DREAM-CIM: A Digital SRAM-Based CIM Accelerator for Energy- and Area-Efficient Edge AI

El Arrassi, A.E.; Huijbregts, L.C.A.; Gomony, Manil Dev; Gebregiorgis, Anteneh; Catthoor, Francky; Taouil, M.; Joshi, Rajiv V.; Hamdioui, S.

Publication date
2025

Document Version
Final published version

Published in
IEEE Transactions on Circuits and Systems for Artificial Intelligence

Citation (APA)
El Arrassi, A. E., Huijbregts, L. C. A., Gomony, M. D., Gebregiorgis, A., Catthoor, F., Taouil, M., Joshi, R. V., & Hamdioui, S. (2025). DREAM-CIM: A Digital SRAM-Based CIM Accelerator for Energy- and Area-Efficient Edge AI. *IEEE Transactions on Circuits and Systems for Artificial Intelligence*, *2*(3), 211 - 221.

DREAM-CIM: A Digital SRAM-Based CIM Accelerator for Energy- and Area-Efficient Edge AI

Asmae El Arrassi, Lucas Huijbregts, Manil Dev Gomony, Anteneh Gebregiorgis, Francky Catthoor, Mottaqiallah Taouil, *Member, IEEE*, Rajiv Joshi, *Life Fellow, IEEE*, and Said Hamdioui, *Senior Member, IEEE*

***Abstract*—With the rise of energy-constrained smart edge applications, there is a pressing need for energy-efficient computing engines that process generated data locally, at least for small and medium-sized applications. To address this issue, this paper proposes DREAM-CIM, a digital SRAM-based computation-in-memory (CIM) accelerator. It targets an energy- and area-efficient implementation of the multiply-and-accumulate (MAC) operation, which is the core operation of neural networks. The accelerator is based on a multi-sub-array macro to increase parallelism, integrates multiplication operations within the memory cells such that they are executed while reading the cells, makes use of pipelining to further optimize the throughput of the MAC operations, and gets rid of the expensive adder-tree structures commonly used in State-of-The-Art (SOTA) digital CIM solutions by replacing them with a custom accumulation circuit to reduce power and area. The SPICE simulation results of the DREAM-CIM accelerator show an energy efficiency of 5097 TOPS/W (normalized to a 1-bit × 1-bit MAC operation) and an area efficiency of 3854 TOPS/mm^2 using 22 nm technology node. The obtained circuit-level results were fed into a python-based system-level simulator to benchmark the system architecture using two applications, i.e., image classification (using MNIST and CIFAR-10 dataset on LeNet5 and Resnet-20 models) and object detection (using COCO dataset on the YoloV6 model). The system-level results show that DREAM-CIM can achieve an energy efficiency of 0.1mJ, 0.2mJ, and 11.02mJ per inference for the MNIST, YOLOv6, and CIFAR-10 datasets, respectively, while maintaining SOTA accuracy.**

***Index Terms*—Computation-in-memory, SRAM, digital, MAC.**

I. Introduction

ARTIFICIAL Intelligence (AI) technology has been adopted by a wide range of applications with accelerated economic and societal impacts. Today's AI solutions mainly target cloud and data centers; they are expensive, power-hungry, consume lots of silicon area, and off-chip memory burns power and taxes memory bandwidth [1], [2]. This makes them unsuitable for energy-constrained edge applications. Recently, AI computing on the edge for Internet-of-Things (IoT) started to gain popularity. It is expected to revolutionize data computing [3]. The total IoT market is forecasted to be over 600 billion US$, while edge computing will reach a market share of 28.8 billion US$ by 2025 [4]. The fundamental operation of edge AI models, in particular neural networks (NN), is a vector-matrix multiplication consisting of multiply-and-accumulate (MAC) operations. To efficiently perform MAC operations on edge devices, new computing paradigms like *Computation-in-memory (CIM)* have been proposed. CIM aims at addressing most of today's computing challenges by integrating computation and storage in the *same* physical location [5], [6], [7], [8], [9], [10]. This integration overcomes the data transfer and bandwidth challenges of conventional architectures, offering significant parallelism to enable energy-efficient implementation of MAC operations [5], [11]. CIM can be implemented using both conventional SRAM or (emerging) embedded non-volatile memory devices such as resistive random access memory (RRAM) [12], [13], magnetic (STT-MRAM) [14], phase change (PCM) [15], or even ferroelectric field transistor (FeFET) [16]. Such non-volatile devices are competitive technologies for edge computing as they can be turned on and off without losing the stored weight. Hence, their usage eliminates the need to reload data when the system is turned on. In addition, they offer a high chip density due to their small device sizes, and several of these technologies offer multi-bit storage per cell, further increasing the integration density [13], [15]. However, their low endurance, high write energy, variability, and other non-idealities (such as RRAM conductance drift) limit the CIM computing accuracy [17]. Moreover, these technologies often rely on bulky Analog-to-Digital Converters (ADCs), which further reduce the overall power efficiency of the designs [18]. In contrast, SRAM offers a lower write energy per bit and has unlimited endurance [19]. Moreover, SRAM is a mature technology that is already in use

Received 14 December 2024; revised 29 April 2025; accepted 26 May 2025. Date of publication 16 June 2025; date of current version 5 September 2025. This work was supported by CONVOLVE under Grant 101070374. The review of this article was arranged by Associate Editor Jongsun Park. *(Corresponding author: Asmae El Arrassi.)*

Asmae El Arrassi, Lucas Huijbregts, Anteneh Gebregiorgis, Mottaqiallah Taouil, and Said Hamdioui are with the Department of Quantum and Computer Engineering, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: a.elarrassi@tudelft.nl; l.c.a.huijbregts@tudelft.nl; a.b.gebregiorgis@tudelft.nl; m.taouil@tudelft.nl; s.hamdioui@tudelft.nl).

Manil Dev Gomony is with Eindhoven University of Technology, 5612 AZ Eindhoven, The Netherlands (e-mail: m.gomony@tue.nl).

Francky Catthoor is with IMEC, 3001 Leuven, Belgium (e-mail: francky.catthoor@imec.be).

Rajiv Joshi is with Thomas J. Watson Research Center IBM, New York, NY 10598, USA (e-mail: rvjoshi@us.ibm.com).

Digital Object Identifier 10.1109/TCASAI.2025.3579709

as on-chip memory in microprocessors and is commercially available in advanced technology nodes. These properties make SRAM-based CIM suitable for a wide range of edge applications [19], [20], [21]. However, SRAM-based CIMs typically rely heavily on power-hungry components such as adder-trees and ADCs [9], [19]. Therefore, developing energy-efficient SRAM-based CIM implementing MAC operations for edge AI is of great importance.

The State-of-the-Art (SOTA) SRAM-based CIM can be classified into two classes depending on the nature of MAC operation: *analog* CIM (A-CIM), which implements the MAC operation in an analog manner, and *digital* CIM (D-CIM) which implements the MAC operation in a purely digital manner. A-CIM exploits Kirchhoff's current law or the charge sharing to perform MAC operations, where the amplitude of the resulting current or voltage represents the computed output [18], [22], [23], [24], [25]. However, these outputs require current-based or voltage-based ADCs, which limits the exploitation of the high parallelism nature of CIM due to the high energy and area penalties associated with the conversion process. D-CIM addresses some of these challenges as it avoids the costly ADCs by performing the operation in the digital domain. Depending on how the accumulation is implemented at the periphery, D-CIM can be classified into two sub-classes: (a) *sequential* accumulation and (b) adder-tree based *parallel* accumulation. In sequential accumulation, the MAC operation is realized sequentially by selecting only one memory row at a time and performing the multiplication in the cell or using dedicated multiplication units in the periphery; the accumulation is then performed using a dedicated adder that accumulates the outputs of the multiplications in a sequential manner [26]. Although this approach has a low power and area consumption, it is energy-inefficient due to its high latency and low throughput. On the other hand, the adder-tree-based parallel approach makes use of some parallelism by selecting multiple memory rows simultaneously; the partial product/multiplications are then summed through multiple stages of adders, which feed the results to shift and accumulation blocks [27], [28], [29], [30], [31]. However, the additional circuitry to perform multiplication and the bulky adder-trees dominate the energy and area consumption. For instance, for the design reported in [27], the adder-tree consumes ≈73% and ≈61% of the CIM macro power and area consumption, respectively (see Section II-B). Hence, optimizing the energy and area overhead, e.g., the adder tree, for digital SRAM-based CIM computing is crucial not only for energy-efficient computing but also for higher computing density.

