UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Computing-in-Memory XNOR-Bitcount Operation of Binary Convolutional Neural Networks based on Spin-Transfer Torque MRAMs

Ariana Musello Vásconez

Ingeniería en Electrónica y Automatización

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniera en Electrónica

Quito, 13 de diciembre de 2021

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Computing-in-Memory XNOR-Bitcount Operation of Binary Convolutional Neural Networks based on Spin-Transfer Torque MRAMs

Ariana Musello Vásconez

Ramiro Taco, PhD **Luis Miguel Prócel, PhD**

Quito, 13 de diciembre de 2021

© DERECHOS DE AUTOR

Por medio del presente documento certifico que he leído todas las Políticas y Manuales de la Universidad San Francisco de Quito USFQ, incluyendo la Política de Propiedad Intelectual USFQ, y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo quedan sujetos a lo dispuesto en esas Políticas.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo en el repositorio virtual, de conformidad a lo dispuesto en la Ley Orgánica de Educación Superior del Ecuador.

Nombres y apellidos:	Ariana Musello Vásconez
Código:	00200447
Cédula de identidad:	1721742151
Lugar y fecha:	Quito, 13 de diciembre de 2021

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

Las redes neuronales binarias (BNN), en las que los pesos y activaciones son representadas con un solo bit, son una alternativa eficiente en cuanto a almacenamiento y computación que permite implementar redes neuronales convolucionales en dispositivos con recursos computacionales limitados. En BNNs, la operación MAC es reemplazada por una operación de XNOR-*bitcount*. El uso de arquitecturas de computación en memoria (CIM) permite un ahorro incluso mayor en energía y tiempo para la implementación de BNNs. En este contexto, el uso de STT-MRAMs ofrece los beneficios de no-volatilidad, gran velocidad, consumo bajo de energía, escalabilidad, entre otros. Este trabajo propone optimizaciones a nivel de hardware y algoritmo para la operación de XNOR-*bitcount* basada en CIM presentada por Wang *et al.* en [1], usando STT-MRAMs. La optimización de hardware reduce a la mitad el espacio de almacenamiento requerido para cada operación XNOR-*bitcount*. Asumiendo un filtro de 9 bits (9 pesos), la optimización algorítmica permite 30% menos tiempo de ejecución y 26.1% menos consumo energético. Para 5 operaciones XNOR-*bitcount* secuenciales usando el mismo filtro, la optimización algorítmica ofrece una reducción de 78% en tiempo de ejecución y una reducción de 85.1% en consumo energético.

Palabras clave: Redes neuronales binarias (BNN), redes neuronales convolucionales (CNN), computación en memoria (CIM), transferencia por torque de spin (STT), STT-MRAM, junturas túnel magnéticas de doble barrera (DMTJ), operación MAC, XNOR-*bitcount*.

ABSTRACT

Binary neural networks (BNN), in which weights and activations are represented with one single bit, are a storage and computationally efficient alternative that allows convolutional neural networks to be implemented in devices with limited computational resources. In BNNs the MAC operation is replaced with a bitwise XNOR-bitcount operation. The use of computation-in-memory (CIM) architectures allows further energy and time savings for BNN implementation. In this context, the use of STT-MRAMs provides the benefits of non-volatility, high speed, low-power consumption, scalability, among others. This work proposes hardware- and algorithmic-optimizations to the CIM-based XNOR-bitcount method presented by Wang *et al.* in [1], using STT-MRAMs. The hardware optimization reduces in half the storage space required for each XNOR-bitcount operation. Assuming a 9-bit filter (9 weights), the algorithmic optimization provides 30% less execution time and 26.1% less energy consumption in a single XNOR-bitcount operation. For 5 sequential XNOR-bitcount operations using the same filter, the algorithmic optimization allows a 78% reduction in execution time and 85.1% reduction in energy consumption.

Key words: Binary neural networks (BNN), convolutional neural networks (CNN), compute-in-memory (CIM), spin-transfer torque (STT), STT-MRAM, double-barrier magnetic tunnel junction (DMTJ), MAC operation, XNOR-bitcount.

