hardware. This involves the following steps:

- **Quantizing weights and activations:** Reducing weights and activations to low-precision integers.
- **Bit-slicing:** Dividing weight and activation tensors into bit-slices compatible with the CiM hardware. SRAM-based CiM only works with 1-bit weight slices, while various post-CMOS technologies may support multiple bits per weight bit-slice [10].
- **Im2Col for convolution:** Transforming convolutions into matrix-vector multiplications using the Im2Col operation [10], [22] shown in Fig. 3. This step is only required for convolutional layers.
- **Tiling:** Dividing the resulting matrix multiplication into appropriately sized blocks to match the physical dimensions of the CiM crossbar.
- **Emulating ADC quantization:** Quantizing the partial sums generated at crossbar columns to emulate the ADC precision.
- **Accumulating partial sums:** Accumulating the quantized partial sum values across different crossbars and bit-slices.

This section outlines our quantization framework, beginning with weight and activation quantization and then focusing on partial sum quantization.

Algorithm 1 Bit-Slicing Algorithm

Input: Integer matrix $\mathrm{v_{int}}$, precision $\mathrm{B_v}$, bit-slice precision $\mathrm{s_v}$
Output: Bit-sliced matrix $\mathrm{v_b}$

$\mathrm{n_b} \leftarrow \lceil \mathrm{B_v}/\mathrm{s_v} \rceil$
for $i \in [0, n_b)$ **do**
  $\mathrm{v_{b_i}} \leftarrow floor(\mathrm{v_{int}}/(2^{\mathrm{s_v}})^i)$
end for
$\mathrm{v_b} \leftarrow remainder(\mathrm{v_b}, 2^{\mathrm{s_v}})$

1) Weight and Activation Quantization: For integer quantization of weights and activations, we implement Learned Step Size Quantization (LSQ) [35]. LSQ methodology performs quantization via scale factor learned using gradient descent during training. Given a full precision value v, scaling factor s and positive and negative quantization levels Q_P, Q_N, respectively, the scaled integer representation v_{int} and the quantized representation v_q are given by,

$$v_{int} = \left\lfloor clip\left(\frac{v}{s}, -Q_N, Q_P\right)\right\rceil, \quad v_q = v_{int} \times s \tag{2}$$

For signed weights : $Q_N : 2^{B_{w-1}}$, $Q_P : 2^{B_{w-1}} - 1$ while for unsigned activations : $Q_N : 0$, $Q_P : 2^{B_a} - 1$ where B_w and B_a are the weight and the activation precision, respectively.

2) Bit-Slicing: CiM accelerators perform bit-serial and bit-parallel execution where bit-slicing determines the computation granularity. The weight matrix is bit-sliced and stored in multiple crossbars while activations are bit-sliced and streamed serially on word lines. The bit-slicing function divides integer values into multiple low precision bit-slices (Algorithm 1). The bit-slicing function utilizes the floor function which is not amenable to gradient descent because it has zero gradients everywhere. Therefore, the floor function is approximated using straight through estimator. Consequently, the gradient output through the bit-slicing function ($\nabla_{v_{int}}$) is given by,

$$v_b = BitSlice(v_{int}) \;\; \nabla_{v_{int}} = \frac{1}{n_b} \times \sum_{i=0}^{n_b-1} \frac{\nabla_{v_{b_i}}}{(2^{s_v})^i}, \tag{3}$$

3) Partial Sum Quantization: We consider extremely low precision partial sums, binary (1-bit) and ternary (1.5-bit). partial sums quantized to a binary value require only a sense amplifier for analog-to-digital conversion. While partial sums quantized to a ternary value require two comparators offering notable efficiency gains compared to traditional high precision ADCs. Given the high precision partial sums ps and scaling factor α, the binary partial sums ps_b is given by,

$$ps_b = Q_b(ps) = \alpha \cdot sign\left(\frac{ps}{\alpha}\right) \tag{4}$$

The scaling factor is learned during training. The backward pass through the quantization function involves deriving gradients for full-precision partial sums and the scaling factor. This is obtained by introducing the following differentials,

$$\frac{\partial ps_b}{\partial ps} = \begin{cases} 1 & \text{if } -1 \le ps \le 1 \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

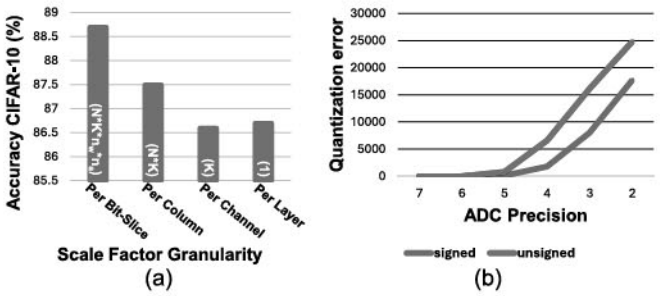


Fig. 4. (a) Impact of scale factor granularity on accuracy (finer scale factor granularity gives higher accuracy). (b) Quantization error induced by ADC precision reduction in crossbars storing signed and unsigned weights (signed weights incur lower quantization error).

$$\frac{\partial ps_b}{\partial \alpha} = sign(ps) \tag{6}$$

For ternary quantization of partial sums, we introduce a comparison threshold β. The ternary partial sum ps_t is given by,

$$ps_t = Q_t(ps) = \alpha \cdot clamp\left(\left\lfloor \frac{ps}{\beta} \right\rceil, -1, 1\right) \tag{7}$$

Similarly, the backward pass through the ternary quantization function is given by introducing the following differentials,

$$\frac{\partial ps_t}{\partial ps} = \begin{cases} \frac{\alpha}{\beta} & \text{if } -1 \le \frac{ps}{\beta} \le 1 \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

$$\frac{\partial ps_t}{\partial \alpha} = clamp\left(\left\lfloor \frac{ps}{\beta} \right\rceil, -1, 1\right) \tag{9}$$

$$\frac{\partial ps_t}{\partial \beta} = \begin{cases} -\alpha * \frac{ps}{\beta^2} & \text{if } -1 \le \frac{ps}{\beta} \le 1 \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

4) Quantization Granularity: The granularity of quantization of partial sums impacts the quantization error. Quantization at a finer granularity captures the distribution of partial sums with higher fidelity and leads to less quantization error. We experimented with different quantization granularities for partial sum quantization, as shown in Fig. 4(a). We experiment with per-layer, per-crossbar column, per output channel and per bit-slice granularity of scaling factors. For a convolution layer with K output channels mapped on N crossbars with n_a activation bit-slices and n_w weight bit slices, the number of scale factors in a layer at per bit-slice granularity is given by,

$$\#\text{ scale factors (bit-slice granularity)} = N * K * n_a * n_w \tag{11}$$

Per column granularity has $N * K$ scaling factors, Per channel granularity has K scaling factors while per Layer granularity has only 1 scaling factor per layer. As expected, the finest quantization granularity (per bit-slice) leads to the highest accuracy. However, per bit-slice quantization of partial sums introduces many scale factors that incur a non-negligible overhead. This includes parameter overhead as well as compute overhead of processing the scaling factors. Additionally, for ternary partial sum quantization, per bit-slice quantization also involves fine grain comparison thresholds which means that processing each bit-slice requires changing the threshold value used by the