This paper advances the state-of-the-art by proposing an adder-tree *free* digital SRAM-based CIM accelerator for (edge) AI applications; it is referred to as *DREAM-CIM*. The architecture is based on a multi-sub-array to realize parallelism and integrates multiplication within the memory cells such that it is executed while reading the cells. In addition, it makes use of pipelining to further optimize the execution time of the MAC operations and gets rid of the expensive adder-tree structures by replacing them with adders. In brief, the architecture integrates the following concepts:

- Area and energy optimization: An efficient hierarchical design for the accumulation is designed not only to minimize the complexity of the design but also to facilitate the pipelining of the macro design. The proposed accumulation periphery uses minimal hardware resources, resulting in high energy and area efficiency.
- Embedded bitwise multiplication: As opposed to standard 6-transistor SRAM cells, 8-transistor cells are used to allow "free" bitwise multiplication by leveraging the memory read operation as a NAND operation.
- Throughput improvement: Eight SRAM sub-arrays with flying Read Bit-Lines (RBLs) are used to enhance parallelism. Each array comprises 16 × 128 cells. The weight matrix is stored in the sub-arrays, and the input vector (in total, eight inputs at a time, each 4-bit wide) is split into four single-bit input slices (where one slice contains 8 1-bit inputs). The eight input bits of a slice are applied simultaneously to the eight sub-array (one bit per sub-array) to perform eight embedded bitwise multiplications in parallel.
- Execution time reduction: A pipeline stage between the memory and the accumulation logic is used to reduce latency. By isolating the memory read operation as the critical path, the design optimizes its timing constraints. This adjustment allows for a higher operational frequency, as the critical path no longer involves complex accumulation logic.

A thorough evaluation of the DREAM-CIM was performed, both at the circuit level as well as system level. At the circuit level, the design was synthesized using the 22 nm technology node and simulated in SPICE to derive the key performance indicators. The results show that the proposed DREAM-CIM achieves an energy efficiency of 318 **T**OPS/W and an area efficiency of 241 **T**OPS/mm^2 for a 4-bit × 4-bit MAC operation. When normalized to a 1-bit × 1-bit MAC operation, the results change into 5.1 **P**OPS/W and 3.9 **P**OPS/mm^2 energy and area efficiency, respectively. Hence, DREAM-CIM realizes, on average, 2× and ≈1.6× energy and area savings over the conventional adder-tree-based digital CIM approaches. At the system level, the obtained circuit-level results were fed into a Python-based high-level system simulation to benchmark the system architecture using two applications. The first application is image classification using the MNIST [32] dataset on LeNet-5 [33] and CIFAR-10 [34] on Resnet-20 [35], achieving 0.16 mJ/inference at 99.4% accuracy and 11 mJ/inference at 89% accuracy, respectively. The second application is object detection using the COCO dataset [36] on the YoloV6 model [37]. It achieves 0.2 mJ/inference with a 42.6 mean Average Precision (mAP), which is the SOTA accuracy level for quantized object detection models. mAP measures the accuracy of object detector models based on several metrics, such as precision and recall. Overall, the results show that DREAM-CIM achieves, on average, 2× energy savings per inference over the adder-tree-based digital CIM approaches while maintaining SOTA accuracy for image classification and object detection applications.

TABLE I
SOTA OPTIMIZATION TECHNIQUES ACROSS DIFFERENT LEVELS OF ABSTRACTION

Levels of Abstraction	**Optimization Techniques**	CICC'11 [38]	SSCL'21 [26]	ISSCC'21 [28]	JSSC'21 [30]	ISSCC'22 [27]	ISSCC'23 [39]
Micro-architecture	In-cell Computation	Yes	Yes	No	Yes (MAC operation)	No	Yes (Multiplication)
	In-periphery Computation	Accumulation	Accumulation	MAC operation	N/A	MAC operation	Accumulation
	Parallelism	No	No	Adder-tree	Systolic array	Multi-bank memory + Adder-tree	Multi-bank memory +Adder-tree
	Reconfigurable Operands Precision	N/A	N/A	4-8-12-16 bits	1-16 bits	N/A	4-8-12 bits
Circuit	Custom Bit-cell Design	Transposable 8-T SRAM	10-T SRAM	Standard	6-T SRAM + XNOR+ FA	12-T SRAM	8-T SRAM
	Low-power Design	N/A	N/A	Low-power adders (14T)	N/A	Low-power adders (24T)	Mixed Vt design
Device	Technology Node	65nm	45nm	22nm	65nm	5nm	4nm

The remainder of the paper is organized as follows. Section II highlights the prior work and its shortcomings. Section III discusses the proposed architecture. Section IV presents the macro-level simulation results and State-of-the-Art (SOTA) comparison. Section V reports the system simulation results. Section VI discusses the proposed architecture results and discussion. Finally, Section VII concludes the paper.

II. Prior Works and Their Shortcomings

As already mentioned in the previous section, prior works on SRAM-based CIM can be classified into A-CIM and D-CIM. Next, we will elaborate on each class separately, including the concept, the SOTA, and the shortcomings.

A. Analog SRAM-Based CIM (A-CIM)

In analog SRAM-based CIM (A-CIM), the first operands (e.g., weight values) are stored in the memory array, while the second operands (e.g., activation inputs) are provided through the word lines (WLs). Each column performs the MAC operation by multiplying the input operands and the weights stored in the memory array. The output currents of all the cells in the column are accumulated through the column Bit Lines (BLs) and form the analog MAC result based on Kirchhoff's law [40], [41]. The accumulated output current through the BL is subsequently fed to an ADC to be converted to its digital value. Despite their advantages, such as high parallelism, A-CIM architectures suffer from numerous challenges, such as process variation, significant ADC area and energy overhead, computing non-linearity, limited resolution/accuracy, and limited parallelism [30].

B. Digital SRAM-Based CIM (D-CIM)

Digital SRAM-based CIM (D-CIM) addresses the challenges of A-CIM by eliminating the need for energy and area-hungry ADCs [42] as the computation is performed in the digital domain. Depending on the accumulation operation, D-CIM implementations can be classified into two main categories, namely (a) sequential accumulation (sequential D-CIM), and (b) adder-tree based *parallel* accumulation (parallel D-CIM) [26], [27], [39], [42], [43].

1) Sequential D-CIM: The MAC operation is realized in sequential D-CIM by storing the first operand (e.g., weight) in the memory array (e.g., SRAM array), and the second operand is provided as an input sequentially. The two operands are then fed to multipliers to perform multiplication, and the result is then accumulated sequentially [26], [38], [43].

The SOTA solutions have explored different optimization techniques across different levels of abstraction. For example, the work in [26] has explored input sparsity to ensure an event-driven system with minimal power dissipation; a custom 11-bit adder is used to perform sequential accumulation of the multiplication results of the weights and inputs. The accumulation result is then stored back in the SRAM array, minimizing area at the cost of additional write energy and time overhead. Sequential D-CIM is area-efficient and consumes low power. However, due to the sequential memory accesses, these approaches need more cycles per single MAC operation and suffer from low throughput, which results in high energy consumption [26], [43].

2) Parallel D-CIM: In the parallel D-CIM approach, multiple rows in the memory array are accessed in parallel and they are provided to multipliers to perform multiplication. The multiplication outputs are then fed to adder trees, which consist of multiple stages of adders to perform the accumulation in a single clock cycle [27], [39].

The parallel D-CIM SOTA solutions, summarized in Table I, have explored various techniques to improve energy efficiency across different levels of hardware abstraction, including microarchitecture, circuit, and device levels.

At the microarchitecture level, several works have explored in-cell and in-periphery computation [27], [30]. These architectural decisions have a huge impact on the overall energy and area consumption. The first adder-tree-based D-CIM was proposed in [28] and reported at that time the highest throughput. However, these solutions come at the cost of increased area and energy consumption.