TABLE OF CONTENTS

I. Introduction 10

II. Spintronic devices and STT-MRAMs 11

A. MTJ-based MRAM bitcells 11

B. STT-MRAM bitcell design 13

C. Equivalency of operations between VC-SOT MRAMs and STT-MRAMs 14

Writing operation 14

Reading operation 15

III. Binary convolutional neural networks and XNOR-bitcount 16

IV. XNOR-bitcount methods based on Computing - in - Memory 18

A. XNOR-bitcount method proposed by Wang *et al.* 18

Single-bit XNOR operation 19

Multi-bit XNOR-bitcount operation 21

B. Hardware-level optimization 22

C. Wang *et al.*'s XNOR-bitcount method with STT-MRAMs and optimized hardware. 24

D. Optimized XNOR-bitcount method with STT-MRAMs and optimized hardware 25

Single-bit XNOR-bitcount operation using algorithmic optimization 25

Multi-bit XNOR-bitcount operation using algorithmic optimization 28

V. Design of STT-MRAM and simulation results of XNOR-bitcount methods 29

A. STT-MRAM bitcell design 29

VDD selection and access transistor sizing 29

B. Implementation of XNOR-bitcount methods and simulation results 33

Logic verification of methods 33

Energy consumption and execution time comparison between XNOR-bitcount methods .. 37

VI. Conclusions 39

Bibliography 40

TABLE INDEX

Table 1: Characteristics of DMTJ model..14

Table 2: Equivalency of writing operation between VC-SOT MTJs and STT MTJs 15

Table 3: Stored states read from $MTJ1$ and $MTJ1$ during last step (simultaneous reading) and total read current according to activations A and weights W using Wang et al.'s XNOR-BC method..20

Table 4: Stored states read from $MTJ1$ and $MTJ1$ during last step (simultaneous reading) and total read current according to activations A and weights W using this work's XNOR-BC method..27

Table 5: Access transistor widths 32

Table 6: Write operation current, delay and energy results at bitcell level 32

Table 7: Read operation current, delay and energy results at bitcell level 32

Table 8: Accumulated currents using Wang's method corresponding to each XNOR-bitcount result..34

Table 9: Accumulated currents using this work's method corresponding to each XNOR-bitcount result..34

Table 10: Weights and activations to evaluate XNOR-bitcount methods...............................34

Table 11: Accumulated currents measured for each XNOR-BC operation.............................36

Table 12: Worst case energies for each XNOR-bitcount step and total energy consumption.37

Table 13: Execution times for each XNOR-bitcount step and total execution time................37

Table 14: Energy and execution time results for 5 consecutive XNOR-bitcount operations..38

FIGURE INDEX

Figure 1: (a) VC-SOT-based bitcell and (b) STT-based bitcell 13
Figure 2: Reading operation for a single STT-based bitcell 15
Figure 3: Example of multiply-and-accumulate (MAC) operation in CNNs 16
Figure 4: MAC operation equivalency to XNOR-bitcount operation when using binary weights and activations 17
Figure 5: Wang et al.'s complementary STT-bitcell structure. (a) Two-cycle process to write weights for single-bit XOR operation, (b) Simultaneous read operation for single-bit OR computation step 20
Figure 6: Wang et al.'s complementary bitcell-strips structure using STT-MTJs for multibit computation with N=9 21
Figure 7: Hardware-level optimization for a pair of complementary bitcells 22
Figure 8: This work's optimized complementary STT-bitcell structure: (a) Two-cycle process to write weights for single-bit XOR operation, (b) Simultaneous read operation for single-bit OR step 23
Figure 9: This work's optimized complementary bitcell strips structure using STT-MTJs for multibit computation with N=9 24
Figure 10: Wang et al.'s single-bit XOR/XNOR operation using STT-MTJs and optimized array structure 25
Figure 11: This work's single-bit XOR/XNOR operation using STT-MTJs and optimized array structure 26
Figure 12: Monte Carlo results for read operation considering process variations 32
Figure 13: Memory array section for 9-bit multi-kernel parallel XNOR-BC computations ... 35
Figure 14: Simulation waveforms using Wang's method of the three parallel XNOR-bitcount operations in Table 10 35
Figure 15: Simulation waveforms using this work's method of the three parallel XNOR-bitcount operations in Table 10 36

I. INTRODUCTION

Convolutional neural networks (CNN) are ideal in the fields of image classification, object detection and object recognition [2] [3]. However, the deep structures that allow them to provide state-of-the-art results also makes them incredibly computationally and memory expensive [3], so they require high-performance hardware like GPUs [2]. This makes deep CNNs unusable in real-world applications and Internet-of-Things (IoT) applications, as they cannot be integrated in smaller devices with limited storage and computational resources such as cell-phones, embedded devices and smart portable devices [4] [2].

