AFSRAM-CIM: Adder Free SRAM-Based Digital Computation-in-Memory for BNN

Arrassi, Asmae El; Yaldagard, Mohammad Amin; Tao, Xingjian; Shahroodi, Taha; Mir, Fouwad; Biyani, Yashvardhan; Gomony, Manil Dev; Gebregiorgis, Anteneh; Joshi, Rajiv; Hamdioui, Said

DOI
10.1109/vlsi-soc62099.2024.10767789

Publication date
2024

Document Version
Final published version

Published in
Proceedings of the 2024 IFIP/IEEE 32nd International Conference on Very Large Scale Integration (VLSI-SoC)

AFSRAM-CIM: **A**dder **F**ree **SRAM**-Based Digital **C**omputation-**in**-**M**emory for BNN

Asmae El arrassi*, Mohammad Amin Yaldagard*, Xingjian Tao†, Taha Shahroodi*, Fouwad Mir*, Yashvardhan Biyani *,Manil Dev Gomony†, Anteneh Gebregiorgis*, Rajiv Joshi ‡, Said Hamdioui*
*Department of Quantum and Computer Engineering, Delft University of Technology, Delft, The Netherlands
Email: {a.elarrassi, m.a.yaldagard, t.shahroodi, f.j.mir, y.biyani, a.b.gebregiorgis, S.Hamdioui} @tudelft.nl
†Eindhoven University of Technology, Eindhoven, The Netherlands
Email: {x.tao1@student, m.gomony@} tue.nl
‡Thomas J. Watson Research Center IBM, New York, USA Email: rvjoshi@us.ibm.com

2024 IFIP/IEEE 32nd International Conference on Very Large Scale Integration (VLSI-SoC) | 979-8-3315-3967-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/VLSI-SOC62099.2024.10767789

***Abstract*—Binary Neural Networks (BNNs) have demonstrated significant advantages in reducing computation and memory costs, all while maintaining acceptable accuracy on various image detection tasks. Thus, BNNs have the potential to support practical cognitive tasks on resource-constrained platforms, such as edge computing devices. To realize this, SRAM-based digital Computation-in-Memory (CIM) has gained growing attention as it overcomes the analog CIM architecture bottlenecks such as limited computing accuracy due to process variation, non-linearity, power and area-hungry Analog-to-Digital Converters (ADCs), etc. However, digital CIM architectures are highly dominated by power-hungry adder-trees, which can nullify the benefits of SRAM-based digital CIM. To address this issue, this paper proposes an adder free SRAM-based digital CIM, AFSRAM-CIM, for BNN acceleration. The proposed CIM architecture utilizes a multi-functional 10-T SRAM cell-based crossbar array and a new energy-efficient approach to perform the popcount operation. Simulation results using the MNIST dataset show that the proposed architecture maintains the state-of-the-art inference accuracy of 99.21% with only 11.86 fJ energy per operation. Moreover, AFSRAM-CIM achieves over 3× energy and ≈17× area savings when compared to the conventional digital CIM approaches.**

***Index Terms*—Computation-in-Memory, SRAM, Fully-digital, BNN, MAC**

I. Introduction

Smart applications have become a crucial part of human daily life. These applications are gaining growing interest for their potential to deliver high accuracy and performance in multiple domains such as computer vision applications [1]. However, due to various architectural challenges and technological limitations, the existing Von-Neumann architecture cannot deliver the computing efficiency required by resource-constrained platforms such as edge devices [2], [3]. Hence, designing energy-efficient architectures and simplified network models is important to deploy AI on edge devices. As a result, emerging CIM architectures are widely studied for their potential to eliminate the excessive time and energy spent on moving massive amounts of data between the memory and processing unit [4], [5]. Moreover, BNNs have shown significant improvements in reducing the complexity of neural network models [6] by aggressively quantizing the parameters to 1-bit precision while maintaining reasonable accuracy [7]. Therefore, there is a clear need to exploit the huge potential of energy-efficient CIM architectures for BNN acceleration.