At the circuit level, SOTA solutions have proposed energy optimization techniques by introducing custom bit cell designs to enable computation within the cell or to enhance the stability of the SRAM cell design [30], [39]. Moreover, different types of adders such as Carry Look-ahead Adders (CLA) and

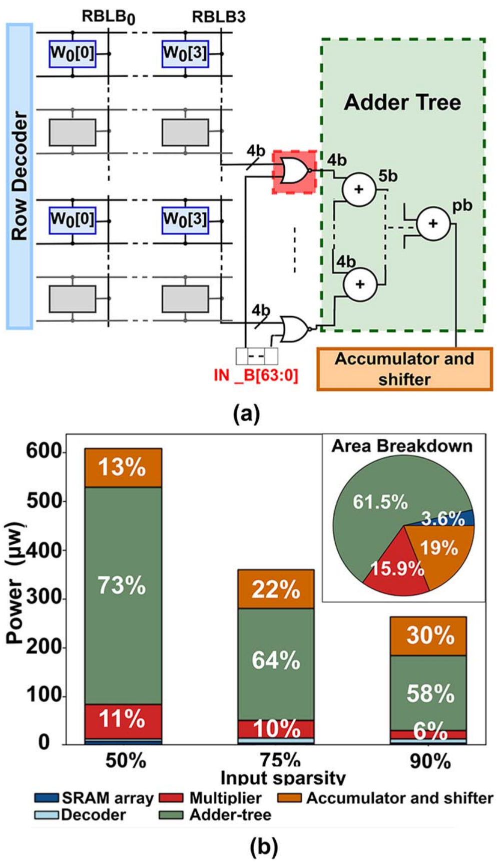
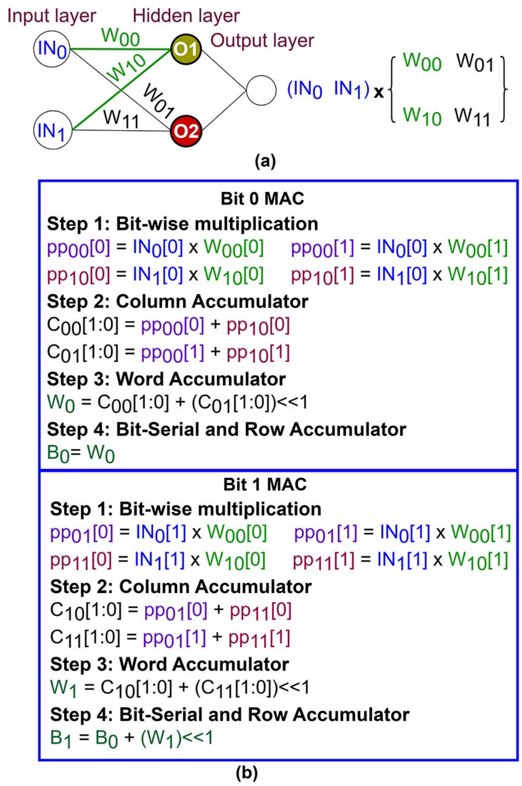


Fig. 1. An illustration of a) digital CIM architecture and b) energy and area breakdown for 32×32 SRAM-based array for 4-bit weights in 28 nm CMOS technology for different input sparsity [27].



Bit 0 MAC

Step 1: Bit-wise multiplication

$pp_{00}[0] = IN_0[0] \times W_{00}[0]$ $pp_{00}[1] = IN_0[0] \times W_{00}[1]$

$pp_{10}[0] = IN_1[0] \times W_{10}[0]$ $pp_{10}[1] = IN_1[0] \times W_{10}[1]$

Step 2: Column Accumulator

$C_{00}[1:0] = pp_{00}[0] + pp_{10}[0]$

$C_{01}[1:0] = pp_{00}[1] + pp_{10}[1]$

Step 3: Word Accumulator

$W_0 = C_{00}[1:0] + (C_{01}[1:0]) << 1$

Step 4: Bit-Serial and Row Accumulator

$B_0 = W_0$

Bit 1 MAC

Step 1: Bit-wise multiplication

$pp_{01}[0] = IN_0[1] \times W_{00}[0]$ $pp_{01}[1] = IN_0[1] \times W_{00}[1]$

$pp_{11}[0] = IN_1[1] \times W_{10}[0]$ $pp_{11}[1] = IN_1[1] \times W_{10}[1]$

Step 2: Column Accumulator

$C_{10}[1:0] = pp_{01}[0] + pp_{11}[0]$

$C_{11}[1:0] = pp_{01}[1] + pp_{11}[1]$

Step 3: Word Accumulator

$W_1 = C_{10}[1:0] + (C_{11}[1:0]) << 1$

Step 4: Bit-Serial and Row Accumulator

$B_1 = B_0 + (W_1) << 1$

Fig. 2. An illustration of a) an example of an NN and b) a simplified example of the MAC operation steps for one output neuron.

reversed polarity adders are used to reduce latency and area consumption. However, these have minimal impact on energy and area improvements as the adder-tree still dominates the area and energy consumption of the D-CIM.

To illustrate the dominance of the adder-tree, we implemented the design proposed in [27]. We synthesized and performed post-layout simulations. The estimated results are illustrated in Fig. 1. The results in the figure show that the adder-tree consumes 73% power and ≈61% area of the CIM macro, while the combination of multipliers, memory array, and accumulators consumes only 24% power and 19% area. Therefore, area and power-efficient alternatives are crucial to address the overhead of adder-tree on SRAM-based digital CIM.

III. DREAM-CIM Architecture

This section introduces the DREAM-CIM architecture. First, it explains the DREAM-CIM concept using a simplified example. Thereafter, it presents an overview of the DREAM-CIM architecture followed by the implementation details.

A. DREAM-CIM Concept

To illustrate the concept, we use the NN shown in Fig. 2(a) as an example. It consists of two input neurons, IN_0 and IN_1, two neurons, O1 and O2, in the hidden layer, and an output neuron. Each neuron is represented by a two-bit number. The right side of the figure shows for O1, which has weights W_{00} and W_{10}, the MAC operations that needed to be performed. In the proposed DREAM-CIM accelerator, a MAC operation is implemented in four distinct steps, as shown in Fig. 2(a) for neuron O1.

Step 1: Bit-wise Multiplication - In this step, a bit-wise multiplication between the first bit of each input ($IN_0[0]$ and $IN_1[0]$) and the weights ($W_{00}[1:0]$ and $W_{10}[1:0]$) are performed; this results in two partial products (pp), namely $pp_{00}[1:0]$ and $pp_{10}[1:0]$.

Step 2: Column Accumulator - In this step, bits belonging to the same position within each pp are accumulated together (i.e., column accumulation). In the example, $pp_{00}[0]$ is accumulated with $pp_{10}[0]$, and $pp_{00}[1]$ with $pp_{10}[1]$. This results in a two-bit intermediate result for each column accumulation, namely $C_{00}[1:0]$ and $C_{01}[1:0]$.

Step 3: Word Accumulator - In this step, the results of the Column Accumulator are added. It is required that the partial sums of the Column Accumulator are properly aligned here, e.g., a 1-bit left shift operation is performed on C_{01}. The result is the partial sum W_0.

