XNOR-Bitcount Operation Exploiting Computing-In-Memory With STT-MRAMs

Ariana Musello, *Member, IEEE*, Esteban Garzón, *Member, IEEE*, Marco Lanuzza, *Senior Member, IEEE*, Luis Miguel Prócel, *Member, IEEE*, and Ramiro Taco, *Member, IEEE*

***Abstract*—This brief presents an energy-efficient and high-performance XNOR-bitcount architecture exploiting the benefits of computing-in-memory (CiM) and unique properties of spin-transfer torque magnetic RAM (STT-MRAM) based on double-barrier magnetic tunnel junctions (DMTJs). Our work proposes hardware and algorithmic optimizations, benchmarked against a state-of-the-art CiM-based XNOR-bitcount design. Simulation results show that our hardware optimization reduces the storage requirement (–50%) for each XNOR-bitcount operation. The proposed algorithmic optimization improves execution time and energy consumption by about 30% (78%) and 26% (85%), respectively, for single (5 sequential) 9-bit XNOR-bitcount operations. As a case study, our solution is demonstrated for shape analysis using bit-quads.**

***Index Terms*—Computing-in-memory, MAC, XNOR-bitcount, BNN, CNN, spin-transfer torque, STT-MRAM, DMTJ, bit-quad.**

I. Introduction

CONVOLUTIONAL neural networks (CNN) are widely used for image classification, object detection and recognition [1], [2], [3]. However, due to their computational requirements and memory expensive structures, they cannot be easily integrated in small, resource-limited devices, as required in embedded and edge applications [1], [4], [5], [6]. Network compression techniques have given rise to binarized neural networks (BNNs), where weights and activations are represented by a single bit [4]. As a result, the generally expensive multiply-accumulate (MAC) operations between weights and activations can be reduced to low-complexity bitwise XNOR-bitcount operations. This makes BNNs power-efficient, computationally faster, and a memory-saving alternative to conventional CNNs [1]. To further optimize BNNs, we can consider the computing-in-memory (CiM) approach, which exploits the analog functionality and internal structure of the memory, as well as the peripheral circuitry, for data computation [7], [8]. Spin-transfer torque magnetic random-access memories (STT-MRAMs) are leading candidates for CiM in edge devices, due to their non-volatility, considerable write/read speed, compatibility with the CMOS manufacturing process, scalability, and high endurance [9], [10], [11], [12]. Among the STT-MRAM solutions, the double-barrier magnetic tunnel junction (DMTJ) offers improved energy consumption thanks to the reduced switching currents, as compared to the conventional single-barrier MTJ (SMTJ) [13], [14].

This brief investigates hardware-level and algorithmic optimizations for the XNOR-bitcount (XNOR-BC) operation of binary CNNs (BCNNs) using a CiM architecture with DMTJ-based STT-MRAMs. Our approach relies on the XNOR-BC method proposed in [15], which is used here as baseline for comparison.

To summarize, the main contributions of this brief are:

- An optimized energy-efficient and high-performance XNOR-BC algorithm, using a DMTJ-based STT-MRAM hardware solution.
- The proposed optimization of the memory array structure allows reducing (–50%) the storage requirement for computing the XNOR-BC operation.
- The proposed algorithmic optimization enables execution time and energy savings of ∼30% (∼80%) for single (5 sequential) XNOR-BCs.
- Both circuit and functional assessments are carried out by applying the proposed architecture to bit-quad matching in binary images for shape analysis [16].

The remainder of this brief is organized as follows. Section II introduces the DMTJ and the adopted STT-MRAM bitcell configuration. Section III describes the XNOR-BC method, and presents the proposed hardware-level and algorithmic optimizations of this brief. Section IV presents the design and evaluation. An application for shape analysis is shown in Section V. Finally, Section VI concludes our work.

Manuscript received 20 December 2022; accepted 23 January 2023. Date of publication 31 January 2023; date of current version 6 March 2023. This work was supported by the Italian Ministry of University and Research (MUR) under Project PRIN 2020LWPKH7. This brief was recommended by Associate Editor A. L. Zimpeck. *(Corresponding author: Esteban Garzón.)*

Ariana Musello is with the Electronic Design Automation Group, Synopsys Chile Ltda, Santiago de Chile 7630229, Chile (e-mail: amusellov@ieee.org).