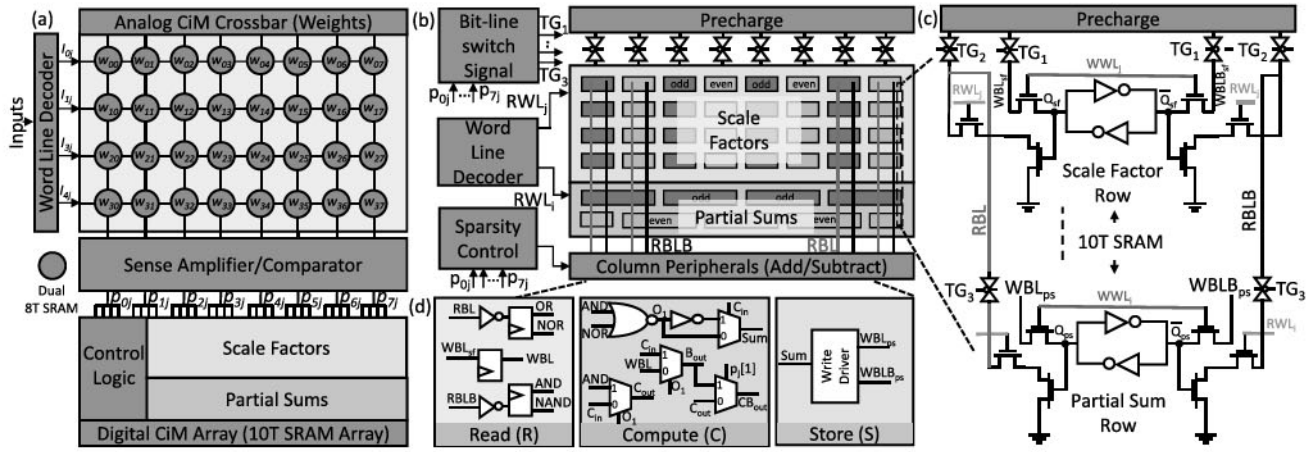


Fig. 5. (a) Hybrid Analog-Digital CiM macro. (b) Architecture of digital CiM array, incorporating column peripherals with a chain of 1-bit adder/subtractor at each column to enable full-adder/subtractor functionality. (c) Read Bit lines (RBL, RBLB) of scale factor and partial sum memories are connected to realize CiM operation. Write Bit line (WBL_{sf}) of scale factor memory helps to perform both read and write operations. (d) Detailed view of the column peripheral, illustrating the implementation of in-memory subtraction and addition operations. CB_{out} represents the final carry/borrow output from the column peripheral.

comparators. Keeping in mind the overheads associated with fine grained quantization, we introduce an efficient variant of partial sum quantization Q^{eff}. For efficient binary quantization (Q_b^{eff}), we quantize the per bit-slice scale factors while for efficient ternary quantization (Q_t^{eff}), we use per layer granularity of comparison threshold and quantize per bit-slice scaling factors.

5) Handling Signed Weights: Handling positive (+ve) and negative weights (−ve) in CiM hardware is a design choice with both hardware and software implications. PUMA [10] stores weights in separate crossbars dedicated to +ve and −ve values and correspondingly subtracts the +ve and −ve partial sums after ADC conversion. ISAAC [22] stores both unsigned +ve and −ve weights in the same crossbars with an additional dummy column subtracting the bias corresponding to negative weight values post adc conversion. XNOR-SRAM [13] utilizes a pull-up and pull-down bit cell design to store +ve and −ve weights in the same crossbar. From the perspective of partial sum quantization, storing +ve and −ve weights in separate crossbars entails that the partial sums are first quantized and then subtracted as shown below,

$$ps_q = Q(ps^+) - Q(ps^-) \tag{12}$$

On the other hand, storing positive and negative weights in the same crossbar ensures that partial sum subtraction happens before quantization as shown below,

$$ps_q = Q(ps^+ - ps^-) \tag{13}$$

Storing +ve and −ve weights in the same crossbar requires signed computation to be implemented within the crossbar. This allows positive and negative products to negate each other, reducing the range of partial sums. This enables better quantization and helps us achieve higher accuracy after training. To quantitatively evaluate the impact of these design choices of handling signed weights on partial sum quantization, we evaluated the error between quantized and full-precision partial sums at different ADC precisions. The quantization error is given by,

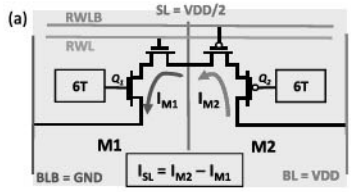
$$error = ||ps_q - ps||^2 \tag{14}$$

The errors at different ADC precision are shown in Fig 4(b). We observe that performing signed operations within crossbars leads to lower quantization error compared to storing +ve and −ve weights in separate crossbars. This error reduction is crucial when training with partial sum quantization, as it allows us to achieve high accuracy even with extremely low-precision partial sums.

B. Proposed Hardware

The proposed quantization framework introduces several scale factors for achieving high accuracy with binary and ternary partial sums. To efficiently incorporate partial sum quantization, we developed HCiM, an ADC-Less hybrid analog-digital CiM architecture. The principal design of the HCiM macro [21] is shown in Fig. 5(a). It consists of an analog CiM crossbar designed for performing the MVM operations between inputs and weights and a digital CiM (DCiM) array to perform the scale factor operations.

In the Analog CiM array, the weights of the DNN are stored in a current-based dual 8T-SRAM crossbar, while input activations are fed through the word line decoder. The bitcell (Fig. 6(a)) consists of two 8T-SRAM cells (M1 and M2). BL and BLB are biased at VDD and ground respectively while SL is biased at VDD/2. M1 conducts current from SL into BLB while M2 conducts current from BL into SL. Consequently, the overall SL current is the subtraction of M1's current from that of M2's. Inputs for the MVM operation are applied along RWL as voltages, following the input encoding in Fig. 6(b). Weights of the MVM operation are stored in cells following the weight encoding in Fig. 6(c). The output of the MVM operation is encoded in the SL current, following the output encoding in Fig. 6(d). It is important to note that the currents for outputs 1



(b)

Input	RWL
0	0
1	VDD

(c)

Weight	Q1	Q2
0	0	0
1	0	VDD
-1	VDD	0

(d)

Input	Weight	I_{M1}	I_{M2}	I_{SL}	Output
0	X	0	0	0	0
1	0	I_{HRS}	I_{HRS}	0	0
1	1	I_{HRS}	I_{LRS}	I_{LRS}-I_{HRS}	1
1	-1	I_{LRS}	I_{HRS}	-(I_{LRS}-I_{HRS})	-1

Fig. 6. (a) Dual 8T-SRAM based bit-cell (b) input and (c) weight mapping (d) truth table showing MVM operation between input and weight.