To address this, several network compression techniques have been proposed, and one of them is the quantization of the network's weights and activations: representing their floating-point values using very low precision [2]. The extreme case is *binarization*, which gives rise to *binary* neural networks (BNNs). In BNNs, the weights and activations are represented with one bit: data can be '-1' ('0') or '+1' ('1') [5]. This implies that the costly multiply-and-accumulate (MAC) operations between neural networks' activations and weights can be replaced with bitwise XNOR-bitcount operations [2]. This makes BNNs power-efficient, computationally faster, and memory-saving [2] [4].

To further optimize BNNs, a compute-in-memory (CIM) architecture can be considered. The gap in performance between memories and computing units, called the 'von-Neumann bottleneck', causes the data movement between memory and the computing unit to be even more energy- and time-expensive than the computation itself [6] [7]. The *computation-in-memory* (CIM) approach allows data computation to be performed by exploiting the memory's internal structure and analog functionality, as well as the peripheral circuitry (sense amplifiers, decoders, etc.) [7]; the memory array is not modified, but the peripheral circuitry can be adapted for this purpose.

In this matter, spin-transfer torque magnetic random-access memories (STT-MRAMs) are leading candidates for non-volatile on-chip memory, Internet of Things (IoT) applications, and logic-in-memory (LIM) applications [8] [9]. This is because they can offer high write/read speed, small area footprint, non-volatility, low power consumption, easy integration with CMOS technology, scalability, and high endurance [10] [11].

In the above context, this work investigates hardware- and algorithmic-optimizations to the XNOR-bitcount (XNOR-BC) operation of binary convolutional neural networks (BCNNs) using a CIM approach and STT-MRAMs. This work focuses particularly on the XNOR-BC method proposed in [1], which uses VC-SOT MRAMs for CIM BCNN implementation.

II. SPINTRONIC DEVICES AND STT-MRAMS

A. MTJ-based MRAM bitcells

The elementary device of spintronic based MRAM is the magnetic tunnel junction (MTJ) [8]. A single-barrier MTJ is composed by two ferromagnetic (FM) layers with a thin oxide layer between them. While one FM layer – the reference layer (RL) – has a fixed magnetic orientation, the magnetization of the other FM layer – the free layer (FL) – can be either parallel (low resistance state) or anti-parallel (high resistance state) to the RL. The low resistance state denotes data ‘0’ and the high resistance state denotes data ‘1’ [12] [10].

Among the switching techniques for the MTJ state, there is spin-transfer torque (STT) [10] and voltage-controlled spin-orbit torque (VC-SOT) [12]. As mentioned before, Wang *et al.* [1] propose a BCNN design based on multibit VC-SOT MRAM. In this work, I replicate their method – and further optimize it – using STT-MRAM.

A VC-SOT MTJ device has three terminals, allowing a separation between writing and reading paths [13]. The MTJ's RL is connected to a voltage source and the FL is in contact with an antiferromagnetic layer or heavy metal [12]. Figure 1.a shows the structure for the MRAM bitcell used in [1] based on a VC-SOT MTJ. When a current (I_{SOT}) flows through the heavy metal, spins accumulate and torque switching is generated on the MTJ's FL, changing its magnetization according to the direction of the current [13]. A bias voltage V_b, related to the VCMA effect, applied to the RL can modulate the energy barrier of the MTJ and, therefore, modulate the SOT threshold current [1]. This means that with a positive V_b – that reaches the MTJ by activating the access transistor through the wordline (WL) –, the SOT current can switch the MTJ state, while with $V_b = 0$, the current has no effect on said state [1].

On the other hand, a STT-MTJ uses the same path to write and read [13]. STT switching is based on applying a current I_{write} through the MTJ greater than its critical switching current I_{c0}, which is the necessary current to write a '0' state [10]. If the current flows from the MTJ's FL to the RL, it is configured into parallel state ('0' is written), and if the current flows in the opposite direction (RL to FL), it is configured into anti-parallel state ('1' is written). To generate said current in the appropriate direction, either the bitline (BL) or sourceline (SL) of the bitcell is charged to VDD, while the other line is grounded [10]. Figure 1.b shows the structure for the MRAM bitcell based on a STT-MTJ that will be used in this work. It uses a double-barrier MTJ (DMTJ) and is in 2T1MTJ-SC configuration; this will be further explained in Section II. B.