Esteban Garzón and Marco Lanuzza are with the Department of Computer Engineering, Modeling, Electronics and Systems, University of Calabria, 87036 Rende, Italy (e-mail: esteban.garzon@unical.it; m.lanuzza@dimes.unical.it).

Luis Miguel Prócel and Ramiro Taco are with the Instituto de Micro y Nanoelectrónica, Universidad San Francisco de Quito, 170901 Quito, Ecuador (e-mail: lprocel@usfq.edu.ec; rtaco@usfq.edu.ec).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2023.3241163.

Digital Object Identifier 10.1109/TCSII.2023.3241163

II. Double-Barrier Magnetic Tunnel Junction (DMTJ) & STT-MRAM Bitcell

The core device of a MRAM architecture is the MTJ [10], whose most common representative is the SMTJ. It consists of two ferromagnetic layers, namely reference and free layer (RL and FL), separated by a thin oxide barrier. The magnetic orientations of the RL and FL can be either parallel (low resistance state or '0') or anti-parallel (high resistance state or '1') [13]. To switch between the MTJ states, the

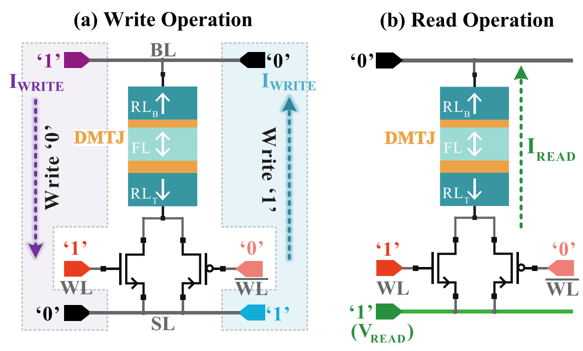

Fig. 1. STT-MRAM bitcell (2T1DMTJ-SC) with write/read operations.

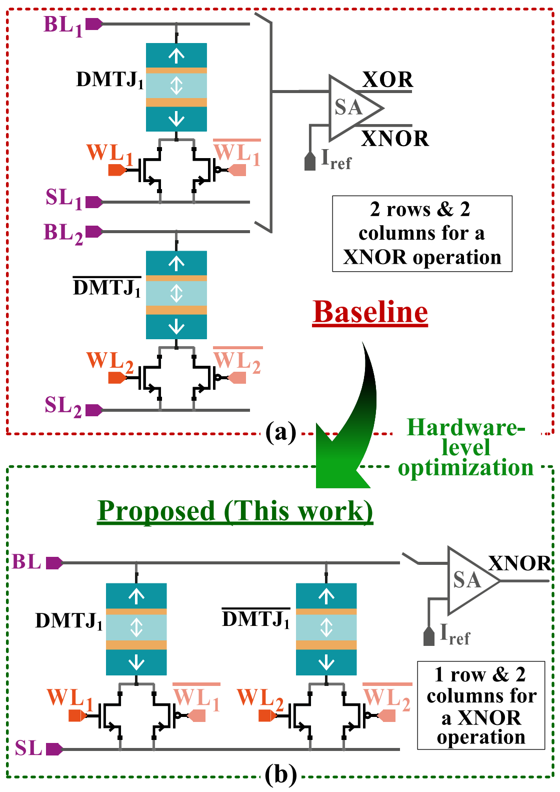
BL_1
DMTJ_1
WL_1
SL_1
BL_2
WL_2
SL_2
SA
XOR
XNOR
I_ref
2 rows & 2 columns for a XNOR operation
Baseline
(a)
Hardware-level optimization
Proposed (This work)
BL
SL
1 row & 2 columns for a XNOR operation
(b)

Fig. 2. Hardware structure proposed in (a) [15], and (b) this brief (optimized) using DMTJ-based STT-MRAM bitcells for a XNOR operation.

STT switching mechanism [13] is exploited, which is triggered by applying a directional current (I_{WRITE}) greater than the critical switching current (I_{c0}). Unfortunately, the conventional SMTJ suffers from considerable large switching current, which adversely impacts on both energy consumption and area occupation. One promising solution concerns the use of the DMTJ [14]. The DMTJ has an additional RL that enhances the total torque acting on the FL, allowing reduced switching current as compared to the SMTJ [13].