and -1 have the same magnitude but flow in opposite directions. This facilitates the cancellation of currents from different rows of the crossbar, resulting in a reduction in the magnitude of the SL current. Consequently, this leads to fewer *IR* drops and thus lesser deviation of SL currents from their ideal values, ultimately enhancing the DNN's inference accuracy as shown in [36] with a 2T2R-based cell. Note that M2 incorporates PMOS read-access transistors instead of NMOS to ensure the V_{GS} biasing of its access transistors matches that of M1. To enable PMOS read-access transistors to utilize the weight encoding in Fig. 6(c), they are connected to the internal node of the 6T SRAM, which stores the complement of the stored data. The input bits are streamed over multiple cycles. The MVM output corresponding to every input bit at the column of the crossbar is processed through a sense amplifier or comparator. We require one comparator in the case of binary PSQ and two comparators in the case of ternary PSQ [20]. The output of the comparator is a set of binary or ternary values represented as $p(-1,0,1)$. To accommodate negative values of p, we use 2-bit numbers: 00 for 0, 01 for 1, and 11 for -1.

The DCiM array stores the quantized scale factors associated with a PSQ-trained DNN. Outputs from the comparator are used in this array to perform the scale factor computation from Eq. 7. During training the shift operation (2^j) is integrated with the scale factor values. While the DCiM array draws inspiration from previous work [37], which was limited to vector-vector addition operations, our system requires additional functionalities for effective scale factor computation (s*p). Specifically, our system requires the capability to perform addition, subtraction, or no operation based on whether p is 1, -1, or 0. Hence, a significant contribution of our work lies in adapting the DCiM array to efficiently execute these specific operations. Further details regarding the DCiM array are provided below.

1) CiM Full Subtractor: To explain how a subtraction operation is carried out in the DCiM array, we first describe how an addition operation can be performed. Adding a scale factor row to a partial sum row requires two cycles due to the higher precision of partial sums compared to scale factors [37]. This involves processing odd columns in one cycle and even columns in the next. For better throughput, the addition operation is pipelined into 3 cycles as depicted in Fig. 7.

Initially, to add row j of the scale factor memory to row i of the partial sum memory, $RWL_{j,i}$ are activated. Then, binary/ternary values p control the bit-line switch signal block (Fig. 5(b), (c)), generating transmission gate ($TG_{2,3}$) signals. For example, when $p = 00$, $TG_{1,2,3}$ is turned OFF, while $p = 01$ activates $TG_{2,3}$.

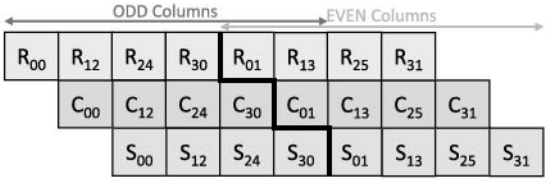


Fig. 7. Read Compute Store pipeline of DCiM array. R_{ji} represents the addition or subtraction operation between row j and i of scale factor and partial sum memory.

TABLE II
BITWISE OPERATION ON READ BIT LINES

Q_{sf}	Q_{ps}	RBL (NOR)	RBLB (AND)
0	0	VDD	GND
0	1	GND	GND
1	0	GND	GND
1	1	GND	VDD

To add row j of scale factor memory to row i of the partial sum memory, bit lines are precharged to V_{DD}, and $RWL_{j,i}$ are activated. Suppose scale factor and partial sums for each analog cim crossbar column are 4-bit and 8-bit, respectively. Assume that each scale factor and partial sum bit (Q_{sf}, Q_{ps}) is zero. In this case, these values will not discharge the RBL, and the signal on the RBL will remain at V_{DD}. However, $\overline{Q_{sf}}$ and $\overline{Q_{ps}}$ values are one, the RBLB will discharge, resulting in a final value of zero on the RBLB. Similarly, we can derive other values on RBL and RBLB for different combinations of Q_{sf} and Q_{ps}, enabling the realization of NOR and AND operations on the bit lines as shown in Table II.

These NOR and AND values are latched during the read cycle, facilitating parallel processing of multiple columns of the analog CiM crossbar. During the Compute cycle, these latched values are used by the chain of column peripherals (Fig 5(d)) to get the final Sum and Carry bits which is the result of adding scale factor value to partial sum value. Finally, the computed sum output is stored in the partial sum memory during the Store cycle.

Implementing subtraction alongside addition involves storing the 2's complement of scale factors and retrieving these values when subtraction is needed. However, this approach requires $2\times$ memory since the same scale factor value can be added to or subtracted from the partial sum value depending on p. The logical expression for the Difference (D) and Borrow (B_{out}) bits of a full subtractor is shown below. Here, A and B are two

input bits, and B_{in} is the input borrow bit.

$$D = A \oplus B \oplus B_{in}, \tag{15}$$

$$B_{out} = \overline{A}B + BB_{in} + B_{in}\overline{A} \tag{16}$$

While the Difference bit aligns with the Sum bit of a full adder, realizing the Borrow output via OR/NOR and $NAND/AND$ operations is not feasible. Previous methods, such as those outlined in [38], employed a two-cycle process for in-memory subtraction, reading one input in the first cycle and performing a bitwise operation between two inputs in the next. This approach, however, increases the latency of subtraction (4 cycles). To address this, our proposed architecture achieves subtraction in 3 cycles, similar to an addition operation. During the read cycle, when the write bit lines and word lines of the scale factor memory are inactive (as depicted in Fig. 5(c)), we read the scale factor value in parallel for columns requiring subtraction ($p = 11$) by activating TG_1. This value is then utilized in the Compute cycle to determine the B_{out} bit, as illustrated in Fig. 5(d). Finally, the binary/ternary value p acts as a select bit to MUX in Fig. 5(d) to decide if the output is a carry or borrow bit.

2) Sparsity Control: At least 50% of ternary values p are zero [21]. Consequently, a notable portion of the scale factor computation ($p * s$) can be skipped. During the Read cycle, when a specific p value is zero, the bit-line switch signal deactivates the corresponding column by keeping $TG_{1,2,3}$ turned OFF. This action prevents the discharge of bit-lines, thereby conserving dynamic energy. Furthermore, we integrate clock gating into the column peripherals, enabling us to bypass computations for columns where $p = 0$ during the Compute cycle. The sparsity control block generates the enable signal for this clock-gating logic. Moreover, in the Store cycle, no value is written to the columns where $p = 0$, resulting in additional energy savings during the write operation.

V. Evaluation Results

A. Experimental Setup

DNN Training Methodology: We adopt a two-stage training approach to obtain the final DNN model with quantized weights, activations, and partial sums. The first step involves, training the DNN model with only weight and activation quantization. In the second step, we introduce partial sum quantization in the training loop. We employ the following training protocol:

- **Datasets:** CIFAR-10, ImageNet.
- **Models:** Resnet-20 and Wide-Resnet20 (CIFAR-10), Resnet-18 (ImageNet).
- **Optimizer:** SGD with momentum.
- **Epochs per training stage:** 100 (CIFAR-10), 60 (ImageNet).
- **Learning Rate:** 0.01 initial, cosine decay scheduler.