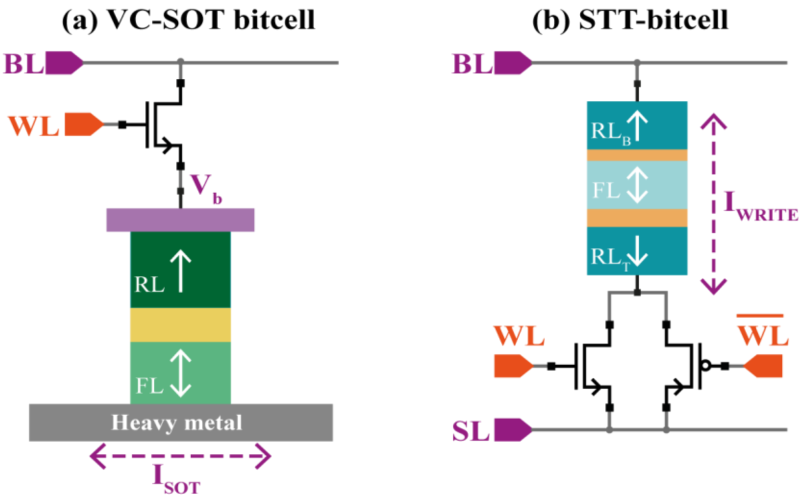

Figure 1: (a) VC-SOT-based bitcell and (b) STT-based bitcell

B. STT-MRAM bitcell design

The critical switching current I_{c0} of a single-barrier MTJ (SMTJ) is generally quite high, so the use of double-barrier MTJs (DMTJs) is a good alternative that allows a reduced I_{c0} due to an additional RL that increases the torque generated over the FL [14]. The reduced switching current allows DMTJ-based STT-MRAM to have reduced write access times, lower write and read energy consumption, smaller area, and reduced leakage power, albeit higher read access times [15]. Moreover, the critical currents for writing ‘0’ and ‘1’ in a DMTJ are equal. These advantages can be appreciated when the bitcell is in 2T1MTJ-SC configuration: 2 access transistors and 1 MTJ in standard connection (SC), where complementary *n*-MOS and *p*-MOS transistors are used, and they are connected to the top reference layer (RL_T) of the DMTJ [15].

The DMTJ device used in this work is described by macrospin-based Verilog-A compact models [16]. For the access transistors, TSMC’s commercial *65-nm* technology node was used. The Synopsys Custom-Compiler tool was used for simulations. Table 1 shows the DMTJ’s characteristics.

Parameter	Description	Value
d	MTJ diameter	$32\ nm$
t_{FL}	Free layer thickness	$1.2\ nm$
$t_{OX,T}$	Top barrier thickness	$0.85\ nm$
$t_{OX,B}$	Bottom barrier thickness	$0.4\ nm$
RA	Resistance-area product	$\sim 12\ \Omega\ \mu m^2$
R_H	High resistance state	$15.3\ K\Omega$
R_L	Low resistance state	$6.9\ K\Omega$
I_c	Critical switching current	$14.1\ \mu A$
Δ	Thermal stability factor	$\sim 71\ k_B T$

Table 1: Characteristics of DMTJ model

C. Equivalency of operations between VC-SOT MRAMs and STT-MRAMs

The equivalency between writing and reading operations using one type of MTJ and the other must be understood in order to implement Wang *et al.*’s XNOR-bitcount method using STT-MRAMs.

Writing operation

The key concept is that both in VC-SOT and STT bitcells, the direction of the current determines *what* state is written, and the access transistor gate voltage determines whether that state *is written or not*. This is because in VC-SOT MTJs, the control voltage V_b is applied to the MTJ by activating the access transistor by using a positive gate voltage V_g and allowing the voltage in the BL to pass. In STT-MTJs, the activation of the access transistors using a positive V_g (for the n-MOS transistor, the p-MOS transistor is always assumed to receive the complementary signal) allows I_{write} to cross through or not cross through the MTJ.

Table 2 shows the four possible combinations of access transistor activations and current directions for each device type, and the corresponding MTJ state that is written.