CIM integrates computing and storage together and provides an efficient implementation of Vector-Matrix Multiplication (VMM), which is the key operation in BNN [8]. Therefore, CIM provides a huge potential for energy-efficient BNN implementation on edge devices [9]. CIM can be realized in an analog domain using emerging/CMOS technologies [10]–[13] or in a digital domain using SRAM cells as its core building block [14], [15]. Digital CIM has an edge over its analog counterpart as it avoids the need for energy and area-hungry ADCs, which represents a vital block for analog CIM [16]. Moreover, CMOS technology maturity and EDA tool support availability make it favorable for fabrication and near future industry-scale adaptation of digital CIM [16]. Several works have presented solutions for SRAM-based digital CIM to perform Multiply-and-Accumulate (MAC) operations [8], [14]–[17]. However, state-of-the-art approaches [8], [14]–[17] require additional circuits to perform multiplication and adder-trees to perform accumulation which dominates the energy and area consumption of the array. As a result, digital SRAM-based CIM computing overheads and challenges need to be minimized to further enhance and optimize CIM architecture for edge applications.

In this work, we propose an energy-efficient adder SRAM-based CIM architecture, AFSRAM-CIM, for BNN accelerators. For this purpose, a multi-functional 10-T SRAM cell is designed to support multiple logic operations such as (XOR, XNOR, OR, etc...). The logic operations are performed within the cell-specific to the targeted application. We propose a new approach to perform accumulation that eliminates the use of the power-hungry adder-trees used in state-of-the-art solutions [9], [14]. The proposed AFSRAM-CIM is evaluated with BNN using the MNIST dataset. Simulation results of post-layout extraction show that the proposed AFSRAM-CIM is highly energy-efficient with a consumption of 11.86 fJ per operation while maintaining state-of-the-art inference accuracy of 99.21%. Thus, AFSRAM-CIM realizes more than 3× and ≈17× energy and area savings over the conventional adder-tree based digital CIM approaches. The main contributions of the paper are summarized as follows:

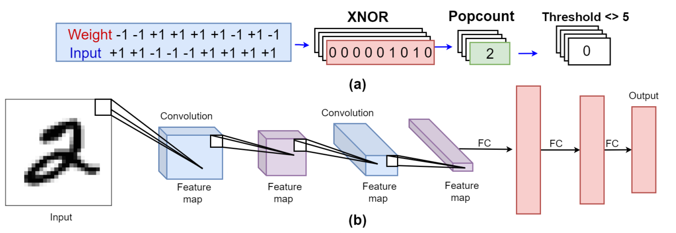


Fig. 1. an illustration of a) BNN inference binary MAC operation and b) Binary Convolutional Neural Network (CNN) architecture.

- A multi-functional SRAM cell design to perform multiple logic operations within the cell;
- An adder-tree free SRAM-based digital CIM architecture to perform binary MAC operation for BNN applications;
- Validation of the proposed architecture shows high energy efficiency of 11.86 fJ per operation with more than $3\times$ energy efficiency improvement compared to state-of-the-art approaches;

The remainder of the paper is organized as follows, Section II presents the basic concepts. Section III presents the proposed architecture. Section IV presents experimental results and discussion. Finally, Section V concludes the paper.

II. Preliminaries

A. Binary Neural Network

BNN is a type of artificial neural network where weights and activation are binarized. BNN reduces memory and computation requirements and offers great potential to improve energy efficiency [18]. Despite the extreme quantization to 1-bit weights and activations, BNN still delivers good inference accuracy for several applications such as object detection and classification [19]. The main operation for BNN is binary MAC operation. The binary MAC operation can be expressed as follows:

$$BinMAC(In, W) = \textbf{popcount}(In\ \textbf{XNOR}\ W) \tag{1}$$

Where In is the input/activation vector and W is the weight vector. Fig. 1(a) illustrates the MAC operation for BNN inference. The signed weights and inputs/activations can be represented by two values "+1" or "-1" [6]. However, previous works [20] explored further simplification of the computation by encoding the values to "1" or "0". The first phase of the MAC operation consists of the XNOR boolean operation between the inputs/activation and the weights. The popcount operation is then performed. The outputs of the popcount represent the accumulation of the output of the XNOR operation. An activation function is then applied to the output of the popcount. The sign activation function can be expressed as follows:

$$sign(x) = \begin{cases} 1 & \text{if } x \geq Threshold \\ 0 & \text{if } x < Threshold \end{cases} \tag{2}$$

Where $Threshold$ can be expressed as follows:

$$Threshold = \frac{Th + N}{2} \tag{3}$$

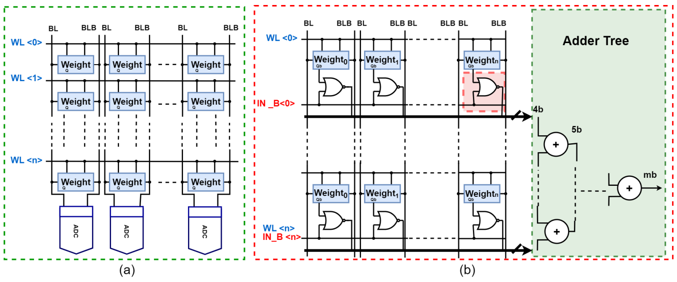


Fig. 2. An illustration of a) analog CIM architecture [24] and b) all digital CIM architecture [14].