Non-ideality Evaluation Methodology: To evaluate the impact of non-idealities on DNN's inference accuracy, we utilize Geniex [23]. Geniex efficiently models the non-ideal currents of crossbars taking data dependencies into account. The parameters of the 32 nm crossbar arrays used are shown in Table III. We use 32 nm Predictive Technology model for the bit-cells [39]. Our wire resistance is 6.5 $\Omega/\mu m$ and via resistance is 8 Ω. Our wire and via resistance and layout rules are consistent with an Intel technology [40], [41].

TABLE III
Key Parameters of 32 nm Crossbar Arrays

Gate Pitch	112.5 nm
Fin Pitch	112.5 nm
Wire Resistance	6.5 Ω/μm
Via Resistance	8 Ω
Driver Resistance	500 Ω
Sink Resistance	100 Ω
VDD	0.8 V
Bits per Input Signal	1b
Bits per Device	1b

TABLE IV
HCiM Configurations for 4-Bit Weight and Activations. #Scale Factors and #Partial Sums Are Reported per Crossbar (Eq. 11)

Configuration	Analog CiM Crossbar Size	#Scale Factors	#Partial Sums	DCiM Array Size
A	128x128	4*128	1*128	24x128
B	64x64	4*64	1*64	24x64

Performance Evaluation: The DCiM array depicted in Fig. 5(b), is designed using 65nm technology. The energy, latency, and area results of the DCiM array are obtained from schematic-level simulations. The 10T-SRAM array is designed using cadence virtuoso, and the control logic is implemented using synopsis design compiler. The supply voltage is set to 1V, and the operating frequency is 500 MHz. We compare the macro-level results of our DCiM array with various types of ADCs [42], [43], [44] for processing all the columns of the analog CiM crossbar. To ensure a fair comparison, we selected ADCs designed in 65nm technology based on the ADC survey [45]. For system-level comparisons, we utilize the cycle-accurate simulator from PUMA [10], replacing the ADCs with our DCiM array. Since other components in the system are designed using 32nm technology, we scale the metrics of ADCs and our DCiM array to 32nm using predictive technology models [46]. Consistent with prior CiM approaches, we assume that weights and scale factors, once trained, are pre-loaded into the memory arrays and can be reused across different inputs. We present accuracy and performance results for two configurations of HCiM as shown in Table IV. The memory sizes for scale factor and partial sum in configuration A are specified as 4*128*4 and 1*128*8 bits, respectively, requiring a 24x128 DCiM array per analog crossbar. For benchmarking HCiM, we present results on the CIFAR-10 dataset using various models such as ResNet-20,32,44, Wide ResNet-20, and VGG-9,11.

B. Accuracy Evaluation Results

1) CIFAR-10 Results: The accuracy of neural networks trained with PSQ training is presented in Table V. We observe a

TABLE V
ACCURACY WITH VARYING ADC PRECISION AND CROSSBAR SIZES

Model (Crossbar Size)	ADC Precision (bits)						
	7	6	4	1.5 (Q_t)	1.5 (Q_t^{eff})	1 (Q_b)	1 (Q_b^{eff})
ResNet-20 (128)	92.26	91.27	90.20	89.0	88.80	86.8	86.30
ResNet-20 (64)	-	91.93	91.00	89.8	89.80	88.3	88.20
Wide-ResNet-20 (128)	93.80	93.70	92.90	92.1	92.03	91.90	91.90
Wide-ResNet-20 (64)	-	93.91	93.10	92.4	92.24	91.93	91.89

minimal drop in accuracy with the reduction of ADC precision, attributable to quantization-aware training. Interestingly, when partial sums are quantized to ternary values, the accuracy is comparable (within 1.5%) to that achieved with 4-bit ADC precision. This can be credited to the fine granularity of scale factors, which effectively enhances the representation ability of the quantized partial sums. The accuracy drops ∼2% as we go from 1.5 to 1-bit ADC in the ResNet-20 model which is due to the extreme quantization of partial sums. However, the drop in accuracy when moving from 1.5-bit to 1-bit ADC is less than 0.5% for Wide Resnet-20 model. Additionally, we present results using a 64x64 analog CiM crossbar, which ideally requires only 6-bit ADCs. It is observed that the accuracy drop with 1,1.5-bit ADC is less pronounced compared to a 128x128 crossbar. This is because, in a 64x64 crossbar, the partial sums undergo less severe quantization from only 6 to 1.5-bits.

2) ImageNet Results: Table VI contains the results obtained on Imagenet and a comparison with related works. Following Quarry [17], the partial sums of the first and last layer are not quantized. With a ternary partial sum quantization, we get no accuracy loss compared to the baseline. Getting higher accuracy than the baseline can be attributed to more training effort introduced by the two-stage training methodology of partial sum quantization. With the efficient version of ternary quantization, there is 4% accuracy drop.

Quarry [17] does not consider bit-slicing in their training methodology and suffers from high accuracy degradation with heavy partial sum quantization (1-bit). By incorporating bit-slicing in our training, we achieve no accuracy loss with ternary partial sums. Bit-split-Net [16] considers bit-slicing of activations but still suffers from high accuracy loss. Bit-Split-Net modifies the bit-wise execution within CiM hardware by considering each bit-path separately for parallel execution. Since accumulation across bit-slices does not occur until the final layer of DNN model, there is a considerable information loss which manifests as a large accuracy degradation. Our approach is based on conventional CiM hardware which performs accumulation across bit-slices after every MVM operation [10], [22]. EPSQ [18] considers bit-slicing of both weights and activations yet achieves substantial accuracy degradation with 3-bit ADCs (2%). This can be attributed to the shortcomings of their training methodology, particularly, the partial sum quantization function. EPSQ uses heuristic based optimization for evaluating scaling factors for their partial sum quantization, while in our methodology, the scaling factors are optimized using gradient descent during training.

TABLE VI
ACCURACY RESULTS ON IMAGENET. (CROSSBAR SIZE 128)

ImageNet						
	W/A	s_w/s_a	ADC Precision	Baseline	Accuracy	Gap
Quantized Resnet-18						
Bit-Split-Net	3/3	3/1	1	67.6	61.2	6.4
Quarry	3/3	3/3	1	67.73	62.93	4.8
Quarry	3/3	3/3	4	67.73	67.93	-0.2
EPSQ	4/4	1/1	3	69.71	67.45	2.26
EPSQ	4/4	1/1	4	69.71	69.06	0.65
ADC-Less (Q_t)	**3/3**	**1/1**	**1.5**	**69.4**	**70.1**	**-0.7**
ADC-Less (Q_t^{eff})	**3/3**	**1/1**	**1.5**	**69.4**	**65.4**	**4.0**

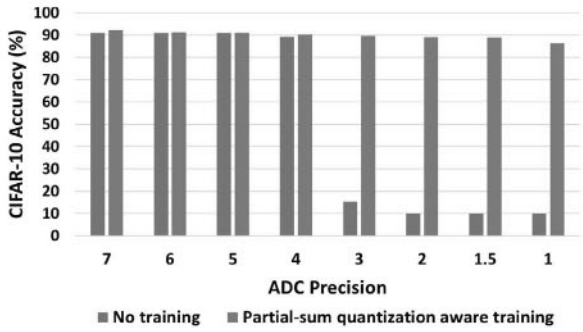


Fig. 8. Impact of partial sum quantization-aware training over static ADC precision reduction.