Step 4: Bit-Serial and Row Accumulator - Finally, in the last step, we add the partial results of the inputs that are applied serially. For example, Fig. 2 shows the same steps for the second bit of the inputs $IN_0[1]$ and $IN_1[1]$. However, in step four, the resulting W_1 is shifted with one bit to the left to compensate for the input weight and then accumulated with the previous MAC cycle result B_0. Note that as we access only one row at a

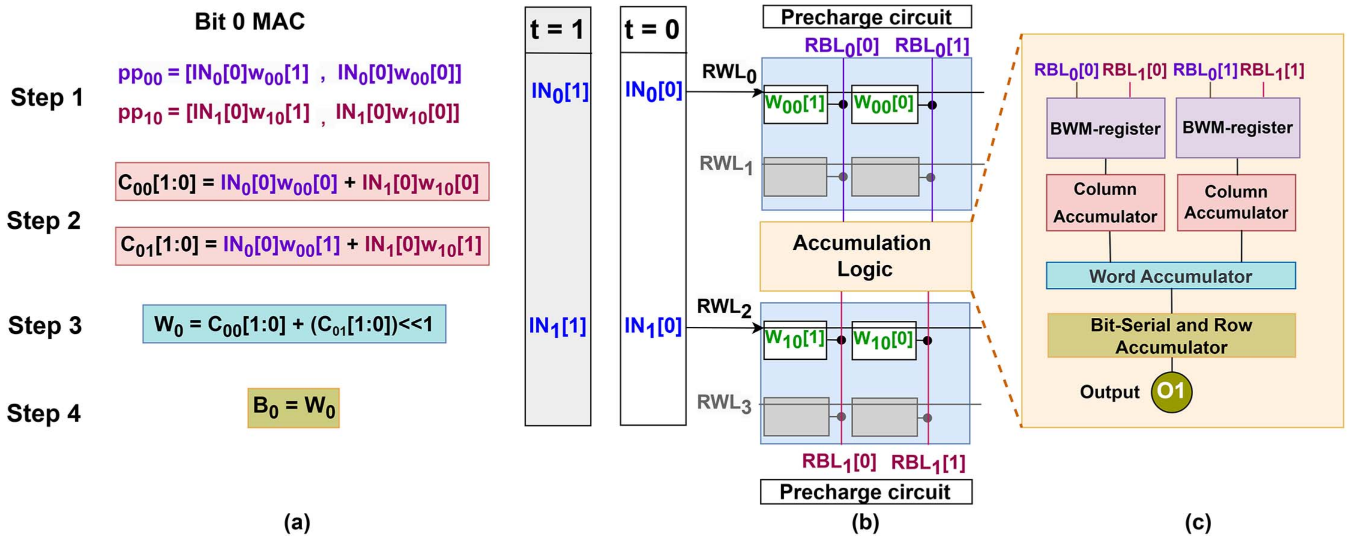


Fig. 3. An illustration of a) the different steps to perform the MAC operation using the DREAM-CIM accelerator, b) conceptual DREAM-CIM accelerator, c) the accumulation logic.

time per sub-array, we need a Row Accumulator to sum up the results when activating the next rows. This case is not shown in the figure.

The MAC operation steps of DREAM-CIM architecture for one output neuron can be generalized as follows:

$$O_x = \sum_{k=0}^{M-1}\sum_{j=0}^{M-1}\sum_{i=0}^{N-1}((IN_i[k]W_{ix}[j]) \ll j) \ll k) \qquad (1)$$

Where Ox is the output neuron index, M is the bit width of the inputs and the weights, and N is the number of sub-arrays. The innermost summation iterates over the sub-arrays i, where each term performs a bit-wise multiplication between the k-th bit of the input I_i and the j-th bit of the weight W_{ix}. The resulting partial product is then left-shifted by j and k bits to correctly align it in the final accumulation, accounting for its bit significance. The nested summations over j and k effectively reconstruct the full-precision MAC operation.

Fig. 3 illustrates a simplified example of the mapping of the MAC operation on the DREAM-CIM architecture. The performed operation is illustrated in Fig. 3(a). the bit-wise multiplication is performed between the first operand stored in the memory cell (i.e., a weight) and the second operand provided as input through the Read World-line (RWL). The weights are stored in a row-wise fashion using multiple columns (two for a 2-bit number) of the memory, while the input bits are provided sequentially to the memory as illustrated in Fig. 3(b). For simplification, the memory array is composed of two sub-arrays, one above and one below the accumulation logic. At any time step, only one row is selected/activated per sub-array. This means for a memory array with 8 sub-arrays, 8 rows are activated at a time. Inside the cells of each sub-array bitwise multiplication is performed. Their results are read through the Read Bit-lines (RBLs). Each bit line of each sub-array will contain the result of one bit-wise multiplication (i.e., one partial product). Bitlines with the same column index will store the result in the same 2-bit register on different indices as shown in Fig. 3(c); note that these indices have the same weight bit-value; e.g., RBL_0[0] and RBL_1[0] feed the same 2-bit register. The register receives parallel inputs and provides a serial output to the Column Accumulator. The Column Accumulator accumulates the bit-wise multiplication results of each RBL, which provides an intermediate result C_{00}[1:0] and C_{01}[1:0] (see also Fig. 2(b)). C_{00} and C_{01} are aligned depending on the column's bit position of the word and then accumulated using the Word Accumulator to get the partial sum W_0 as shown in Fig. 3(c). Finally, the Bit-Serial and Row Accumulator have dual functionality: (a) shift and accumulate the partial sum results W of different input bit-position, and (b) accumulate the partial sum results of different rows of each sub-array as only one row is activated at a time in each sub-array.

B. DREAM-CIM Implementation

In our case study, we target an architecture based on a single 16kb SRAM array. Note that the overall capacity can easily be scaled by replicating the individual macro. The memory array of DREAM-CIM is partitioned into multiple sub-arrays to increase the level of parallelism by activating multiple rows simultaneously, as shown in Fig. 4. To optimize the design, we have chosen a configuration based on 8 sub-arrays, which are further divided into two arrays of 4 sub-arrays (i.e., 0 to 3 and 4 to 7).

This division is strategically selected as we can efficiently fit four flying RBLs over the SRAM with minimal area overhead, as shown in Fig. 5. The flying RBLs contain partial products belonging to the same neuron. These four RBLs align seamlessly on top of the SRAM cell, ensuring a minimal area penalty while enabling parallel data access. This configuration balances the performance and area efficiency of the DREAM-CIM accelerator. By mirroring two sets of 4 sub-arrays with the

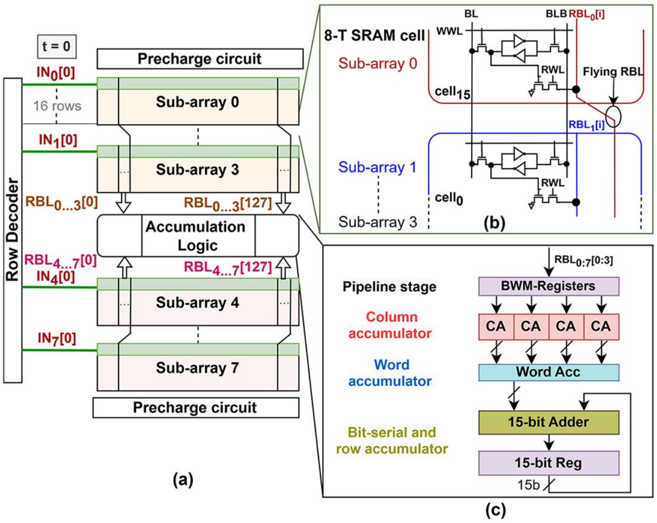


Fig. 4. DREAM-CIM Hardware architecture a) multi Sub-array memory array, b) the SRAM-cell design, and c) periphery for accumulation.

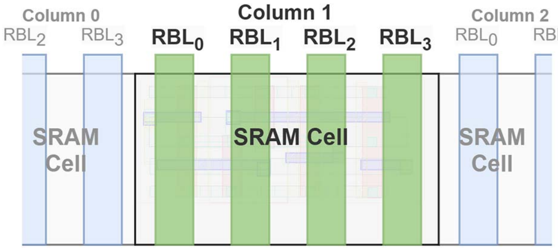


Fig. 5. The four flying RBLs in one column of the proposed SRAM macro.

accumulation logic in between, we can effectively access eight sub-arrays in parallel.

Fig. 4(a) shows the macro architecture of DREAM-CIM for eight inputs (each 4-bit wide and belonging to a different sub-array), 4-bit weights, and eight memory sub-arrays of 2kb each. Each sub-array consists of 16 rows and 128 columns; each row stores 32 weights, each 4-bit wide. The sub-arrays consist of two-port 8T-SRAM cells as shown in Fig. 4(b): one Read/Write (R/W) port and one Read-Only (RO) port. Cells in the same column of all sub-arrays share the same R/W bit-lines (i.e., BL and BLB for the R/W port). The Read Bitlines (RBLs) are connected to all cells in a column in a single sub-array and fly over the other sub-arrays to connect to their dedicated pipelining register at the border between the SRAM and accumulation circuitry. Each of the four RBLs connects to a different sub-array. Cells in the same row share the same word line (i.e., Write World-Line (WWL) for the R/W port and Read World-Line (RWL) for the RO port). As such, one RWL can be activated in each sub-array without creating read conflicts, allowing for eight parallel reads.