Where Th is the activation threshold (ex. Th = 1) and N is the input vector size (ex. N = 9). Fig. 1(b) illustrates an example of a CNN architecture for binary classification. The binary MAC operation reduces the computation costs and memory storage of BNNs.

B. SRAM-based CIM architectures

CIM architectures have the potential to overcome the Von-Neumann challenges by integrating computation and storage in the same physical location [21]. CIM can be realized using different memory technologies such as memristors [22], SRAMs [21], and DRAMs [23]. SRAM-based CIM can be realized in the analog [24] or digital [9] domain. In the analog domain, the first operands (e.g., weight values) are stored in the SRAM cells while the second operands (e.g., activation inputs) are provided through the wordlines (WLs). Then, each column performs the MAC operation by multiplying the input operands and the operands stored in the SRAM cells, the output current is accumulated through the bitlines (BLs) and forms a dot-product according to Kirchhoff's law [3]. The output current through the Bitline (BL) is then fed to an ADC to be converted to its digital value. Fig 2.(a) shows an SRAM-based analog CIM macro with SRAM crossbar array to perform the MAC operation and ADC to convert the MAC output current to digital values. Several works have studied SRAM-based analog CIM. The work in [25] proposes a 10-T SRAM cell to store 1-bit of the weight. The MAC operation is performed in the charge domain, and each column has a dedicated ADC. This approach has addressed the write disturb limitation by decoupling the write path and the read path. However, the power-hungry ADCs used for each column decrease the energy efficiency of the array. Analog-based CIM offers advantages such as high parallelism [9], [10]. However, there are certain design challenges associated with analog-based CIM, including process variation, ADCs overhead, and computing non-linearity, which can limit its scalability [16].

To address these limitations SRAM-based digital CIM eliminates the need for energy and area excessive ADCs [9]. MAC operation can be realized in the digital domain by storing the first operand (eg. weight) in the SRAM-based CIM array. The second operand is provided through the WLs. In each column, the two operands are provided to multipliers to perform multiplication. The multiplication outputs are then fed to an adder-tree to perform the accumulation. Fig. 2(b) illustrates an example of an SRAM-based digital CIM for MAC operation [14]. The work in [9] has proposed an

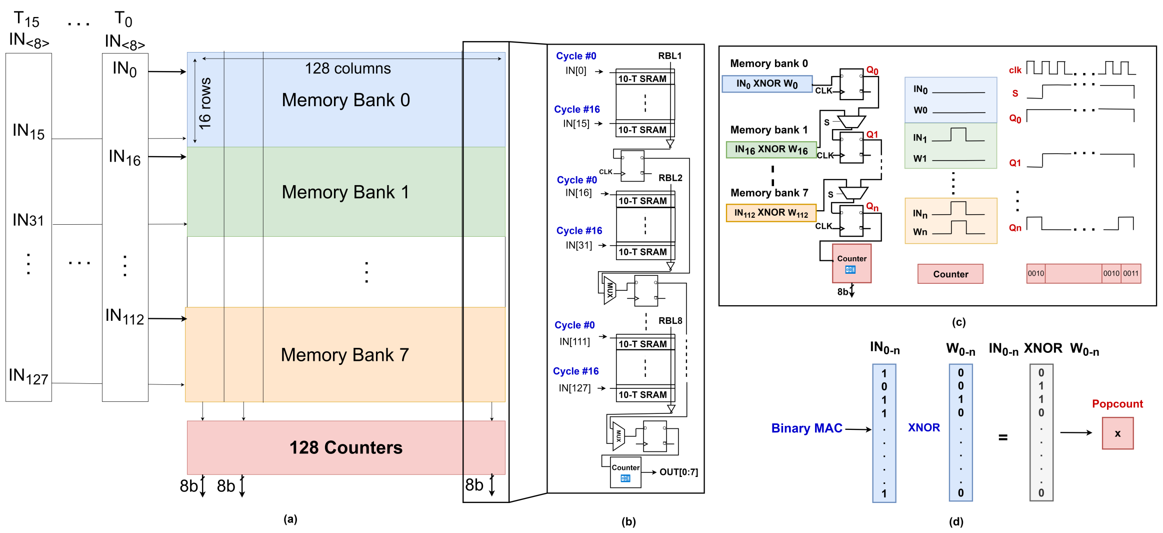
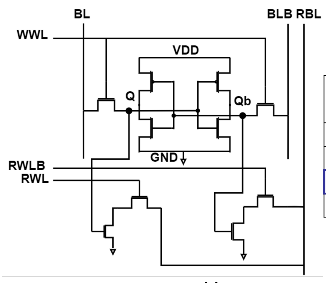