3) Impact of Partial Sum Quantization-Aware Training: Several approaches attempt to reduce ADC precision post-hoc [9], [22]. Such approaches fundamentally rely on inherent bit level sparsity abundant in modern DNNs. We observe that reducing ADC precision by 3-bits (from 7-bit to 4-bit) has a minimal impact on CIFAR-10 accuracy. This implies that there is enough bit-level sparsity to restrict the dynamic range of partial sums such that even 4-bit ADC is enough to accurately capture the partial sums. However, if ADC precision is reduced further, there is a catastrophic decline in accuracy as shown in Fig. 8. With the partial sum quantization-aware approach described in this paper, we are able to push the frontier of accuracy to much lower ADC precision.

4) Impact of Non-Idealities on Accuracy: Computation errors induced by analog non-idealities is a challenge in analog CiM accelerators. From the perspective of ADC-Less CiM hardware, the impact of non-ideality on compute accuracy is influenced by two factors,

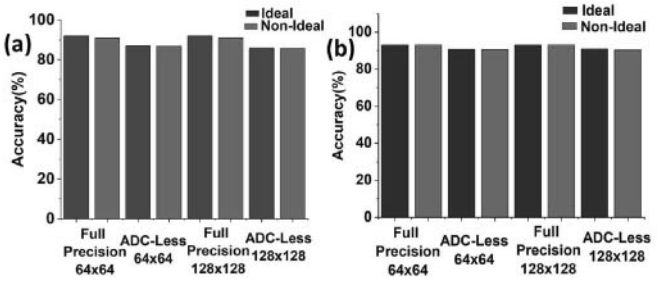


Fig. 9. Impact of Non-Idealities on inference accuracy of (a) ResNet-20 and (b) Wide-ResNet-20. ADC-Less hardware has partial sums with ternary quantization.

1) High sense margin offered by ADC-Less CiM makes it resilient to small perturbations in analog signals.
2) Switching in output states caused by non-idealities causes a larger error in output value in ADC-Less CiM compared to a baseline with high precision ADC. For example, in ADC-Less CiM with ternary PSQ, the output state may switch from +1 to 0 in the presence of non-idealities while the error will be lower in high-precision ADCs.

Based on the two above factors, it is non-trivial to state how ADC-Less CiM hardware performs in the presence of analog non-idealities. To answer this question, we analyze the effect of non-idealities on both full-precision and ADC-Less networks for two crossbar array sizes (64x64 and 128x128) for Resnet-20 and Wide-Resnet-20 on the CIFAR-10 dataset (Fig. 9). Our analysis reveals that the impact of non-idealities on both types of networks is similar. Both ADC-Less and high precision ADC baseline observer similar degradation in workload accuracy in the presence of non-idealities. Additionally, for both baseline and ADC-Less CiM the drop in accuracy is fairly low. This robustness is attributed to two key factors. Firstly, the utilization of the dual 8T-SRAM cell, which facilitates a reduction in the current magnitude along SL and consequently mitigates *IR* drops. Secondly, the wire resistance and via resistance encountered at the 32 nm technology node are relatively modest and thereby do not introduce a significant *IR* drop.

C. Performance Results

Energy vs Sparsity: Skipping the scale factor computation for columns of the analog CiM crossbar where $p = 0$ (for ternary partial sum quantization) leads to a substantial reduction in energy consumption, as depicted in Fig. 12. Specifically, going from 0% sparsity (binary quantization) to 50% sparsity results in a 24% reduction in energy. This reduction in energy results from various factors: there is no precharge of bit-lines for these columns, adder/subtractor circuits are clock-gated, and no store operation is performed in the store cycle. However, it is important to note that while sparsity helps to reduce energy, it does not affect latency. This is because multiple columns are processed in parallel in each cycle; thus, even with 50% sparsity, there may still be columns with non-zero p values.

DCiM array vs ADCs: Table VII presents a comparison of the DCiM array in HCiM with various types of ADCs.

TABLE VII
DCiM ARRAY COMPARISON WITH ADC TO PROCESS ONE COLUMN OF ANALOG CiM CROSSBAR. A&B ARE CONFIGS FROM TABLE IV

Analog CiM Column Peripheral	ADC Precision	Latency (ns)	Energy (pJ)	Area (mm^2)
Area Optimized SAR [42]	7	1.52	4.1	0.004
Energy Efficient SAR [43]	6	0.15	0.59	0.027
Latency Efficient Flash [44]	4	0.05	1.86	0.003
DCiM Array (A)	-	0.06	0.22	0.009
DCiM Array (B)	-	0.1	0.22	0.005

Since the DCiM array can handle multiple columns in parallel, the latency is averaged over columns. Compared to 6 or 7-bit ADCs, DCiM (A, B) exhibits significantly lower latency due to its ability to process multiple columns of the analog CiM crossbar simultaneously. Regarding energy efficiency, DCiM (A, B) consumes 12× less energy than the 4-bit ADC. However, DCiM (A) demonstrates a 3.6× higher area-normalized latency compared to 4-bit ADC. Furthermore, comparing the latency of DCiM (A) with DCiM (B), it's evident that configuration A has twice lower latency as it can process twice the number of columns in parallel. For all system-level simulations in the next section, only one ADC and DCiM array are considered per analog CiM crossbar.

System Level Comparison: HCiM is compared to analog CiM accelerators employing low precision ADCs. HCiM (binary, ternary) shows lower energy consumption compared to the baseline analog CiM accelerator using low precision ADCs as shown in Fig. 10(a). On average across all models, HCiM exhibits at least 3× lower energy compared to all the baselines. Additionally, HCiM (ternary) achieves at least 15% lower energy compared to HCiM (binary), attributed to the sparsity support in HCiM. HCiM also demonstrates significantly lower latency, ranging from 3 to 12× lower compared to baselines using SAR ADCs as shown in Fig. 10(b). This improvement can be attributed to the low energy and latency of DCiM compared to SAR ADCs. However, compared to the analog CiM baseline using 4-bit Flash ADC, HCiM shows 11% higher latency due to the very low latency and area of the Flash ADC.

We also considered HCiM configuration B with a 64x64 crossbar size. In this configuration, as we reduce the crossbar size in PUMA, we increase the number of physical crossbars in the PUMA core to maintain the same number of multiplication units as in the 128x128 crossbar size. However, increasing the number of analog crossbars may lead to increased data movement of partial sums across the crossbars compared to configuration A. Therefore, the advantage of reducing the ADC overhead might decrease. Nevertheless, even in this scenario, HCiM demonstrates at least 2.5× lower energy consumption compared to baselines that use 6-bit and 4-bit ADCs as shown in Fig. 11(a). However, HCiM exhibits 1.4× higher latency compared to the 4-bit ADC (Fig. 11(b)), which can be attributed to the low area and latency overhead of the Flash ADC.