Fig. 1 shows the MRAM bitcell configuration used in this brief: two-transistor-one-DMTJ in standard connection, *i.e.,* 2T1DMTJ-SC. According to our previous studies [13], this is the most write-energy efficient DMTJ-based bitcell configuration. Fig. 1 also illustrates the write (a) and read (b) operations. Activation of the access transistors (ATs) through the wordlines (WLs) controls the flow of the current I_{WRITE}; its direction, dependent on the bitline (BL) and sourceline (SL) voltages, defines what state is written into the DMTJ. As for the read operation, we exploit a current sensing scheme: a constant read voltage V_{READ} is applied to the SL while the ATs are turned on. The resulting current I_{READ} in the BL is compared to a reference I_{ref} by a transimpedance sense amplifier (SA) to determine the stored bit [17].

A macrospin-based Verilog-A compact model [14] describes the DMTJ device, whose physical parameters are reported in [13], [14]. The DMTJ features a diameter of 32 nm, high (low) resistance state of 15.3 kΩ (6.9 kΩ), and critical switching current (I_{c0}) of 14.1 µA. For the access transistors, a commercial 65 nm CMOS technology was considered.

III. XNOR-Bitcount Methods Based on CiM

The MAC is the main forward propagation operation of any neural network: weights and activations are multiplied element-wise and then the results are added [4]. In CNNs, each filter ($k \times k$ weight 2D matrix) is convolved with each feature map (input activation 2D matrix). The filter slides throughout the feature map, performing MAC operations between its coefficients and $k \times k$ portions of the feature map [4]. BCNNs are based on binarized weights and activations, *i.e.,* data can be '−1' (represented by logic '0') or '+1' (represented by logic '1') [4]. Therefore, the MAC operation can be simplified to a bitwise XNOR-BC operation [15]: multiple XNORs between weights and activations, followed by a "bitcount" of those results. The XNOR-BC result is the amount of XNOR '1's minus '0's, and once binarized using the sign activation function it is equal to the XNOR results' majority [4].

A. XNOR-BC Method Proposed in [15] Exploiting DMTJ-Based STT-MRAMs

Wang et al.'s XNOR-BC method [15] exploits the write and read operations of voltage-controlled spin-orbit torque (VC-SOT) based MRAM to perform multiple Boolean operations in parallel. We implemented this method using two-terminal STT-DMTJs (instead of three-terminal SOT devices) with the main purpose of reducing circuit complexity. The implementation with STT-MRAM simply requires an adaptation of its write/read operations, utilizing the polarization of the BL/SL and the activation of the access transistors that allow the current to flow. The single-bit XNOR process uses two complementary bitcells, $DMTJ_1$ and $\overline{DMTJ_1}$. This structure, adapted to STT-DMTJs, is shown in Fig. 2(a). XNOR is calculated as follows: XOR $\rightarrow$ $A \oplus W = \overline{A} \cdot W + A \cdot \overline{W}$, where A and W are the binary input activation and filter weight, respectively [15]. The desired complement (XNOR) is obtained through the inverted output of the SA. For a single STT-MRAM based XNOR operation, the basic steps are as follows [15]:

1) *Write weights:* Write filter weights W and $\overline{W}$ to $DMTJ_1$ and $\overline{DMTJ_1}$, respectively, in two cycles (for $DMTJ_1$, 1^{st} write '0's and 2^{nd} write '1's; inversely for $\overline{DMTJ_1}$).
2) *AND operation:* Apply input activations A and $\overline{A}$ to the WLs for $DMTJ_1$ and $\overline{DMTJ_1}$, respectively, and attempt to write '0' to both bitcells. This stores $\overline{A} \cdot W$ in $DMTJ_1$, and $A \cdot \overline{W}$ in $\overline{DMTJ_1}$.
3) *OR operation:* Read both bitcells simultaneously; the accumulated sensing current in the connected BLs represents $\overline{A} \cdot W + A \cdot \overline{W}$, *i.e.,* the XOR result.