VC-SOT bitcell		STT-bitcell		MTJ state
I_{SOT}	V_g	I_{write}	V_g	
⟶	0	↓	0	*Previous state*
⟶	+	↓	+	'0'
⟵	0	↑	0	*Previous state*
⟵	+	↑	+	'1'

Table 2: Equivalency of writing operation between VC-SOT MTJs and STT MTJs

Reading operation

The reading operation in the VC-SOT MRAM is based on activating the access transistor and measuring the current in the BL [1]. For the reading operation in the STT-MRAM, the same concept is used. This work uses a current sensing scheme in which a constant read voltage V_{read} is applied to the SL while the WL is activated (activated access transistors); the resulting current in the BL is compared to a reference I_{ref} by a SA to determine the stored bit [17]. Figure 2 shows this process.

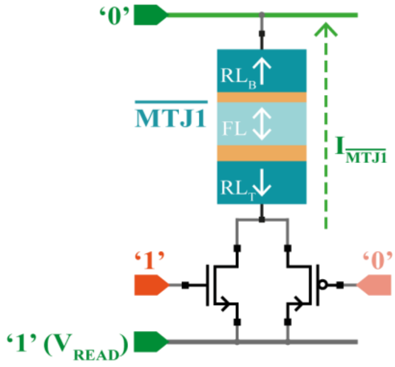

Figure 2: Reading operation for a single STT-based bitcell

III. BINARY CONVOLUTIONAL NEURAL NETWORKS AND XNOR-BITCOUNT

As in any neural network, the main operation in the forward propagation process is 'multiply-and-accumulate' (MAC): the weights and input activations are multiplied element-wise and the results are added or accumulated – basically, the dot product [4] [5]. This result is the activation of a neuron in the next layer. Regarding convolutional neural networks (CNNs), the weights and input activations are represented with 2-D matrixes. This makes them useful for image recognition and object detection tasks [2]. The 2-D weight matrixes are called *filters* or *kernels*, while the 2-D input activations are called *feature maps*. Each filter is *convolved* with each feature map, which consists on sliding the kernel through the complete feature map and repeating dot-product operations between the filter and the corresponding part of the input feature map, thus generating a new feature map for the next layer of the neural network [4]. This convolution consists of multiple MAC operations, each one between the $k \times k$ kernel and a $k \times k$ portion of the input feature map. Figure 3 shows a simple example of a single MAC operation in a CNN.

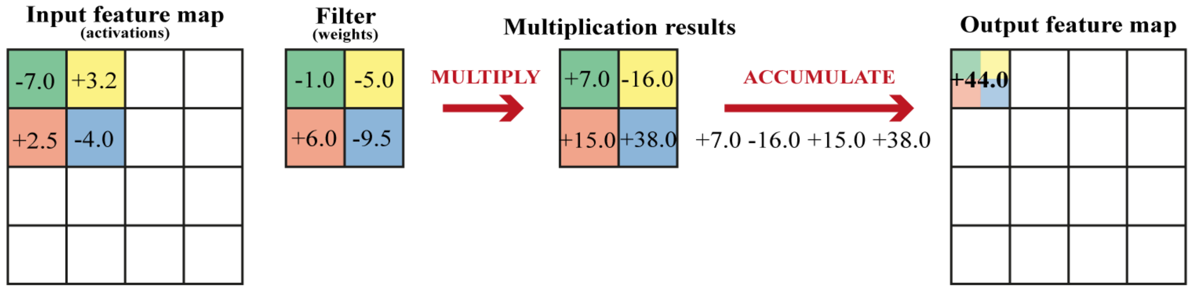

Figure 3: Example of multiply-and-accumulate (MAC) operation in CNNs

Binary convolutional neural networks (BCNNs) are based on binarized weights and activations: they can have the values '-1' (represented by '0') and '+1' (represented by '1') [5]. This means that the MAC operation can be simplified to a bitwise XNOR-bitcount operation [2]. The *bitcount* operation refers to counting the amount of *set* bits, which is the number of

‘1’ bits, that I will refer to as P. An extra operation must be computed: $2P$-N, where N is the size of the filter. In this way, the result is basically the sum between the ‘-1’ and ‘+1’ XNOR results. Figure 4 shows an example of how the MAC operation using binary weights and activations is equivalent to XNOR-bitcount, where P is the number of ‘+1’s / ‘1’s, M is the number of ‘-1’s / ‘0’s, and N is the size of the filter.