Fig. 3. Illustration of a) the overview of the proposed CIM architecture, b) A column and periphery structure to perform binary MAC operation,c) the Binary MAC operation implementation, and d) A BNN inference binary MAC operation.

SRAM-based CIM architecture for BNN applications. In this approach [9], the weights and activations are stored in the SRAM array. The XNOR operation is performed by activating two rows of the array. For each column, two sense amplifiers (SAs) are added to read the output of the XNOR operation. Furthermore, the outputs of each column are accumulated using an adder-tree. Additionally, the work in [14] has presented a new adder-tree structure to perform accumulation. In this approach, each 4-bit column has a dedicated adder tree which results in a high area consumption. SRAM-based digital CIM offers advantages such as high precision, energy efficiency, and high scalability. However, these approaches suffer from the overhead of the adder-tree units that dominate the area and energy consumption of the array. To address these limitations, we propose a new energy-efficient SRAM-based digital CIM architecture, AFSRAM-CIM, for BNN accelerators.

III. AFSRAM-CIM ARCHITECTURE

The proposed AFSRAM-CIM architecture is realized as follows. First, a multi-functional 10-T SRAM bit cell is designed to perform different logic operations. Then, an adder-tree free CIM macro is designed using the 10-T SRAM cell and popcount logic as its building blocks. In this section, first the design of the 10-T SRAM bit cell is discussed. Then, the proposed CIM macro is presented followed by the discussion of the BNN mapping to the CIM macro.



RWL(IN)	Q	RBL	
		XOR	XNOR
0	0	0	1
0	1	1	0
1	**0**	**1**	0
1	1	0	1

(b)

Fig. 4. Illustration of a) the 10-T SRAM cell and b) the XOR operation performed within the memory cell.

A. *10-T SRAM cell design*

In this work, we propose a 10-T SRAM cell design that supports multiple logic operations within the cell-specific to the targeted application. Fig. 4(a) illustrates a schematic of the proposed 10-T SRAM cell. The write operation is performed by driving the Write Wordline (WWL) and BL to write 1 (or 0). To perform the read operation, The Read Bitline (RBL) is first precharged to V_{DD} and the Read Wordline (RWL) is then activated. The RBL is discharged to 0 when the content of the cell Q = 1 and remains charged when Q = 0. The output of the RBL is then inverted for each column using an inverter as shown in Fig. 3(b). Additionally, the 10-T SRAM cell can support multiple logic operations. The XOR/XNOR operation can be performed within the cell by storing the first operand in the cell and driving the second operand to the RWL/RWLB and its invert to RWLB/RWL. The RBL is precharged to V_{DD} initially when the RWL or RWLB is activated and the content of the cell is Q = 1 or Qb = 1, respectively, the RBL discharges while it remains charged in the other cases. Fig. 4 illustrates how XOR and XNOR operations can be performed within the memory cell.

The SRAM cell supports other logic operations such as NAND and OR operation depending on how the inputs are driven. NAND and OR operations can be performed by disabling RWLB/RWL and driving the IN/INB to RWL/RWLB. The RBL is discharged when (IN = 1 and W = 1) or (IN = 0 and W = 0) to perform NAND or OR operation, respectively. The 10-T SRAM cell has decoupled read-and-write paths which eliminate the read-disturb limitation present in the conventional 6-T SRAM. The cell design is used as the core building block of the CIM macro.

B. *10-T SRAM-based CIM macro*

The proposed CIM macro is composed of 10-T SRAM crossbar array, shared flip-flops, shared multiplexers and counters. The crossbar array consists of multiple subarrays that are arranged in a bank structure. Each bank is organized into

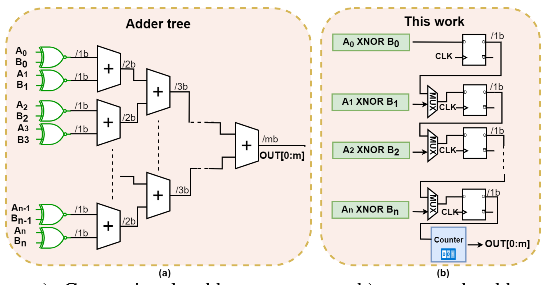


Fig. 5. a) Conventional adder-tree versus b) proposed adder-tree free architecture.