HCiM vs Related works: The accuracy vs energy-delay-area-product (EDAP) for HCiM compared to related works is shown in Fig. 12(b). Quarry [17] and BitSplitNet [16] did not report the performance results, we get their performance results using

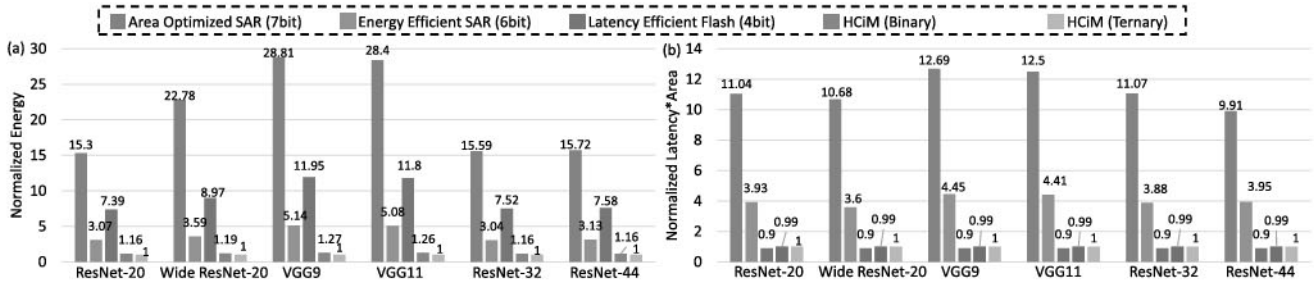


Fig. 10. (a) Energy and (b) Latency*Area for different workloads using HCiM Configuration A from Table IV compared to baseline using low precision ADCs. Both Energy and Latency*Area are normalized with respect to HCiM (Ternary).

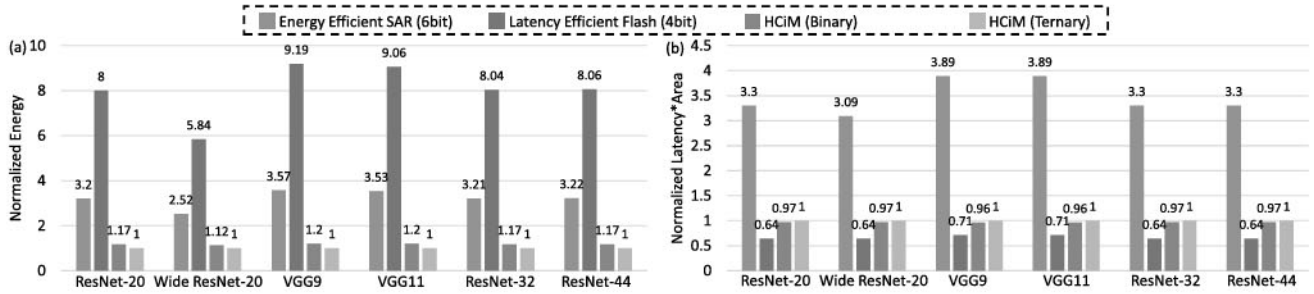


Fig. 11. (a) Energy and (b) Latency*Area for different workloads using HCiM Configuration B from Table IV compared to baseline using low precision ADCs. Both Energy and Latency*Area are normalized with respect to HCiM (Ternary).

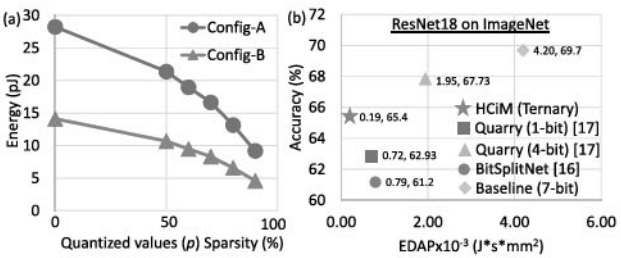


Fig. 12. (a) Energy to process all the columns of analog CiM crossbar with ternary quantization. (b) Accuracy vs EDAP comparison of HCiM with baselines on ImageNet dataset.

cycle-accurate simulator in [10]. Quarry uses analog and digital multipliers to implement scale factors. The energy and area for 1-bit ADC in [17] is estimated as 1/16 of 4-bit flash ADC [44]. The energy for the digital multiplier is obtained from [10]. Therefore, compared to Quarry with 1-bit ADC, HCiM achieves 2.5% higher accuracy with 3.8× lower EDAP. Moreover, compared to Quarry (4-bit), HCiM has 2.3% lower accuracy with 10.4× lower EDAP. Compared to BitSplitNet, HCiM has 4.2% higher accuracy with 4.2× lower EDAP. BitSplitNet uses independent paths to process each input and weight bit. Hence, energy and area for ResNet-18 with 4-bit inputs and weights are obtained by scaling 1-bit energy and area by 4.

VI. Conclusion

In this work, we propose an algorithm-hardware co-design approach to develop ADC-Less CiM accelerators. We introduce a quantization framework to train DNNs with extremely low precision partial sums. DNNs with binary partial sums require just a sense amplifier for analog to digital conversion while DNNs with ternary partial sums require 2 comparators for the conversion. We propose an efficient version of quantization functions with quantized scale factors and shared comparator thresholds. Our training methodology achieves accuracy within 1.5% of baseline with ternary partial sums. We provide analysis into the impact of weight mapping and the impact of non-idealities on the inference performance of trained DNNs. Additionally, to efficiently process the scale factors introduced by partial sum quantization, we develop an optimized hardware macro. The proposed hardware HCiM integrates an analog CiM crossbar to perform MVM operation between input and weights, alongside a digital CiM array optimized for scale factor processing. The analog CiM crossbar is based on current-based dual 8T-SRAM technology. We also introduce an innovative method for performing in-memory subtraction and addition within the digital CiM array. Our system-level evaluation using a cycle-accurate simulator shows up to 28× and 12× reduction in energy compared to the baseline that uses 7-bit and 4-bit ADCs respectively.

References

[1] N. Maslej et al., "Artificial intelligence index report 2023," 2023, *arXiv:2310.03715*.

[2] P. Villalobos, J. Sevilla, T. Besiroglu, L. Heim, A. Ho, and M. Hobbhahn, "Machine learning model sizes and the parameter gap," 2022. [Online]. Available: https://arxiv.org/abs/2207.02852

[3] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *ACM SIGARCH Comput. Archit. News*, vol. 23, no. 1, pp. 20–24, 1995.

[4] S. A. McKee, "Reflections on the memory wall," in *Proc. 1st Conf. Comput. Frontiers*, 2004, p. 162.