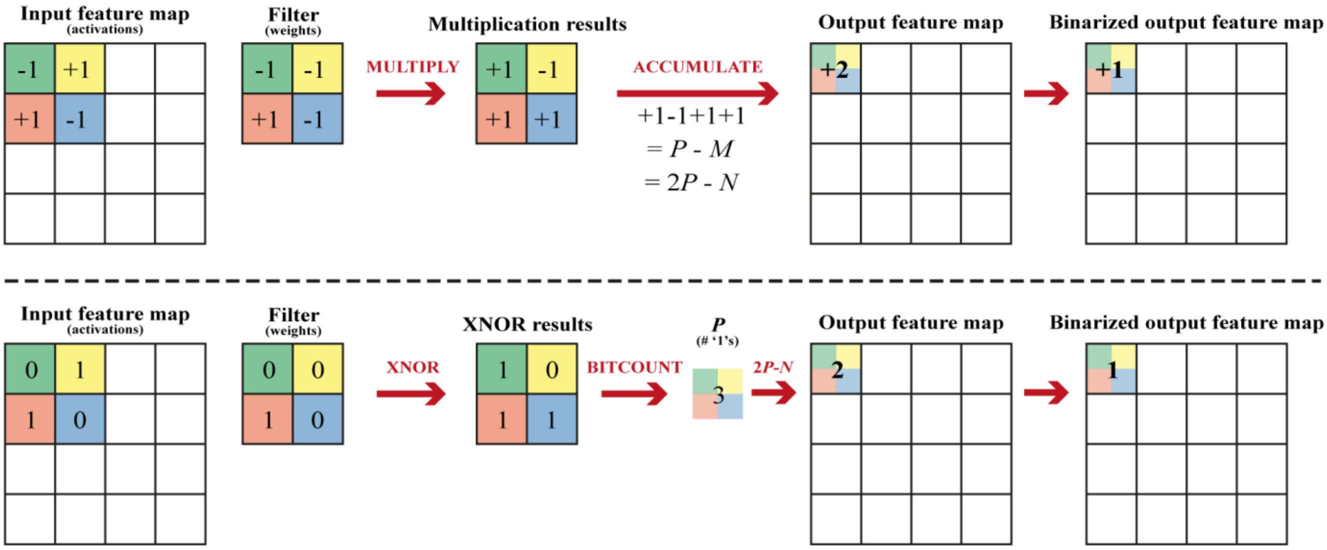

Figure 4: MAC operation equivalency to XNOR-bitcount operation when using binary weights and activations

This work only refers to the process of forward propagation, assuming that the network has already been trained and that the weights of the filters used have been set. Regardless, full-precision operations that are necessary for optimal BCNN performance – *i.e.* batch normalization, pooling – and the training process can be taken care of through auxiliary process units [18] and enhancements or modifications to the peripheral circuitry of the memory array. The STT-MRAM itself is only responsible of performing convolutions through XNOR-bitcount [18].

In forward propagation, once the output tensor – or feature map – is generated, its elements have to be binarized as it becomes the activation tensor of the next layer of the BCNN. Keep in mind that each layer of the BCNN can be made up of multiple filters, such that each one outputs a different feature map; all of those output tensors become the activation tensors of the next layer, and each one is convolved with each filter of that layer. The binarization is typically done using the sign activation function [5].

IV. XNOR-BITCOUNT METHODS BASED ON COMPUTING - IN - MEMORY

Wang *et. Al* [1] propose a method to achieve the XNOR-bitcount operation using MRAMs based on VC-SOT MTJs. In this work, I replicate this method using STT-MTJs, but with a hardware-level optimization. I further propose an algorithm-level optimization to the method presented in [1].

A. XNOR-bitcount method proposed by Wang *et al*.

Wang *et al.*'s [1] XNOR-bitcount (XNOR-BC) method uses the write and read operations of a normal memory to perform multiple Boolean operations in parallel; the bias voltage on each bitcell's VC-SOT MTJ and the direction of the SOT current are configured for this purpose. The implementation of the method in [1] in a STT-MRAM simply requires an adaptation of the write and read operations, which has been explained in Section II.C. In STT-MRAMs, the polarization of the shared BL/SL that determines the current direction through the MTJ, and the activation of the access transistors that allow said current to flow, are configured to implement Boolean logic.

Single-bit XNOR operation

For the XNOR operation in [1], the authors consider that its complementary operation, XOR, can be expressed through two AND operations and one OR operation, as shown in Equation 1:

$$XOR \longrightarrow A \oplus W = \bar{A} \cdot W + A \cdot \bar{W} \tag{1}$$

Where A is one binary input activation and W is one binary filter weight. The single-bit XOR/XNOR process uses two complementary bitcells or MTJs – MTJ_1 and $\overline{MTJ_1}$ – and the basic steps are the following:

1) *Write weights.* Write kernel weights W and $\bar{W}$ to MTJ_1 and $\overline{MTJ_1}$, respectively, in two cycles (see Figure 5.a).