16 rows and 128 columns. For each column in a bank, the rows share the RBLs, BLs, and BLBs while for each row, all the columns share the WWLs, RWLs, and RWLBs as shown in Fig. 3(a). The columns in the first bank are connected to dedicated flip-flops that receive as input the output of their respective RBLs (digitalized with the column inverters). However, the columns in the remaining banks are connected to a multiplexer (MUX) and a flip-flop. The MUXs receive an input signal from the column RBLs and a signal from the flip-flops outputs of the preceding bank. The MUX outputs are then stored in their respective flip-flops. Finally, the flip-flop outputs of the last bank are connected to the counters dedicated for each column as shown in Fig. 3(b).

The proposed AFSRAM-CIM architecture increases the parallelism and reduces the RBLs, BLs, and BLBs charging and discharging delays. The digital counter offers high area efficiency compared to the adder tree approach. Fig. 5 illustrates a comparison between an adder-tree approach and the proposed adder-tree free architecture.

C. BNN mapping

1) BNN implementation of the proposed SRAM-based CIM: The BNN architecture adopted in this work is the LeNet-5 network topology [26]. Fig. 6 illustrates the mapping of the BNN architecture on the proposed CIM design. To map the convolutional layers each filter weight vector is stored in the same column and distributed on different banks while, for the fully connected layer, the weights connected to the same output neuron are stored in one column of the SRAM-based crossbar array. We activate in parallel one row from each bank where the input is provided through the RWLs. The memory bank structure can allow maximum use of the memory storage and parallelism in the array for different layer parameters.

2) Binary MAC operation implementation: To implement binary MAC operation, We perform the XOR operations within the SRAM cell between the weights stored in each cell and the inputs provided through the RWLs. The XOR results are inverted in the RBLs to get the intended XNOR results as shown in equ. 1. We activate n RWLs in parallel, where n is the number of banks in the SRAM-based crossbar array.

To perform the popcount operation, the RBLs of each bank drive the output of the XNOR operation of one SRAM cell at a time. Next, The first bank RBLs are connected to flip-flops that store the signal and provide it to the next bank. The next bank consists of MUXs that receive input signals from

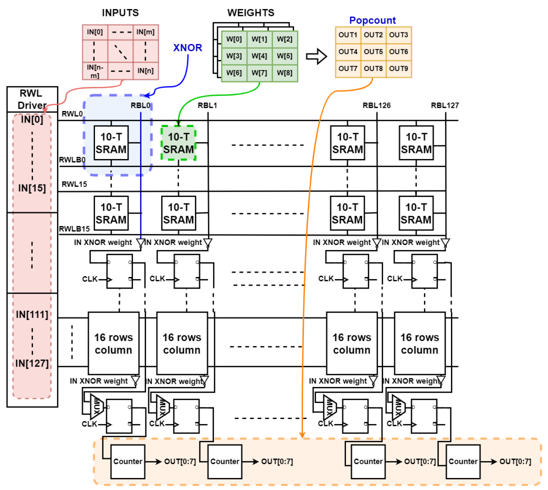


Fig. 6. BNN mapping scheme to the proposed CIM architecture.

the RBLs of the same memory bank and signals from the flip-flops output of the preceding bank. The MUXs are connected to flip-flops. During the first clock cycle, the MUXs provide as output the XNOR outputs provided by the connected RBLs. During the remaining cycle time, These MUXs provide as output the outputs of the preceding flip-flops. The last flip-flops provide inputs to digital counters connected to each column. These flip-flops deliver the RBLs output signal of each bank sequentially. Therefore, in each clock cycle the counter receives the XNOR operation outputs from all the array rows sequentially as shown in Fig. 3(c). The binary MAC operation implementation using digital counters is illustrated in Fig. 3(c)-(d).

To implement large kernel sizes of the convolutional layer, multiple arrays can be used to store the weights, and the resulting partial sum from the digital counters can be accumulated. AFSRAM-CIM architecture offers high accuracy due to the digital logic units used for computation that give an exact and accurate output similar to software implementation. additionally, The low area overhead of the proposed adder-tree free architecture enables accumulation units to be assigned to every column, which offers high parallelism with low overhead.