Fig. 4(c) illustrates the periphery of the whole accumulation process. The accumulation process starts by latching the partial products (step 1 in Fig. 3) via the column bit line values using 8-bit registers containing partial products of the eight different sub-arrays. In total, there are 128 8-bit registers, referred to as Bit-Wise Multiplication (BWM) -Registers in Fig. 4(c). Although the BWM-Registers are strictly not needed, the accelerator frequency can be doubled, improving the throughput twice. The outputs of the BWM-Registers are connected to the Column Accumulator for column-wise accumulation (step 2 in Fig. 3). The Column Accumulators add the eight 1-bit partial products to a single 4-bit number (step 2 in Fig. 3). Thereafter, the Word Accumulators (step 3 in Fig. 3) add the 4 bits of partial results together; as the inputs are applied sequentially, there are four partial results that are added. During their addition, proper shifts are in place to adjust the input weights. The result is a 7-bit number. Finally, the Bit-serial and Row accumulator adds the output of the Word accumulator to the running accumulation and store it in the accumulation register (step 4). As the input activations are provided row-by-row and bit-by-bit to the SRAM, this accumulator needs to keep track of the total accumulation over all the cycles needed to perform the full MAC. Additionally, it performs shifting of the running accumulation to account for the bit significance of the input activations. Note that the Word Accumulator, Bit-Serial, and Row Accumulator are shared between 4 columns of the SRAM.

TABLE II
SIMULATION PARAMETERS

Technology	22 nm
Supply voltage (V)	0.8 V
Simulations	SPICE-level (Cadence Spectre)
Parasitic extraction	Calibre, PEX
Synthesis	Cadence Genus
Temperature	27 °C
SRAM cell	8-T
Unit macro size	16 kb (128x128b)

IV. CIRCUIT LEVEL SIMULATIONS

A. Experiment Setup

The proposed DREAM-CIM architecture is simulated in SPICE using GlobalFoundries 22 nm CMOS technology. The simulation parameters are presented in Table II. The energy and latency results are extracted using post-layout SPICE-level simulations. The area of the SRAM array has been derived from the custom layout design, and for its surrounding periphery logic using Cadence Genus synthesis results. It is worth noting that the results were reported for the worst-case scenario where we used a bit-sparsity of 50%, i.e., 50% of the weight bits were initialized to ‘0’ and the other 50% to ‘1’.

B. Energy and Area Results

To evaluate the energy of the macro, an SRAM array made up of 4 sub-arrays, each 16×128 cells, was simulated at SPICE level using the GF 22 nm technology. Parasitics were added to the word lines and bit lines of the array, derived from PEX extraction results based on foundry-provided 8T SRAM cells. Additionally, the accumulation circuitry, decoding, and control

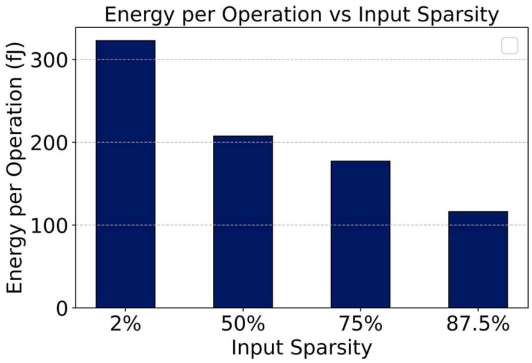


Fig. 6. Energy consumption/operation for different input sparsity.

logic were all synthesized to a gate-level netlist using Cadence Genus and simulated at the SPICE level. Fig. 6 shows the energy per MAC operation for different input sparsity levels. As the input sparsity increases (i.e., more zeros in the input), the energy per operation decreases significantly, highlighting the efficiency gains from sparsity. As a result, the recorded energy efficiency and peak throughput are 318 TOPS/W and 1024 GOPS, respectively.

Fig. 7(a) shows the energy breakdown of the DREAM-CIM components, which are the SRAM array, BWM-Register, Accumulation logic (Acc Logic), and Acc-register. As shown in the figure, the energy consumption of the accumulation circuitry (Acc Logic and Acc Registers) comprises 46% of the total energy consumption, a comparable amount to the energy consumption of the SRAM array at 40%. The remaining 14% of the energy is consumed by the Mem Registers.

Fig. 7(b) illustrates the layout of the macro, which has an area of 0.017 mm^2. Fig. 7(c) shows the breakdown of this area between the SRAM array and Accumulation logic and I/O. The 16 kb SRAM array occupies 46.6% of the total area, while the accumulation logic occupies the remaining 53.4% of the total area. Fig. 7(d) shows the area breakdown of all the accumulation logic and the registers. The accumulation logic consumes 71% of the total accumulation area, while the BWM-Registers consume 19%, and the remaining registers consume 9% of the total area.

C. State-of-the-Art Comparison

Table III shows a comparison of the proposed DREAM-CIM accelerator and state-of-the-art SRAM-based digital CIM accelerators. The SOTA solutions can be classified as Sequential D-CIM [26] or Parallel D-CIM [27], [28], [29], [30], [31]. The work in [26] presented a Sequential D-CIM SRAM-based CIM. In this approach, memory rows are accessed sequentially, and a dedicated accumulator is assigned to each column to perform MAC operations, resulting in a low area and power consumption. However, the sequential accumulation significantly increases the overall execution time of the MAC operation, which results in high energy consumption, as shown in Table III. The first parallel D-CIM-based solution was proposed in [28], where an adder-tree structure was used to increase the parallelism. Since then, several solutions have explored different adder-tree structures. For instance, [31] proposed a compressor-based adder-tree. In [29], the authors used look-up tables to replace a single stage of the adder-tree. Instead of the traditional 28-T full adders, the authors in [27] used 24-T full adders with reversed inputs. Despite these improvements, the adder-tree still dominates the power and area consumption of the D-CIM macro.

In Table III, we summarized some key metrics to evaluate DREAM-CIM compared to SOTA solutions. The estimated results show an impressive energy efficiency of 318 TOPS/W and area efficiency of 78.7 TOPS/mm^2. For many SOTA accelerators, two sets of measurements are reported. One at a higher supply voltage and frequency, focusing on throughput and area efficiency by enabling faster operations, and one at a lower voltage and frequency, focusing on energy efficiency for more energy-constrained applications. In Table III, both results are shown, and a similar dual measurement is reported for DREAM-CIM; one for the nominal supply voltage of $0.8V$ with the frequency of 2 GHz and one at $0.6V$ with the frequency of 1 GHz. For DREAM-CIM, the lower supply voltage results in 1.68× higher energy efficiency at the cost of halving the throughput. Finally, DREAM-CIM achieves a high energy efficiency while occupying a relatively low area with an area efficiency of 78.7 TOPS/mm^2. Overall, the proposed solution effectively balances performance, area, and energy efficiency, offering a highly optimized solution.

In Table III we have added in the final column a projection of the performance of DREAM-CIM when implemented in 7nm technology node. The macro size is scaled down 4 times based on the SRAM scaling factor from 22nm to 7nm provided in [45]. The power and delay are projected using the following scaling factors: 2.28 and 1.17, respectively [46]. The estimated results show an impressive energy efficiency of 860.3 TOPS/W and area efficiency of 330.6 TOPS/mm^2, which heavily outperforms the SOTA.

D. Technology Normalized SOTA Comparison

The most recent studies, such as [39] and [44], have prioritized reconfigurability and enhanced bit-density. However, these improvements come with significant area overheads and relatively lower energy efficiency. To the best of our knowledge, the design in [27] achieves the highest energy and area efficiency in the state-of-the-art.