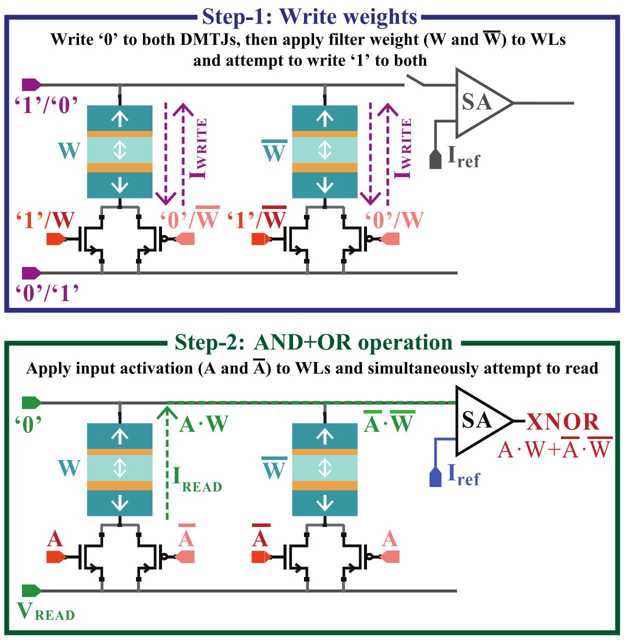
Step-1: Write weights
Write '0' to both DMTJs, then apply filter weight (W and $\overline{W}$) to WLs and attempt to write '1' to both
'1'/'0'
W
$\overline{W}$
I_{WRITE}
SA
I_{ref}
'1'/W
'0'/$\overline{W}$
'1'/$\overline{W}$
'0'/W
'0'/'1'
Step-2: AND+OR operation
Apply input activation (A and $\overline{A}$) to WLs and simultaneously attempt to read
'0'
A·W
$\overline{A}\cdot\overline{W}$
XNOR
A·W+$\overline{A}\cdot\overline{W}$
I_{READ}
A
$\overline{A}$
V_{READ}

Fig. 3. Proposed single XNOR operation exploiting STT-DMTJs and the optimized array structure.

To perform the multi-bit XNOR-BC operation between a $k\times k$ filter and a $k\times k$ feature map portion (N weights and input activations), the corresponding N XOR operations can be computed in parallel and the concept of 'analog majority' (AM) can be used [15]. In **Step-3**, all the bitcells are read simultaneously, so the accumulated current $I_{N\text{-XORs}}$ in the connected BLs represents the addition of the N XOR results. $I_{N\text{-XORs}}$ is compared to a properly set reference current $I_{ref(N)}$ in the SA, such that the XOR results' majority is detected and the SA outputs the binarized XOR-BC/XNOR-BC result (**OR+AM operation**). If the majority of XOR results are '1'('0'), then $I_{N\text{-XORs}} < (>)\ I_{ref(N)}$, so the binarized XOR-BC result given by the SA is '1'('0'), and XNOR-BC is '0'('1').

B. Hardware-Level Optimization

In the design presented in [15], each bitcell of the complementary pair requires a different BL, and the WL is not shared (see Fig. 2(a)). Therefore, a single XNOR operation demands for 2 BLs and 2 WLs of the memory array; N parallel XNOR operations require 2 BLs and $2\times N$ WLs. Moreover, a connection between the separate BLs to a single SA is needed. Our hardware-level optimization consists of locating each pair of complementary bitcells on the same BL, as shown in Fig. 2(b). As for storage density, since the complementary bitcell strips are now on the same BL, N parallel XNOR operations require just one BL and $2\times N$ WLs.

C. Algorithmic Optimization

Step-2 (AND operation) and Step-3 (OR operation) of the single XNOR method in [15] can be merged together into a single read step based on the input activations. We consider the following expression: XNOR $\rightarrow \overline{A\oplus W} = A\cdot W + \overline{A}\cdot\overline{W}$. This methodology allows the direct computation of the XNOR result, instead of the XOR. The basic steps of the optimized method are as follows:

1) *Write weights:* In two cycles, write filter weight W into $DMTJ_1$, and $\overline{W}$ into $\overline{DMTJ_1}$ (1st write '0' to both DMTJs; 2nd apply W and $\overline{W}$ to WLs of $DMTJ_1$ and $\overline{DMTJ_1}$, respectively, and attempt to write '1' to both). See Fig. 3, Step-1.
2) *AND+OR operation:* Apply input activations A and $\overline{A}$ to the WLs of $DMTJ_1$ and $\overline{DMTJ_1}$, respectively, and attempt to read both bitcells simultaneously. See Fig. 3, Step-2.

Only one bitcell is really read because of the complementary activations used. The sensing current - or lack thereof - of $DMTJ_1$ ($\overline{DMTJ_1}$) represents $A\cdot W$ ($\overline{A}\cdot\overline{W}$). The OR operation occurs because of the shared BL, and the resulting current in it represents the XNOR result $A\cdot W + \overline{A}\cdot\overline{W}$.