2) *AND operation.* Apply input activations A and $\bar{A}$ to the gates of the access transistors for MTJ_1 and $\overline{MTJ_1}$, respectively, and apply current to write ‘0’ to both bitcells. This stores $\bar{A} \cdot W$ in MTJ_1, and $A \cdot \bar{W}$ in $\overline{MTJ_1}$.

3) *OR operation.* Read both bitcells simultaneously (see Figure 5.b); each sensing current represents the stored state $\bar{A} \cdot W$ and $A \cdot \bar{W}$, respectively, and they are added because of the connected bitlines. The resulting current represents the XOR result $\bar{A} \cdot W + A \cdot \bar{W}$. Table 3, in the fourth and fifth columns, shows the state that is read from the bitcell with MTJ_1 and the bitcell with $\overline{MTJ_1}$, respectively, during this step for all the possible activation A and weight W combinations. The last column shows the accumulated read current values that represent the XOR result.

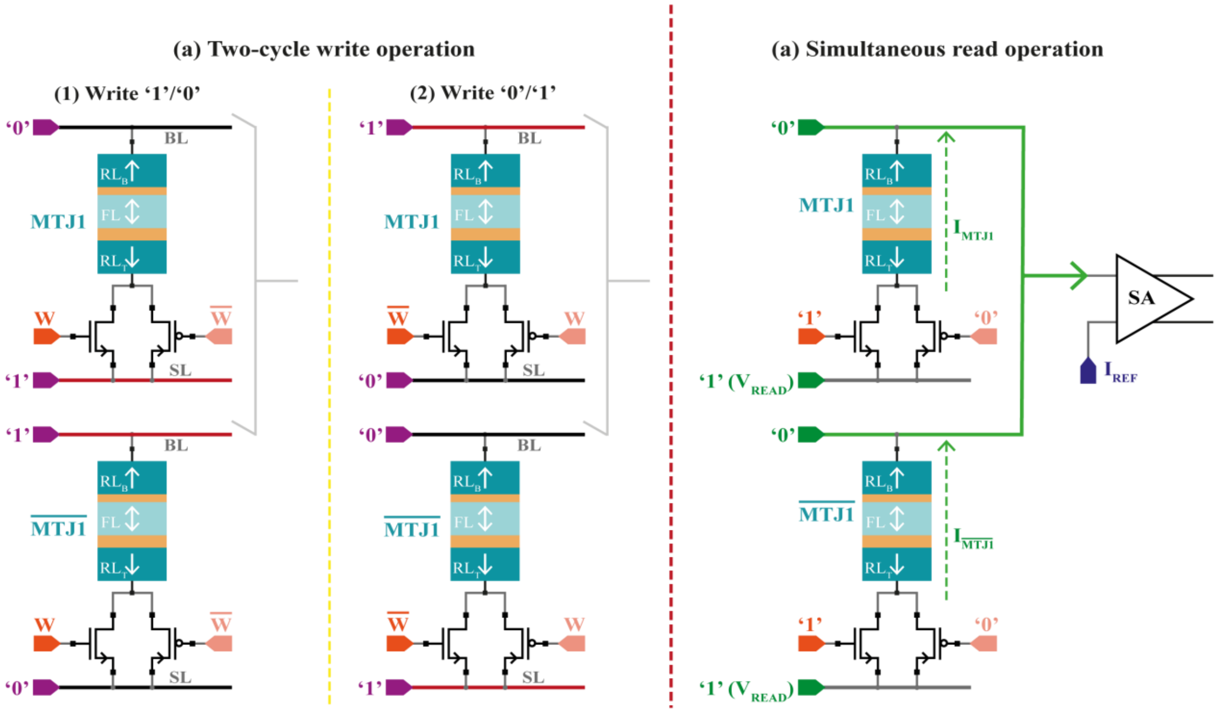

Figure 5: Wang et al.'s complementary STT-bitcell structure. (a) Two-cycle process to write weights for single-bit XOR operation, (b) Simultaneous read operation for single-bit OR computation step