For a fair comparison, we simulated both our design and the design in [27] using post-layout extraction in the 22 nm technology. This comparison focuses specifically on the MAC operation periphery, which for DREAM-CIM comprises the Acc Logic and Acc Registers, and for [27] includes the bit-wise multipliers, adder-tree, and accumulator periphery in the design proposed in [27]. The two designs are tested using similar array sizes, input vectors, and weight matrices to ensure similar operand sparsity and accumulation results. Fig. 8 compares the two designs in terms of area consumption, execution time, energy consumption, and a combined metric, i.e., the product of the aforementioned metrics. The comparison includes results

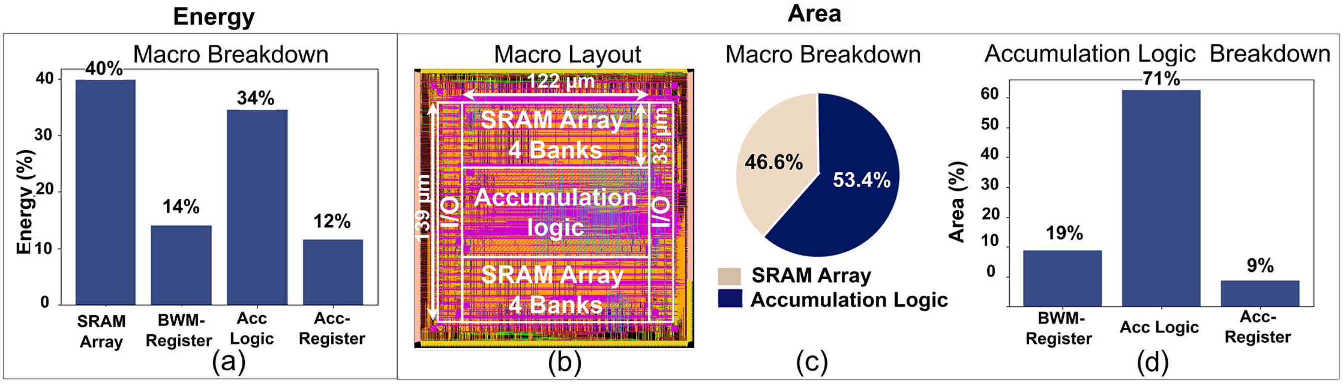


Fig. 7. An illustration of a) the energy breakdown of the different Sub-Components of the DREAM-CIM macro, b) SRAM macro layout, c) the area consumption macro breakdown, and d) accumulator breakdown.

TABLE III
State-of-the-Art Comparison of the SRAM-Based Digital CIM Macros

	SSCL'21 [26]	ISSCC'21 [28]	ISSCC'22 [27]		ISSCC'23 [39]		ISSCC'24 [44]		**DREAM-CIM**		**DREAM-CIM Projection**
Technology (nm)	65	22	5		4		3		22		7
Supply Voltage (V)	0.85	0.92	0.9	0.5	0.9	0.32	0.9	0.4	0.8	0.6	0.6
Input/Weight Precision (bit)	1/6	(1-8)/4	4/4		8/8		12/12		4/4		4/4
Peak Throughput (GOPS)	0.2 (6b)	5900 (4b)	2950 (4b)	740 (4b)	858 (8b)	235 (8b)	7771 (4b)	1438 (4b)	1024 (4b)	512 (4b)	600.9 (4b)
Energy efficiency (TOPS/W)	0.99 (6b)	42.4 (4b)	69.9	254 (4b)	24.9 (8b)	87.4 (8b)	N/A	484 (4b)	189.7 (4b)	318.6 (4b)	860.3 (4b)
Energy efficiency (TOPS/W/bit)	5.94	672.4	1118	4064	398.4	6163.2	N/A	7744	3035	5097.6	13760.1
Array size (Kb)	11	64	64		54		60.75		16		16
Bitcell area (μm^2)	N/A	0.379	0.075		0.161		0.026		0.18		0.053
Macro area (mm^2)	0.089	0.202	0.0133		0.0172		0.0157		0.013		0.004
Area efficiency (TOPS/mm^2**)**	0.0022 (6b)	29.21	221.2 (4b)	55.3 (4b)	49.9 (8b)	13.7 (8b)	495 (4b)	94.5 (4b)	78.7 (4b)	39.3 (4b)	228.6 (4b)
Area efficiency (TOPS/mm^2**/bit)**	0.012 (6b)	407.3	3539.2	884.8	3193	876.8	7920	1512	314.96	628.8	4297.6

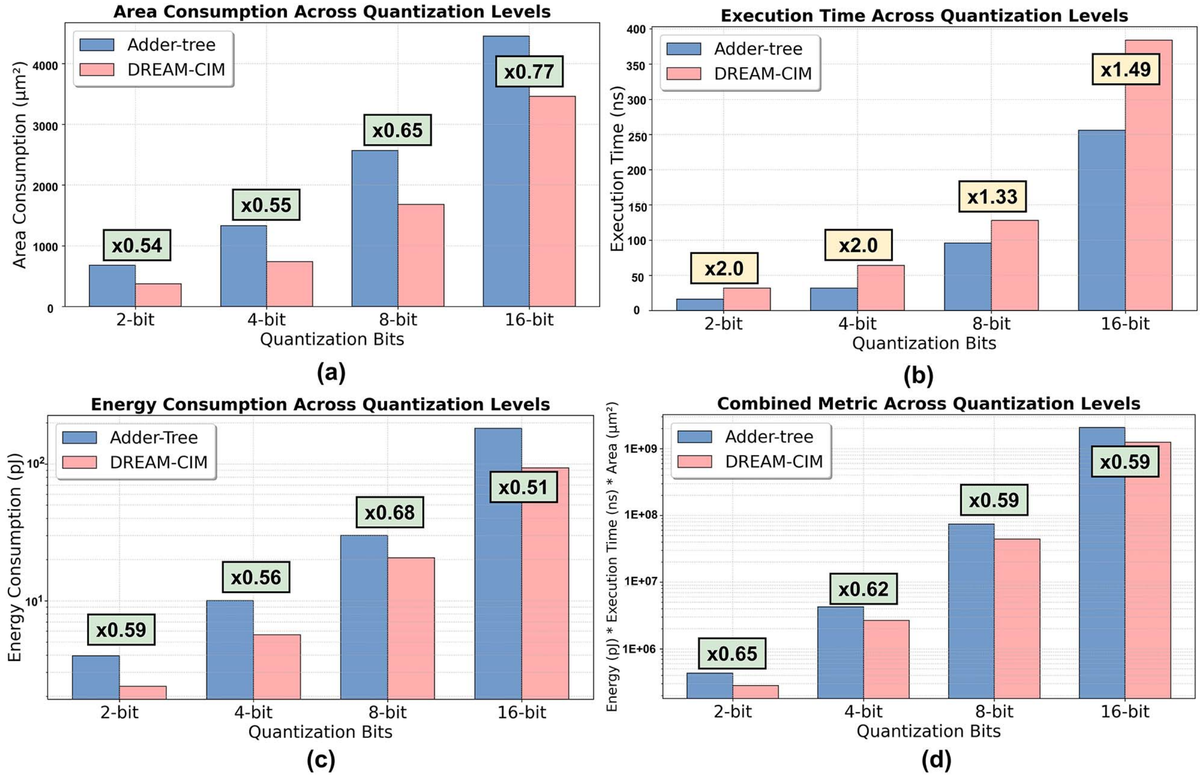


Fig. 8. a) Area consumption, b) execution time, c) energy consumption, and d) combine metric product across quantization levels comparison of our proposed architecture and an adder-tree based approach [27].

TABLE IV
SYSTEM LEVEL SIMULATION SETUP

Application	Image Classification		Object Detection
Topology	LeNet5	Resnet-20	Yolov6-S
Datasets	MNIST	CIFAR-10	COCO
Complexity	Low	High	High
Number of parameters	60K	268K	41K

for both designs for various input and weight precisions. Note that the proposed DREAM-CIM architecture supports a fixed-bit configuration only. We have made custom designs for each case. The reported measurements are taken for MAC operations using always 128 rows and either 2, 4, 8, or 16 columns, corresponding to 2-bit, 4-bit, 8-bit, and 16-bit weight configurations, respectively. The figure demonstrates that our proposed solution outperforms the adder-tree-based approach [27] in both energy and area efficiency (see Fig. 8(a), 8(c)). On average, DREAM-CIM achieves 1.83× and 1.67× improvements in area and energy efficiency, respectively. It is important to note that the area improvement reported here is only based on the accumulation logic, and the SRAM area is assumed to be equal between the designs. In reality, the SRAM area for the adder-tree design will be even larger as the SRAM is split into many smaller banks. Each of these banks needs to be surrounded by its own set of edge cells. As such, splitting the SRAM into many banks results in an even lower area efficiency. Fig. 8(b) shows that DREAM-CIM has a higher execution time compared to the adder-tree approach. However, this penalty is acceptable, as the energy efficiency (Fig. 8(c)) and combined metrics (Fig. 8(d)) demonstrate significant benefits over the adder-tree approach.