As for the multi-bit XNOR-BC operation, the same parallel XNOR computations and 'analog majority' concept (BC portion of the operation) are used. When **Step-2** is simultaneously performed, the accumulated current $I_{N\text{-XNORs}}$ in the BL is compared to $I_{ref(N)}$ in the SA, and the binarized XNOR-BC result is obtained (**operation AND+OR+AM**).

The advantages of the optimized XNOR-BC method are as follows. *(a)* Only two write operations are needed instead of three (energy/time savings). *(b)* The weights stored in the bitcells are not overwritten during the process. Therefore, if consecutive XNOR-BC operations are performed using the same filter (weights) with different feature map portions (activations), *i.e.,* cross-correlation, weight-writing is done just once at the beginning, and the following XNOR-BCs consist of AND+OR+AM steps only. *(c)* During the reading step, only half of the bitcells are activated instead of all of them. This means less accumulated read current and reduced energy consumption. *(d)* The direct result of the SA is XNOR-BC, instead of XOR-BC.

IV. STT-MRAM Design and XNOR-BC Simulations

A. STT-MRAM Bitcell Design

The STT-MRAM design features a low supply voltage of 0.8 V. The write current is set to ensure a write-error-rate (WER) of 10^{-7} [13]. For the read operation, read disturbance failures (accidental switching of the DMTJ states [17]), and read decision failures (incorrect reading of the DMTJ states [18]) should be avoided. For this, a V_{READ} that ensures $I_{READ(0/1)} < I_{READ_{max}} = 9.987\,\mu A$ must be used, considering a read disturbance rate RDR $= 10^{-9}$, and a read pulse width t_{READ}=1 ns. Based on this information, and trade-offs between area, delay, energy and robustness, the access transistor widths and cell write/read characteristics are: $w_n = w_p = 310\,nm$, $I_{WRITE}/I_{c0} \approx 2.6$, and $V_{READ} = 95\,mV$. Table I reports bitcell-level nominal results, from which we take a 3 (1) ns long write (read) pulse for the remainder of this brief.

B. Implementation of XNOR-BC Methods and Simulations

Synopsys' Custom Compiler is used to build the hardware and simulate the XNOR-BC methods proposed in [15] and this brief, both implemented with DMTJ-based STT-MRAMs. The memory array with the optimized structure for a M-filter N-bit parallel computation is shown in Fig. 4. The digital control circuitry was behaviorally simulated for both implementations

TABLE I
BITCELL-LEVEL RESULTS: WRITE AND READ OPERATIONS

Write Operation ($I_{WRITE}/I_{c0} \sim 2.6$)			
Transition	I_{WRITE} (µA)	t_{WRITE} (ns)	E_{WRITE} (fJ)
0 to 1	37.22	2.578	76.80
1 to 0	37.19	2.580	76.80
Read Operation (V_{READ} = 95 mV)			
State	I_{READ} (µA)	t_{READ} (ns)	E_{READ} (fJ)
'0'	7.853	1	0.7460
'1'	4.599	1	0.4367

TABLE II
EVALUATION OF MULTI-FILTER XNOR-BC IMPLEMENTATIONS

Filter	Activations	XNORs	$I^{\dagger}_{9\text{-XORs}}$	$I^{\dagger}_{9\text{-XNORs}}$
010100001	010001110	111010000 (4 '1's)	125.09 µA	57.66 µA
101011110	010001110	000101111 (5 '1's)	128.34 µA	54.41 µA
101010101	010001110	000100100 (2 '1's)	118.58 µA	64.17 µA

† $I_{ref(9)}$ is ≈ 126.5 µA and 55.8 µA for $I_{9\text{-XORs}}$ (baseline) and $I_{9\text{-XNORs}}$ (this work), respectively.