[5] A. Ankit et al., "Circuits and architectures for in-memory computing-based machine learning accelerators," *IEEE Micro*, vol. 40, no. 6, pp. 8–22, Nov./Dec. 2020.
[6] S. Yu, H. Jiang, S. Huang, X. Peng, and A. Lu, "Compute-in-memory chips for deep learning: Recent trends and prospects," *IEEE Circuits Syst. Mag.*, vol. 21, no. 3, pp. 31–56, 3rd Quart. 2021.
[7] K. Roy, I. Chakraborty, M. Ali, A. Ankit, and A. Agrawal, "In-memory computing in emerging memory technologies for machine learning: An overview," in *Proc. 57th ACM/IEEE Des. Automat. Conf. (DAC),* Piscataway, NJ, USA: IEEE Press, 2020, pp. 1–6.
[8] W. He et al., "2-bit-per-cell RRAM-based in-memory computing for area-/energy-efficient deep learning," *IEEE Solid-State Circuits Lett.*, vol. 3, pp. 194–197, 2020.
[9] M. Ali, I. Chakraborty, U. Saxena, A. Agrawal, A. Ankit, and K. Roy, "A 35.5-127.2 TOPS/W dynamic sparsity-aware reconfigurable-precision compute-in-memory sram macro for machine learning," *IEEE Solid-State Circuits Lett.*, vol. 4, pp. 129–132, 2021.
[10] A. Ankit et al., "Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference," in *Proc. ASPLOS*, 2019, pp. 715–731.
[11] M. Ali et al., "A 65 nm 1.4-6.7 TOPS/W adaptive-SNR sparsity-aware CIM core with load balancing support for DL workloads," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC),* Piscataway, NJ, USA: IEEE Press, 2023, pp. 1–2.
[12] J.-W. Su et al., "16.3 a 28nm 384kb 6T-SRAM computation-in-memory macro with 8b precision for AI edge chips," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, vol. 64, Piscataway, NJ, USA: IEEE Press, 2021, pp. 250–252.
[13] S. Yin, Z. Jiang, J.-S. Seo, and M. Seok, "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks," *IEEE J. Solid-State Circuits*, vol. 55, no. 6, pp. 1733–1743, Jun. 2020.
[14] R. Liu et al., "Parallelizing SRAM arrays with customized bit-cell for binary neural networks," in *Proc. 55th Annu. Des. Automat. Conf.*, 2018, pp. 1–6.
[15] H. Jiang, S. Huang, X. Peng, and S. Yu, "MINT: Mixed-precision RRAM-based in-memory training architecture," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS),* Piscataway, NJ, USA: IEEE Press, 2020, pp. 1–5.
[16] H. Kim et al., "Algorithm/hardware co-design for in-memory neural network computing with minimal peripheral circuit overhead," in *Proc. 57th DAC,* Piscataway, NJ, USA: IEEE Press, 2020, pp. 1–6.
[17] A. Azamat et al., "Quarry: Quantization-based ADC reduction for reram-based deep neural network accelerators," in *Proc. IEEE/ACM ICCAD.* Piscataway, NJ, USA: IEEE Press, 2021, pp. 1–7.
[18] Y. Kim et al., "Extreme partial-sum quantization for analog computing-in-memory neural network accelerators," *ACM J. Emerging Technol. Comput. Syst.*, vol. 18, no. 4, pp. 1–19, 2022.
[19] U. Saxena et al., "Towards ADC-less compute-in-memory accelerators for energy efficient deep learning," in *Proc. Des., Automat. & Test Europe Conf. & Exhib. (DATE),* Piscataway, NJ, USA: IEEE Press, 2022, pp. 624–627.
[20] U. Saxena and K. Roy, "Partial-sum quantization for near ADC-less compute-in-memory accelerators," in *Proc. IEEE/ACM ISLPED,* Piscataway, NJ, USA: IEEE Press, 2023, pp. 1–6.
[21] S. Negi, U. Saxena, D. Sharma, and K. Roy, "HCiM: ADC-less hybrid analog-digital compute in memory accelerator for deep learning workloads," 2024, *arXiv:2403.13577*.
[22] A. Shafiee et al., "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 14–26, 2016.
[23] I. Chakraborty, M. F. Ali, D. E. Kim, A. Ankit, and K. Roy, "GENIEx: A generalized approach to emulating non-ideality in memristive Xbars using neural networks," in *Proc. 57th ACM/IEEE Des. Automat. Conf. (DAC),* Piscataway, NJ, USA: IEEE Press, 2020, pp. 1–6.
[24] S. Huang et al., "Mixed precision quantization for ReRAM-based DNN inference accelerators," in *Proc. 26th Asia South Pacific Des. Automat. Conf.*, 2021, pp. 372–377.
[25] Y. Kim, H. Kim, and J.-J. Kim, "Neural network-hardware co-design for scalable RRAM-based BNN accelerators," 2018, *arXiv:1811.02187*.
[26] Y. Kim, H. Kim, J. Park, H. Oh, and J.-J. Kim, "Mapping binary resnets on computing-in-memory hardware with low-bit adcs," in *Proc. Des., Automat. & Test Europe Conf. & Exhib. (DATE),* Piscataway, NJ, USA: IEEE Press, 2021, pp. 856–861.
[27] J. Yue et al., "15.2 A 2.75-to-75.9 TOPS/W computing-in-memory nn processor supporting set-associate block-wise zero skipping and ping-pong cim with simultaneous computation and weight updating," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, vol. 64, Piscataway, NJ, USA: IEEE Press, 2021, pp. 238–240.
[28] S. Kim et al., "DynaPlasia: An eDRAM in-memory computing-based reconfigurable spatial accelerator with triple-mode cell," *IEEE J. Solid-State Circuits*, vol. 59, no. 1, pp. 102–115, Jan. 2024.
[29] P.-C. Wu et al., "An 8b-precision 6T SRAM computing-in-memory macro using time-domain incremental accumulation for AI edge chips," *IEEE J. Solid-State Circuits*, vol. 59, no. 7, pp. 2297–2309, Jul. 2024.
[30] D. Challagundla, I. Bezzam, and R. Islam, "A resonant time-domain compute-in-memory (rTD-CiM) ADC-less architecture for MAC operations," in *Proc. Great Lakes Symp. VLSI*, 2024, pp. 268–271.
[31] P. Chen et al., "RIMAC: An array-level ADC/DAC-free reram-based in-memory DNN processor with analog cache and computation," in *Proc. ASPDAC*, 2023, pp. 1–6.
[32] Y. Li et al., "An ADC-less RRAM-based computing-in-memory macro with binary CNN for efficient edge AI," *IEEE TCAS II: Exp. Briefs*, 2023, vol. 70, no. 6, pp. 1871–1875, Jun. 2023.
[33] S. Kim et al., "Neuro-CIM: ADC-less neuromorphic computing-in-memory processor with operation gating/stopping and digital–analog networks," *IEEE J. Solid-State Circuits*, vol. 58, no. 10, pp. 2931–2945, Oct. 2023.
[34] K. Yoshioka, "34.5 a 818-4094TOPS/W capacitor-reconfigured CIM macro for unified acceleration of CNNs and transformers," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, vol. 67, Piscataway, NJ, USA: IEEE Press, 2024, pp. 574–576.
[35] S. K. Esser et al., "Learned step size quantization," 2019, *arXiv:1902.08153*.
[36] T. Andrulis, J. S. Emer, and V. Sze, "RAELLA: Reforming the arithmetic for efficient, low-resolution, and low-loss analog PIM: No retraining required!" in *Proc. 50th Annu. Int. Symp. Comput. Archit.*, 2023, pp. 1–16.
[37] A. Agrawal et al., "IMPULSE: A 65-nm digital compute-in-memory macro with fused weights and membrane potential for spike-based sequential learning tasks," *IEEE Solid-State Circuits Lett.*, vol. 4, pp. 137–140, 2021.
[38] N. S. Dhakad, E. Chittora, V. Sharma, and S. K. Vishvakarma, "R-InMAC: 10T SRAM based reconfigurable and efficient in-memory advance computation for edge devices," *Analog Integr. Circuits Signal Process.*, vol. 116, no. 3, pp. 161–184, 2023.
[39] Y. Cao, "What is predictive technology model (PTM)?" *ACM SIGDA Newslett.*, vol. 39, no. 3, pp. 1–1, 2009.
[40] "32 nm lithography process," Accessed: May 12, 2024. [Online]. Available: https://en.wikichip.org/wiki/32_nm_lithography_process
[41] R. Brain et al., "Low-k interconnect stack with a novel self-aligned via patterning process for 32nm high volume manufacturing," in *Proc. IEEE Int. Interconnect Technol. Conf.,* Piscataway, NJ, USA: IEEE Press, 2009, pp. 249–251.
[42] C.-H. Chan et al., "A 3.8mW 8b 1GS/s 2b/cycle interleaving SAR ADC with compact 8mW structure," in *Proc. Symp. VLSIC,* Piscataway, NJ, USA: IEEE Press, 2012, pp. 86–87.
[43] C.-H. Chan et al., "26.5 a 5.5mW 6b 5GS/S 4×-Interleaved 3b/cycle SAR ADC in 65nm CMOS," in *Proc. IEEE ISSCC Dig. Tech. Papers,* Piscataway, NJ, USA: IEEE Press, 2015, pp. 1–3.
[44] H. Chung et al., "A 7.5-GS/s 3.8-ENOB 52-mw flash ADC with clock duty cycle control in 65nm CMOS," in *Proc. Symp. VLSIC,* Piscataway, NJ, USA: IEEE Press, 2009, pp. 268–269.
[45] B. Murmann, "ADC Performance Survey 1997–2023," Accessed: Nov. 20, 2023. [Online]. Available: https://github.com/bmurmann/ADC-survey
[46] A. Stillmaker and B. Baas, "Scaling equations for the accurate prediction of CMOS device performance from 180 nm to 7 nm," *Integration*, vol. 58, no. 0167–9260, pp. 74–81, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167926017300755