IV. Experiment Results and Discussion

A. Experiment setup

The simulation setup used in this work is presented in Table. I. The proposed AFSRAM-CIM architecture is simulated with SPICE using TSMC 40 nm CMOS technology. The energy, area, and latency results are extracted using post-layout simulations. It is worth noting that the results were reported in a worst-case scenario with high sparsity (IN = 1, weights 50%). The network is trained offline and then implemented in hardware with VHDL. We performed synthesis using digital design flow for area evaluation.

In this work, We have trained a BNN architecture based on the LeNet-5 network topology [27] as shown in Table. I. The architecture delivers high performance and accuracy in a compact topology [26]. For the MNIST dataset, the size of

TABLE I
SIMULATION SETUP.

Technology	40 nm
Supply voltage (V)	1 V
Temperature	27 °C
SRAM cell	10-T
Unit macro size	16 kb (128x128b)
BNN topology	Lenet-5
Dataset	MNIST
Conv1	(5,5)×20
Conv2	(5,5)×50
Fully-connected layer 1	800×500
Fully-connected layer 2	500×10

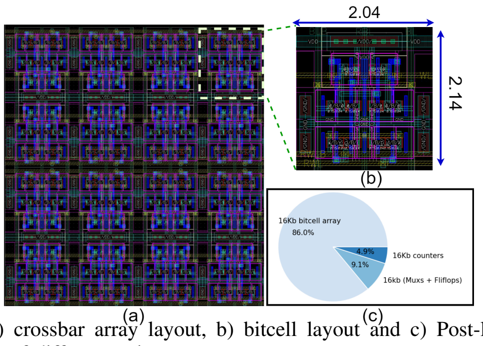


Fig. 7. a) crossbar array layout, b) bitcell layout and c) Post-layout area breakdown of different units.

the input matrix is (28,28). In the first convolutional layer, we apply a filter of dimension (5x5) with 20 channels followed by a pooling layer performing max pooling with a pool size of (2x2). In the second convolutional layer, the applied filter size is (5x5) with 50 channels and a pooling layer with a pool size of (2x2). The network contains two fully connected layers with a size of (800,500) and (500,10). We have trained the network using the BCIM framework [27] using a feedforward model and backpropagation algorithm. The trained network is mapped to implement the inference on the proposed AFSRAM-CIM architecture. the network parameters are presented in Table. I.

B. Energy and area results

To perform energy and area evaluation, a 128×128 SRAM-based crossbar array was simulated in SPICE using the TSMC 40 nm technology. Results show that the proposed AFSRAM-CIM implementation performs MAC operation in an energy-efficient manner with an energy consumption of 11.86 fJ per operation. An operation is defined as one binary multiplication and accumulates operation performed in the array. The recorded energy efficiency and peak throughput are 157 TOPS/W and 496 GOPS, respectively.

Fig. 7 illustrates the array and Bitcell layout and a comparison of the area consumption of different units. The Bitcell occupies an area of 3.407 μm^2. The SRAM 16 kb array occupies 86% area. The counters and the additional units to perform the popcount operation occupy 14% of the total area of the array as shown in Fig. 7(c).

C. Accuracy results

To evaluate the application accuracy, we trained the BNN network with LeNet-5 topology using BCIM framework [27]. The MNIST dataset is used for training. We have mapped the trained weights to the array for inference. The weights are preloaded before performing the MAC operation through the write ports to the SRAM cell. To ensure the maximum use of parallelism and memory storage the weights of the same filter were stored in the same column and the same row of each memory bank. The inputs were provided in parallel through the read ports as described in Section. III-A. The AFSRAM-CIM architecture has achieved the ideal accuracy of 99.28%.

D. State-of-the-art comparison

Table. II shows the performance comparison of the proposed AFSRAM-CIM architecture and state-of-the-art SRAM-based digital CIM neural network implementations. The work in [16] has presented an SRAM-based digital CIM architecture where each bitcell consists of a 6-T SRAM, full adder, two MUXs, and an XNOR gate. The multiplication and partial sum are performed within the array. However, the additional computing units in the bitcell increase the area consumption, resulting in a low-density array. Additionally, the work in [14] has presented an SRAM-based digital CIM architecture. In this work, additional circuitries and adder-trees are used to perform multiplication and accumulation, respectively. However, the high number of adder-trees makes it area inefficient.