V. SYSTEM LEVEL SIMULATIONS

A. Simulation Setup

To evaluate the performance of DREAM-CIM at the system level, we analyze the accuracy and energy of inference across the selected datasets. This includes examining the input and weight sparsity of each layer and calculating the number of operations performed per inference. SPICE simulations are set up using data derived from software simulations, and these results are subsequently fed into a system-level simulator to extract system-level evaluation metrics [47].

We analyze two types of applications: image classification and object detection. Table IV shows the characteristics of the selected neural networks and their associated data sets. For image classification, we implement the LeNet5 [33] and ResNet-20 [35] models trained on the MNIST [32] and CIFAR-10 [34] datasets, respectively. For object detection, we implement the Yolov6-S model [37] trained on the COCO dataset [36]. To be able to map the floating-point weights and activation function results of these neural networks to the integer-based macro, they need to be quantized. Quantization is performed using the QNN method [48] due to its high accuracy, and the network is trained off-line on the RedBit framework [47]. The training is performed using back-propagation [49] with stochastic gradient descent [50]. Subsequently, the trained networks are mapped on the DREAM-CIM architecture.

TABLE V
ACCURACY RESULTS

Application	Image Classification		Object Detection
Topology	LeNet5	Resnet-20	YoloV6-S
Datasets	MNIST	CIFAR-10	COCO
Input/weight Precision	4/4	8/8	8/8
Baseline Accuracy	99.6%	91.7%	45.0 mAP
DREAM-CIM Accuracy	99.4%	89.0%	42.6 mAP

B. Accuracy and Energy Results

Fig. 9 shows the energy consumption per inference and accuracy when running the three neural networks on DREAM-CIM and the adder-tree-based architecture proposed in [27]. The results focus on the MAC operation periphery and the bank size of both designs. For each neural network, the input and weight precision have been varied from 2 to 16 bits. As the state-of-the-art solutions (see Table III) did not provide system-level simulations, we have re-implemented the design proposed in [27] at the same 22nm technology node.

Generally, for lower input and weight precisions, the energy requirement per inference is lower at the cost of lower accuracy. To achieve sufficient accuracy, we need at least a 4-bit precision for the LeNet5 and Resnet-20, and an 8-bit precision for Yolov6. These results are summarized in Table V. In the table, the baseline accuracy represents the case where floating point numbers have been used. The row with DREAM-CIM Accuracy shows the accuracy derived for the selected input and weight precision. With the selected precisions, the accuracy loss compared to the baseline is acceptable.

Fig. 9 shows that DREAM-CIM has significantly lower energy consumption for a given weight precision. For example, for Resnet-20, for 8-bit precision, the energy requirement per inference equals 17.5 mJ/inference for the Adder-tree SRAM-Based CIM design, while 3.3 mJ/inference for DREAM-CIM. On average considering all cases, DREAM-CIM needs 4.46x less energy as compared to the Adder-tree SRAM-Based CIM design on the different benchmarks for image classification and object detection.

VI. DISCUSSION

Based on our case studies and results, we would like to discuss the following topics and aspects.

- **Comparison with SOTA:** The different SOTA solutions consisting of parallel D-CIM suffer from the adder-trees as they come with high energy and area overheads. On the other hand, the sequential D-CIM SOTA solutions have a high execution time and energy consumption. Our solution tries to overcome the existing limitations by introducing a design that achieves a high energy and area efficiency while minimizing the performance tradeoff. This is achieved by replacing the adder tree with a sequential adder circuit. To minimize the performance penalties, parallelism is increased using multiple sub-arrays, and a pipeline stage is introduced to allow for a higher operating frequency.

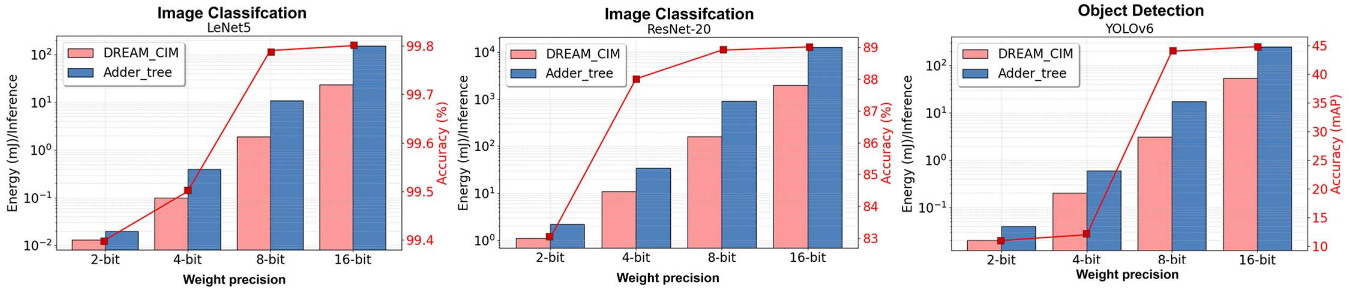


Fig. 9. The energy consumption and accuracy of the DREAM-CIM accelerator and the adder-tree based approach [27] for two applications benchmarks: Image classification (LeNet5, ResNet-20) and object detection (Yolov6). the networks are evaluated using 2,4,8,16-Bits weight precision.

- **Scalability:** The presented methodology to perform MAC operations in the crossbar can be extended to larger crossbar sizes. However, for very large applications, multiple crossbars are needed, and the communication between these crossbars has to be carefully designed to prevent a high latency and energy consumption when moving data between them. Therefore, more research and analysis are needed to support larger applications that need multiple crossbars.
- **Reconfigurability:** Currently, the accumulation circuitry has been designed and optimized for a fixed bit-width (i.e., 4-bit). However, to support diverse applications with different requirements, the accumulation circuitry should be reconfigurable to efficiently support variable widths (e.g., 4-bit, 8-bit, 16-bit, etc.).

VII. Conclusion

In this work we presented DREAM-CIM, an adder-tree-free digital SRAM-based CIM accelerator. The proposed architecture demonstrates gains in energy and area efficiency compared to SOTA. DREAM-CIM is evaluated using circuit-level simulations on GF 22nm technology and system-level simulations using two different use cases (image classification and object detection applications) on different datasets complexities (MNIST, CIFAR-10, and Coco) and topologies (LeNet5, Resnet-20, and YoloV6). The simulations demonstrated more than 2× power and ≈1.6× area savings compared to conventional CIM architectures based on adder-tree approaches. DREAM-CIM opens a new research direction for SRAM-based CIM architectures and shows that there is potential to optimize such architectures further.

References

[1] S. Srinath, O. Mutlu, H. Kim, and Y. N. Patt, "Feedback directed prefetching: Improving the performance and bandwidth-efficiency of hardware prefetchers," in *Proc. IEEE 13th Int. Symp. High Perform. Comput. Archit.*, 2007, pp. 63–74.

[2] D. Fujiki et al., "Multi-layer in-memory processing," in *Proc. MICRO*, 2022, pp. 920–936.

[3] F. Firouzi et al., "The convergence and interplay of edge, fog, and cloud in the AI-driven internet of things (IoT)," *Inf. Syst.*, vol. 107, p. 101840, Dec. 2022.

[4] V. Sharma et al., "Future of wearable devices using IoT synergy in AI," in *Proc. ICECA*, 2019, pp. 138–142.

[5] J. Ahn et al., "A scalable processing-in-memory accelerator for parallel graph processing," in *Proc. 42nd Annu. Int. Symp. Comput. Archit.*, 2015, pp. 105–117.

[6] S. Hamdioui et al., "Memristor based computation-in-memory architecture for data-intensive applications," in *Proc. DATE*, 2015, pp. 1718–1725.

[7] J. Wang et al., "A 28-nm compute SRAM with bit-serial logic/arithmetic operations for programmable in-memory vector computing," in *Proc. JSSC*, 2019, pp. 76–86.

[8] J. Ahn et al., "PIM-enabled instructions: A low-overhead, locality-aware processing-in-memory architecture," in *Proc. ACM SIGARCH Comput. Archit. News*, 2015, pp. 336–348.