Shubham Negi (Student Member, IEEE) received his dual B.Tech. and M.Tech. degrees from the Indian Institute of Technology Kharagpur, India, in 2019. He is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. His primary research interests involve hardware-software co-design for machine learning workloads. He is the recipient of the S.N Bose Scholarship in 2018 and the Ross Fellowship in 2019.

Utkarsh Saxena received the bachelor's degree in technology from the Indian Institute of Technology, Delhi, India. He is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. His primary research interests involve hardware and algorithms for efficient deep learning.

Deepika Sharma (Graduate Student Member, IEEE) received the bachelor's degree in technology from the Indian Institute of Technology, Roorkee, India. She is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. Her primary research interests involve circuits and architectures for edge AI applications.

Jeffry Victor received the bachelor's degree in technology from Birla Institute of Technology and Science Pilani, India. He is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. His primary research interests involves cross-layer design of compute-enabled memories for emerging neural workloads.

Imtiaz Ahmed received the bachelor's degree in technology from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh. He is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. His primary research interests involves design of in-memory computing circuits using emerging devices.

Sumeet Kumar Gupta (Senior Member, IEEE) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology Delhi, New Delhi, India, in 2006, and the M.S. and Ph.D. degrees in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2008 and 2012, respectively. Currently, he is an Elmore Associate Professor of Electrical and Computer Engineering with Purdue University. He was an Assistant Professor of Electric Engineering with The Pennsylvania State University, State College, PA, USA, from 2014 to 2017, and a Senior Engineer with the Qualcomm Inc., San Diego, CA, USA, from 2012 to 2014. He has also worked as an Intern with the National Semiconductor, Santa Clara, CA, USA, in 2005; Advanced Micro Devices Inc., Santa Clara, in 2007; and Intel Corporation, Hillsboro, OR, USA, in 2010. He has published over 100 articles in refereed journals and conferences. His research interests include low-power variation-aware VLSI circuit design, neuromorphic computing, inmemory computing, nanoelectronics and spintronics, device-circuit co-design, and nanoscale device modeling and simulations. He was the recipient of the DARPA Young Faculty Award in 2016, the Early Career Professorships by Purdue and Penn State in 2021 and 2014, respectively, and the 6th TSMC Outstanding Student Research Bronze Award in 2012. He has also received the Magoon Award and the Outstanding Teaching Assistant Award from Purdue University in 2007 and Intel Ph.D. Fellowship in 2009. He is a Senior Member of EDS.

Kaushik Roy (Fellow, IEEE) received the B.Tech. degree from the Indian Institute of Technology, Kharagpur, and the Ph.D. degree from the University of Illinois at Urbana-Champaign, in 1990. He is the Edward G. Tiedemann, Jr., Distinguished Professor of Electrical and Computer Engineering at Purdue University. He joined the Semiconductor Process and Design Center of Texas Instruments, Dallas, where he worked for three years on FPGA architecture development and low-power circuit design. His research focuses on cognitive algorithms, circuits and architecture for energy-efficient neuromorphic computing/machine learning, and neuro-mimetic devices. He has supervised more than 100 Ph.D. dissertations and his students are well placed in universities and industry. He is the Co-Author of two books on Low Power CMOS VLSI Design (John Wiley & McGraw Hill). Throughout his career, he received several awards including the National Science Foundation Career Development Award in 1995, IBM Faculty Partnership Award, ATT/Lucent Foundation Award, 2005 SRC Technical Excellence Award, SRC Inventors Award, Purdue College of Engineering Research Excellence Award, Outstanding Mentor Award in 2021, Humboldt Research Award in 2010, 2010 IEEE Circuits and Systems Society Technical Achievement Award (Charles Desoer Award), IEEE TCVLSI Distinguished Research Award in 2021, Distinguished Alumnus Award from Indian Institute of Technology (IIT), Kharagpur, Fulbright-Nehru Distinguished Chair, DoD Vannevar Bush Faculty Fellow (2014–2019), SRC Aristotle Award in 2015, Purdue Arden L. Bement Jr. Award in 2020, SRC Innovation Award in 2022, and an Honorary Doctorate from Aarhus University in 2023.