TABLE III
EXECUTION TIME AND WORST-CASE ENERGY FOR SINGLE 9-BIT XNOR-BC OPERATION

XNOR-BC Operation	Energy (fJ)		Time (ns)	
	Baseline	This work	Baseline	This work
Write weights (Step-1)	2707.2	2707.2	6	6
AND (Step-2)	968.5	6.7	3	1
OR+AM (Step-3)	10.6		1	
Total	3686.3	2713.9	10	7

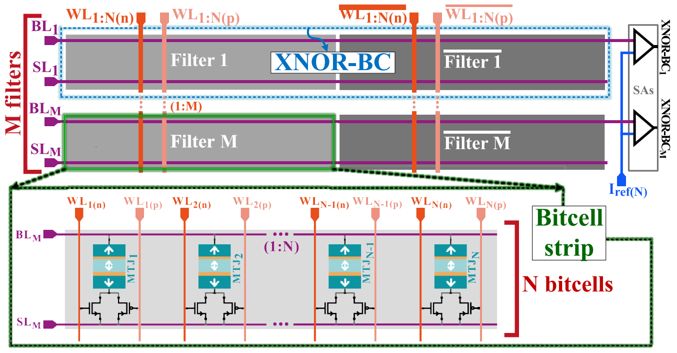

Fig. 4. Memory array for M-filter N-bit parallel XNOR-BC computations.

(*i.e.,* our and the reference one) to verify the whole functionality. Performance results reported hereafter do not take into account the electrical effect of the digital control.

The functional evaluation for multi-filter operations is simulated using the same three 9-bit XNOR-BC operations ($M = 3$ and $N = 9$) reported in [15]: three 3×3 filters and a single 3×3 feature map portion (see Table II, Filter and Activations columns). Table II shows the measured accumulated read current $I_{9\text{-XORs}}$ and $I_{9\text{-XNORs}}$ for the method in [15] and this brief, respectively. Each result is equal to the expected current value according to the number of '1' XNOR results, P, in the XNOR-BC operation: $I_{9\text{-XORs}} = (18 - (9 - P)) \cdot I_{READ(0)} + (9 - P) \cdot I_{READ(1)}$ and $I_{9\text{-XNORs}} = (9 - P) \cdot I_{READ(0)} + P \cdot I_{READ(1)}$. This verifies that the implementations output the correct results, exhibiting high parallelism and throughput. The simulation waveforms for this example are reported in Fig. 5.

For a single 9-bit XNOR-BC operation (one 3×3 filter), Table III compares our proposal and its baseline in terms of execution time and worst-case energy consumption. As the

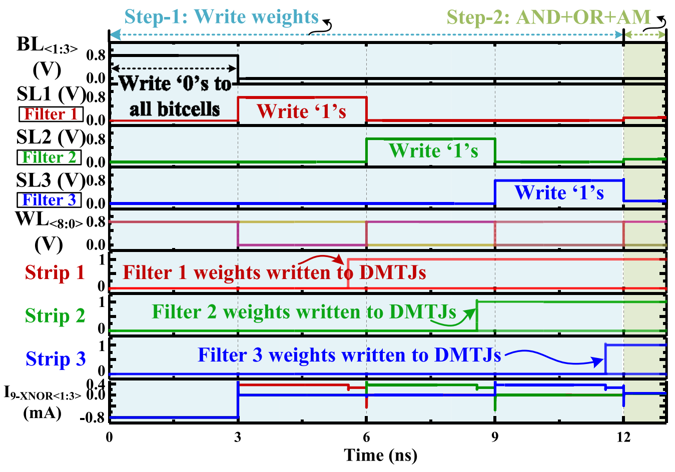

Fig. 5. Simulation waveforms of the 3-filter 9-bit parallel XNOR-BC operations in Table II using this brief's proposed method.

bitcells are not overwritten with '0' during AND operation and only half of them are read, our design exhibits reduced energy consumption (−26.4%) and less execution time (–30%).

Further improvements can be obtained when multiple consecutive XNOR-BCs are performed between the same 3×3 filter (hardware-wise, the same pair of complementary 9-bitcell strips) and multiple 3×3 portions of a feature map. Take 5 consecutive XNOR-BC operations: with the method in [15], each XNOR-BC costs 3686.3 fJ and 10 ns, with a total of 18431.5 fJ and 50 ns for the five; with the proposed method, the first XNOR-BC costs 2713.9 fJ and 7 ns, but the following four only cost 6.7 fJ and 1 ns, with a total of 2740.7 fJ and 11 ns. This implies a reduction of about 85.1% to the total energy consumption, and of 78% in terms of execution time.

V. APPLICATION: BIT-QUADS FOR SHAPE ANALYSIS

Bit-quads are 2×2 binary patterns that can be used to compute the area, perimeter, and Euler number of a binary image by counting the number of matches ($n\{Q_x\}$) between them and portions of such an image [16]. For example, the Euler number under four-connectivity can be calculated with $E = 0.25[n\{Q_1\} - n\{Q3\} + 2 \cdot n\{Q_D\}]$.