In order to compare our solution with the state-of-the-art SRAM-based digital CIM presented in [8], we simulated the CIM architecture using 28 nm technology. The power consumption comaprison of AFSRAM-CIM and the digital CIM [8] is presented in Fig. 8. The reported average power shown in Fig. 8 represents wide range of weight sparsities (percentage of binary weights with the value of ”1”) in order to compare the worst-case and best-case scenarios. Fig. 8 demonstrates that the proposed AFSRAM-CIM architecture consumes $\approx 4\times$ less power compared to the approach in [8]. As illustrated in the figure low-weight sparsity (less binary ”1” weight values) reduces the average power consumption. The 50% weight sparsity shows the highest average power consumption due to the high switching activity of the XNOR results. Moreover, it is worth mentioning that the reported reduction in power consumption is without considering the power reduction benefit of 28 nm implementation over AFSRAM-CIM’s 40 nm. Thus, for similar technology nodes, the power saving of AFSRAM-CIM will be even higher than what is shown in Fig. 8.

For area evaluation, we simulated a 128-bit adder-tree and the proposed adder-tree free approach including all the additional units (digital counter, flip-flops, and MUXs) using the

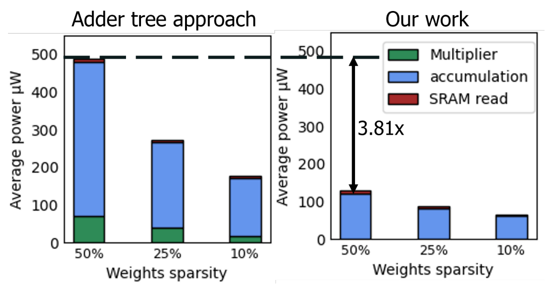


Fig. 8. Power consumption comparison of (right) the proposed architecture and (left) an adder-tree based approach [8] for 32×32 array.

TABLE II
STATE-OF-THE-ART COMPARISON OF THE SRAM-BASED DIGITAL CIM MACROS.

	ICTA'22 [17]	ISSCC'21 [14]	VLSIS'22 [15]	JSSC'21 [16]	ISSCC'22 [8]	**This Work**
Technology (nm)	28	22	12	65	5	40
Supply Voltage (V)	0.9-0.8	0.72	0.72	0.6-0.8	0.5-0.9	1
Input/Output Precision (bit)	4-to-4	(1-8)-to-4	(1-8)-to-(4-16)	1-to-16	4-to-4	1-to-1
GOPS/mm^2	N/a	16000	N/A	6750 (1b)	221000 (4b)	7638 (1b)
Throughput (GOPS)	2632 (4b)	N/A	1343 (4b)	576 (1b)	N/A	496.48 (1b)
Energy efficiency (TOPS/W)	61 (4b)	89 (4b)	121 (4b)	117.3 (1b)	254 (4b)	157.1 (1b)
Energy per operation (pJ/op)	N/A	N/A	N/A	0.017 (1b)	N/A	0.0118
Bitcell density (kb)	64	64	8	64	64	16
Bitcell area ($\mu\mathrm{m}^2$)	N/A	0.379	N/A	10.53	0.075	3.407
Bitcell array area (mm^2)	0.19	0.202	0.0323	0.19	0.0133	0.0659

same 40 nm technology node. The results show that the pop-count units of the proposed AFSRAM-CIM architecture for 128-bit binary MAC operations occupy $\approx 17\times$ less area than a conventional adder-tree structure. Furthermore, the proposed solution occupies 74.91 $\mu\mathrm{m}^2$, while the conventional adder-tree approach occupies 1246 $\mu\mathrm{m}^2$ for 128-bit accumulation. Therefore, AFSRAM-CIM achieves high energy efficiency while occupying a relatively low area with an area efficiency of 7638 GOPS/mm^2.

V. CONCLUSION

In this work, we proposed an energy-efficient BNN implementation using SRAM-based digital CIM. The proposed AFSRAM-CIM architecture minimizes the high energy overhead of the accumulation units by presenting an adder-tree free SRAM-based digital CIM architecture to perform binary MAC operation. Simulation results demonstrated that the proposed AFSRAM-CIM architecture is highly energy-efficient with an energy consumption of 11.86 fJ per operation while maintaining state-of-the-art accuracy of 99.21%. This work achieved over $3\times$ energy and $\approx 17\times$ area savings when compared to the conventional adder-tree approach.

ACKNOWLEDGEMENTS

This work is funded by CONVOLVE (Grant No. 101070374).