[9] A. Sebastian et al., "Memory devices and applications for in-memory computing," *Nat. Nanotechnol.*, vol. 15, no. 7, pp. 529–544, Jul. 2020.

[10] S. Hamdioui et al., "Memristor for computing: Myth or reality?" in *Proc. DATE*, 2017.

[11] S. Hamdioui et al., "Guest editorial: Computation-in-memory (CIM): From device to applications," *JETC*, vol. 18, no. 2, pp. 1–3, 2021.

[12] D. Ielmini et al., "In-memory computing with resistive switching devices," *Nature Electron.*, vol. 1, no. 6, pp. 333–343, Jun. 2018.

[13] S. Yu, W. Shim, X. Peng, and Y. Luo, "RRAM for compute-in-memory: From inference to training," *IEEE Trans. Circuits Syst. I*, vol. 68, no. 7, pp. 2753–2765, Jul. 2021.

[14] P. Zhou et al., "Energy reduction for STT-RAM using early write termination," in *Proc. Int. Conf. Comput.-Aided Des.*, 2009, pp. 264–268.

[15] X. Sun et al., "PCM-based analog compute-in-memory: Impact of device non-idealities on inference accuracy," in *Proc. ITED*, 2021, pp. 5585–5591.

[16] Y. Zhou et al., "Hybrid-FE-layer FeFET with high linearity and endurance towards on-chip CIM by array demonstration," *IEEE Electron Device Lett.*, vol. 45, no. 2, pp. 276–279, Feb. 2024.

[17] W. Banerjee, "Challenges and applications of emerging nonvolatile memory devices," *Electronics*, vol. 9, no. 6, pp. 1029, Jun. 2020.

[18] A. Singh et al., "Low-power memristor-based computing for edge-AI applications," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Piscataway, NJ, USA: IEEE Press, 2021, pp. 1–5.

[19] C.-J. Jhang, C.-X. Xue, J.-M. Hung, F.-C. Chang, and M.-F. Chang, "Challenges and trends of SRAM-based computing-in-memory for AI edge devices," *IEEE Trans. Circuits Syst. I*, vol. 68, no. 5, pp. 1773–1786, May 2021.

[20] A. E. Arrassi et al., "AFSRAM-CIM: Adder free SRAM-based digital computation-in-memory for BNN," in *Proc. IFIP/IEEE 32nd Int. Conf. Very Large Scale Integration (VLSI-SoC)*, Piscataway, NJ, USA: IEEE Press, 2024, pp. 1–6.

[21] L. Huijbregts et al., "Energy-efficient SNN architecture using 3nm FinFET multiport SRAM-based CIM with online learning," in *Proc. 61st ACM/IEEE Des. Autom. Conf.*, 2024, pp. 1–6.

[22] Q. Dong et al., "15.3 a 351TOPS/W and 372.4 GOPs compute-in-memory SRAM macro in 7nm FinFET CMOS for machine-learning applications," in *Proc. ISSCC*, 2020, pp. 242–244.

[23] R. Kozma et al., *Advances in Neuromorphic Memristor Science and Applications*, Springer Science & Business Media, vol. 4, 2012.

[24] S. Yin et al., "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks," in *Proc. JSSC*, 2020, pp. 1733–1743.

[25] H. Benmeziane et al., “AnalogNAS: A neural network design framework for accurate inference with analog in-memory computing,” in *Proc. Edge*, 2023, pp. 233–244.

[26] 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,” in *Proc. ISSCL*, 2021, pp. 137–140.

[27] H. Fujiwara et al., “A 5-nm 254-TOPS/W 221-TOPS/mm 2 fully-digital computing-in-memory macro supporting wide-range dynamic-voltage-frequency scaling and simultaneous mac and write operations,” in *Proc. ISSCC*, 2022, pp. 1–3.

[28] Y.-D. Chih et al., “16.4 an 89TOPS/W and 16.3 TOPS/mm 2 all-digital SRAM-based full-precision compute-in memory macro in 22nm for machine-learning edge applications,” in *Proc. ISSCC*, 2021, pp. 252–254.

[29] C.-F. Lee et al., “A 12nm 121-TOPS/W 41.6-TOPS/mm2 all digital full precision SRAM-based compute-in-memory with configurable bit-width for AI edge applications,” in *Proc. ISVLSI*, 2022, pp. 24–25.

[30] H. Kim et al., “Colonnade: A reconfigurable SRAM-based digital bit-serial compute-in-memory macro for processing neural networks,” *IEEE J. Solid-State Circuits*, vol. 56, no. 7, pp. 2221–2233, Jul. 2021.

[31] D. Wanq et al., “All-digital full-precision in-SRAM computing with reduction tree for energy-efficient MAC operations,” in *Proc. ICTA*, 2022, pp. 150–151.

[32] L. Deng, “The MNIST database of handwritten digit images for machine learning research,” *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.

[33] Y. LeCun et al., “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[34] A. Krizhevsky et al., “Cifar-10 (Canadian institute for advanced research).” 2010. [Online]. Available: http://www.cs.toronto.edu/kriz/cifar.html

[35] K. He et al., “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[36] T.-Y. Lin et al., “Microsoft coco: Common objects in context,” in *Proc. Comput. Vis. ECCV*, 2014, pp. 740–755.

[37] C. Li et al., “Yolov6: A single-stage object detection framework for industrial applications,” 2022, *arXiv:2209.02976*.

[38] J-s Seo et al., “A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons,” in *Proc. CICC*, 2011, pp. 1–4.

[39] H. Mori et al., “A 4nm 6163-TOPS/W/b 4790-TOPS/mm2/b SRAM based digital-computing-in-memory macro supporting bit-width flexibility and simultaneous MAC and weight update,” in *Proc. ISSCC*, 2023, pp. 132–134.

[40] G. W. Burr et al., “Ohm’s law+ Kirchhoff’s current law= better AI: Neural-network processing done in memory with analog circuits will save energy,” *IEEE Spectr.*, vol. 58, no. 12, pp. 44–49, Dec. 2021.

[41] Y. Biyani et al., “C3CIM: Constant column current memristor-based computation-in-memory micro-architecture,” in *Proc. Des., Automat. & Test Europe Conf. (DATE)*, Piscataway, NJ, USA: IEEE Press, 2025, pp. 1–7.

[42] A. Agrawal et al., “Xcel-RAM: Accelerating binary neural networks in high-throughput SRAM compute arrays,” in *Proc. ISCAS-I*, 2019, pp. 3064–3076.

[43] D. Wang et al., “Always-on, sub-300-nw, event-driven spiking neural network based on spike-driven clock-generation and clock-and power-gating for an ultra-low-power intelligent device,” in *Proc. A-SSCC*, 2020, pp. 1–4.

[44] H. Fujiwara et al., “34.4 a 3nm, 32.5 TOPS/W, 55.0 TOPS/mm 2 and 3.78 mb/mm 2 fully-digital compute-in-memory macro supporting int12× int12 with a parallel-mac architecture and foundry 6t-SRAM bit cell,” in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, vol. 67, Piscataway, NJ, USA: IEEE Press, 2024, pp. 572–574.

[45] S.-Y. Wu et al., “A 7nm CMOS platform technology featuring 4 th generation FinFET transistors with a 0.027 um 2 high density 6-t SRAM cell for mobile SOC applications,” in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, Piscataway, NJ, USA: IEEE Press, 2016, pp. 2–6.

[46] S. Sarangi et al., “DeepScaleTool: A tool for the accurate estimation of technology scaling in the deep-submicron era,” in *Proc. ISCAS*, 2021, pp. 1–5.

[47] A. Santos et al., “RedBit: An end-to-end flexible framework for evaluating the accuracy of quantized CNN,” 2023, *arXiv:2301.06193*.

[48] I. Hubara et al., “Quantized neural networks: Training neural networks with low precision weights and activations,” *J. Mach. Learn. Res.*, vol. 18, no. 187, pp. 1–30, 2017.

[49] H. J. Kelley, “Gradient theory of optimal flight paths,” *ARS J.*, vol. 30, no. 10, pp. 947–954, Oct. 1960.

[50] S. Ruder, “An overview of gradient descent optimization algorithms,” 2016, *arXiv:1609.04747*.