Our proposed architecture and method can be used to find the bit-quad patterns in an image by performing parallel cross-correlations between each bit-quad (16 in total) and the image. Thus, multiple consecutive 4-bit XNOR-BC operations are performed between each bit-quad - the filter whose weights are stored in the DMTJs - and each 2×2 section of the image - whose pixels are the input activations of the AND+OR+AM step. We determine a match through the BL sensed current: if $I_{4\text{-XNORs}} = 4 \times I_{READ(1)} = 18.40\,\mu A < I_{ref(4)}$, it means the 4 XNOR results of the XNOR-BC operation are 1, so the 4 pixels of the bit-quad and the image section are equal. Thus, the binarized XNOR-BC result of the SA is 1. For this, the array in Fig. 4 is used with $M = 16$ and $N = 4$.

This is illustrated by the example reported in Fig. 6, where the 16 bit-quads (divided in categories) are shown to the left, and some of the 25 2×2 portions of the 6×6 binary image inset are shown at the top. Fig. 6 shows the $I_{4\text{-XNORs}}$ waveforms for the AND+OR+AM steps, and highlights them when equal to 18.40 µA, which corresponds to the coincidences between bit-quad and image portion (see match pattern in Fig. 6). This

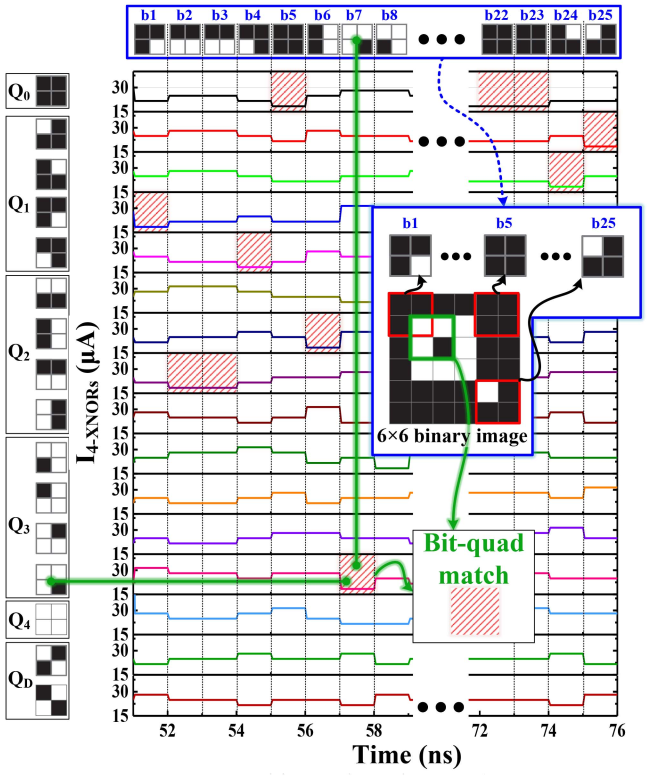

Fig. 6. $I_{4\text{-XNORs}}$ waveforms and bit-quad matching results.

proves that the proposed approach works correctly for bit-quad matching. With the matches counted and stored, the image's Euler number can be calculated as $E = 0.25[6-4+2(1)] = 1$.

For the above example, execution time was 76 ns: 51 ns for Step-1 (write weights) and 25 ns for consecutive Steps-2 (AND+OR+AM). The worst-case energy consumption for a single bit-quad is about 1.28 pJ. If the method in [15] was used, total execution time would be 25×55 ns $= 1375$ ns, with worst-case bit-quad energy consumption of 40.96 pJ. This would be $\sim 18\times$ and $\sim 32\times$, respectively, as compared to our approach.

VI. Conclusion

In this brief, we propose a XNOR-BC architecture using computing-in-memory along with DMTJ-based STT-MRAMs. The XNOR-BC was optimized from a state-of-the-art approach, which was used as baseline to evaluate the cost at hardware- and algorithm-level. Such a benchmark analysis has been carried out by performing circuit-level simulations with Synopsys' Custom Compiler tool. The hardware-level optimization of the memory array structure halved the storage requirements needed to compute the XNOR-BC operation while simplifying BL connections. For a single 9-bit XNOR-BC operation (one 3×3 filter), our optimized method allowed 30% less execution time and energy savings of 26.4%. Considering 5 sequential 9-bit XNOR-BC operations with a single filter, our method achieved a 78% decrease in execution time and a 85.1% reduction in energy consumption. The optimized architecture was also demonstrated to be a time- and energy-efficient tool in the field of shape analysis, particularly for the process of bit-quad matching in binary images for later topological analysis. Future research efforts will involve the demonstration of a complete neural network using the proposed architecture.