REFERENCES

[1] G. Wang and J. Gong, "Facial expression recognition based on improved lenet-5 cnn," in *CCDC*, 2019.
[2] S. Hamdioui *et al.*, "Memristor based computation-in-memory architecture for data-intensive applications," in *DATE*, 2015.
[3] G. W. Burr, A. Sebastian, T. Ando *et al.*, "Ohm's law+ kirchhoff's current law= better ai: Neural-network processing done in memory with analog circuits will save energy," *IEEE Spectrum*, 2021.
[4] H. A. Du Nguyen, J. Yu, L. Xie, M. Taouil, S. Hamdioui, and D. Fey, "Memristive devices for computing: Beyond cmos and beyond von neumann," in *VLSI-SoC*, 2017.
[5] M. Hu, C. E. Graves, C. Li, Y. Li, Ge *et al.*, "Memristor-based analog computation and neural network classification with a dot product engine," *Advanced Materials*, 2018.
[6] M. Courbariaux, I. Hubara, D. Soudry *et al.*, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv*, 2016.
[7] M. Rastegari, V. Ordonez *et al.*, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *ECCV*, 2016.
[8] H. Fujiwara, H. Mori, Zhao *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 *ISSCC*, 2022.
[9] A. Agrawal, A. Jaiswal, D. Roy, B. Han *et al.*, "Xcel-ram: Accelerating binary neural networks in high-throughput sram compute arrays," *ISCAS-I*, 2019.
[10] Q. Dong, M. E. Sinangil, B. Erbagci *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 *ISSCC*, 2020.
[11] R. Kozma, R. E. Pino, and G. E. Pazienza, *Advances in neuromorphic memristor science and applications*, 2012.
[12] S. Yin, Z. Jiang, J.-S. Seo, and M. Seok, "Xnor-sram: In-memory computing sram macro for binary/ternary deep neural networks," *IEEE Journal of Solid-State Circuits*, 2020.
[13] H. Benmeziane *et al.*, "Analognas: A neural network design framework for accurate inference with analog in-memory computing," in *EDGE*, 2023.
[14] Y.-D. Chih, P.-H. Lee, Fujiwara *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 *ISSCC*, 2021.
[15] C.-F. Lee, C.-H. Lu, C.-E. Lee, H. Mori *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 *ISVLSI*, 2022.
[16] H. Kim, T. Yoo, T. T.-H. Kim, and B. Kim, "Colonnade: A reconfigurable sram-based digital bit-serial compute-in-memory macro for processing neural networks," *IEEE Journal of Solid-State Circuits*, 2021.
[17] D. Wanq, Z. Li, C. Chang, W. He *et al.*, "All-digital full-precision in-sram computing with reduction tree for energy-efficient mac operations," in *ICTA*, 2022.
[18] S. Liang, S. Yin, L. Liu, W. Luk, and S. Wei, "Fp-bnn: Binarized neural network on fpga," *Neurocomputing*, 2018.
[19] R. Liu, X. Peng, X. Sun, W.-S. Khwa *et al.*, "Parallelizing sram arrays with customized bit-cell for binary neural networks," in *DAC*, 2018.
[20] X. Sun, X. Peng, P.-Y. Chen *et al.*, "Fully parallel rram synaptic array for implementing binary neural network with (+ 1,- 1) weights and (+ 1, 0) neurons," in *ASP-DAC*, 2018.
[21] Z. Lin, Z. Tong, J. Zhang *et al.*, "A review on sram-based computing in-memory: Circuits, functions, and applications," *Journal of Semiconductors*, 2022.
[22] A. Gebregiorgis, A. Singh, A. Yousefzadeh *et al.*, "Tutorial on memristor-based computing for smart edge applications," *Memories-Materials, Devices, Circuits and Systems*, 2023.
[23] D.-Y. Lim, I.-J. Jung, D.-H. Kim *et al.*, "Computing-in-memory using 1t1c embedded dram cell with micro sense amplifier for enhancing throughput," in *ICCE-Asia*, 2022.
[24] J.-S. Kim, J.-W. Lee *et al.*, "10t sram computing-in-memory macros for binary and multibit mac operation of dnn edge processors," *IEEE Access*, 2021.
[25] A. Biswas and A. P. Chandrakasan, "Conv-sram: An energy-efficient sram with in-memory dot-product computation for low-power convolutional neural networks," *IEEE Journal of Solid-State Circuits*, 2018.
[26] E. Kussul and T. Baidyk, "Improved method of handwritten digit recognition tested on mnist database," *Image and Vision Computing*, 2004.
[27] M. Zahedi, T. Shahroodi, S. Wong, and S. Hamdioui, "Bcim: Efficient implementation of binary neural network based on computation in memory," *arXiv*, 2022.