References

[1] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, "Binary neural networks: A survey," *Pattern Recognit.*, vol. 105, Sep. 2020, Art. no. 107281.

[2] K. Kang et al., "T-CNN: Tubelets with convolutional neural networks for object detection from videos," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 10, pp. 2896–2907, Oct. 2018.

[3] Z. Chen, Y. Ma, and Z. Wang, "Hybrid stochastic-binary computing for low-latency and high-precision inference of CNNs," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 7, pp. 2707–2720, Jul. 2022.

[4] X. Si et al., "A dual-split 6T SRAM-based computing-in-memory unit-macro with fully parallel product-sum operation for binarized DNN edge processors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 11, pp. 4172–4185, Nov. 2019.

[5] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 525–542.

[6] M. Yayla et al., "FeFET-based binarized neural networks under temperature-dependent bit errors," *IEEE Trans. Comput.*, vol. 71, no. 7, pp. 1681–1695, Jul. 2022.

[7] G. Santoro, G. Turvani, and M. Graziano, "New logic-in-memory paradigms: An architectural and technological perspective," *Micromachines*, vol. 10, no. 6, p. 368, 2019.

[8] E. Garzón, A. Teman, M. Lanuzza, and L. Yavits, "AIDA: Associative in-memory deep learning accelerator," *IEEE Micro*, vol. 42, no. 6, pp. 67–75, Nov./Dec. 2022.

[9] H. Cai et al., "Proposal of analog in-memory computing with magnified tunnel magnetoresistance ratio and universal STT-MRAM cell," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 4, pp. 1519–1531, Apr. 2022.

[10] P. Barla, V. K. Joshi, and S. Bhat, "Spintronic devices: A promising alternative to CMOS devices," *J. Comput. Electron.*, vol. 20, no. 2, pp. 805–837, 2021.

[11] K. Asifuzzaman, R. S. Verdejo, and P. Radojković, "Enabling a reliable STT-MRAM main memory simulation," in *Proc. Int. Symp. Memory Syst.*, 2017, pp. 283–292.

[12] F. Cutugno, E. Garzón, R. De Rose, G. Finocchio, M. Lanuzza, and M. Carpentieri, "Field-free magnetic tunnel junction for logic operations based on voltage-controlled magnetic anisotropy," *IEEE Magn. Lett.*, vol. 12, pp. 1–4, 2021.

[13] E. Garzón et al., "Assessment of STT-MRAMs based on double-barrier MTJs for cache applications by means of a device-to-system level simulation framework," *Integration*, vol. 71, pp. 56–69, Mar. 2020.

[14] R. De Rose, M. d'Aquino, G. Finocchio, F. Crupi, M. Carpentieri, and M. Lanuzza, "Compact modeling of perpendicular STT-MTJs with double reference layers," *IEEE Trans. Nanotechnol.*, vol. 18, pp. 1063–1070, 2019.

[15] H. Wang, W. Kang, B. Pan, H. Zhang, E. Deng, and W. Zhao, "Spintronic computing-in-memory architecture based on voltage-controlled spin–orbit torque devices for binary neural networks," *IEEE Trans. Electron Devices*, vol. 68, no. 10, pp. 4944–4950, Oct. 2021.

[16] W. Pratt, *Digital Image Processing*, 4th ed. New York, NY, USA: JWS, 2007.

[17] A. Khvalkovskiy et al., "Basic principles of STT-MRAM cell operation in memory arrays," *J. Phys. D Appl. Phys.*, vol. 46, no. 7, 2013, Art. no. 074001.

[18] R. Bishnoi, M. Ebrahimi, F. Oboril, and M. B. Tahoori, "Read disturb fault detection in STT-MRAM," in *Proc. IEEE Int. Test Conf.*, 2014, pp. 1–7.

Open Access funding provided by 'Università della Calabria' within the CRUI CARE Agreement