A	W	**XOR**	State read from MTJ_1 $(\bar{A}\cdot W)$	State read from $\overline{MTJ_1}$ $(A\cdot\overline{W})$	**Total read current**
0	0	0	0	0	$2\times I_{read(0)}$
0	1	1	1	0	$I_{read(1)} + I_{read(0)}$
1	0	1	0	1	$I_{read(1)} + I_{read(0)}$
1	1	0	0	0	$2\times I_{read(0)}$

Table 3: Stored states read from MTJ_1 and $\overline{MTJ_1}$ during last step (simultaneous reading) and total read current according to activations A and weights W using Wang et al.'s XNOR-BC method

The two-cycle weight-writing process steps are (see Figure 5.a):

a. Apply W to the access transistor gates of both bitcells; apply current to write '1' in MTJ_1 and current to write '0' in $\overline{MTJ_1}$.

b. Apply $\overline{W}$ to the access transistor gates of both bitcells; apply current to write '0' in MTJ_1 and current to write '1' in $\overline{MTJ_1}$.

Multi-bit XNOR-bitcount operation

To compute the MAC or XNOR-bitcount result between a $k \times k$ kernel with N weights and a $k \times k$ feature map portion with N input activations, the corresponding N XOR/XNOR operations can be computed in parallel using the concept of *'analog majority'* [1]. Two complementary bitcell *strips* are used: all the bitcells with MTJ_Ns are on the same bitline ($strip_1$), and all the $\overline{MTJ_N}$s are on another single bitline ($\overline{strip_1}$); their BLs are connected to the same SA. Figure 6 shows this memory-array structure for $N = 9$ using STT-MTJs.

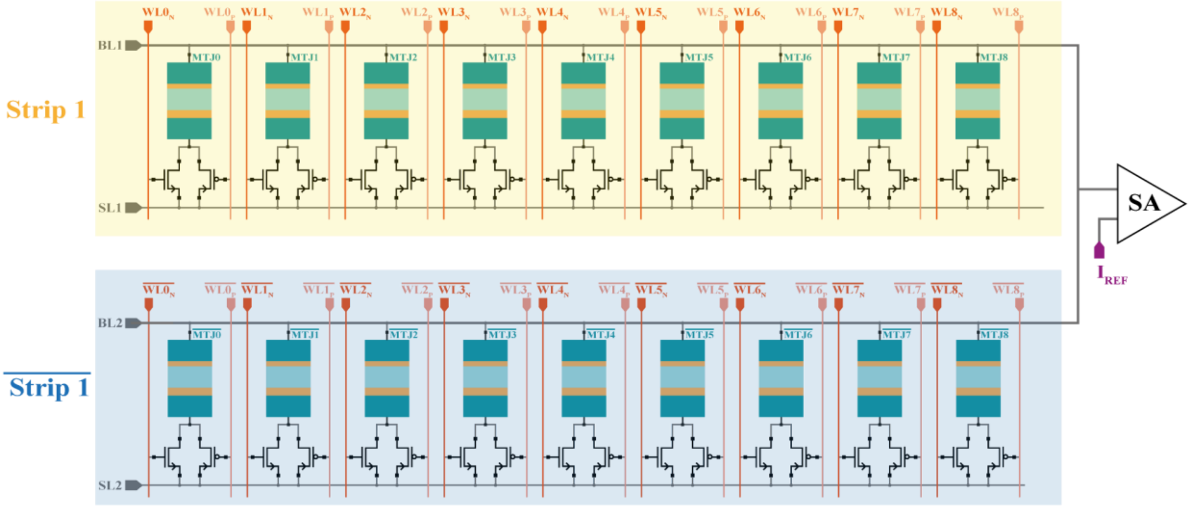
$WL0_N$ $WL0_P$ $WL1_N$ $WL1_P$ $WL2_N$ $WL2_P$ $WL3_N$ $WL3_P$ $WL4_N$ $WL4_P$ $WL5_N$ $WL5_P$ $WL6_N$ $WL6_P$ $WL7_N$ $WL7_P$ $WL8_N$ $WL8_P$
BL1
SL1
MTJ0 MTJ1 MTJ2 MTJ3 MTJ4 MTJ5 MTJ6 MTJ7 MTJ8
Strip 1
SA
I_{REF}
$\overline{WL0_N}$ $\overline{WL0_P}$ $\overline{WL1_N}$ $\overline{WL1_P}$ $\overline{WL2_N}$ $\overline{WL2_P}$ $\overline{WL3_N}$ $\overline{WL3_P}$ $\overline{WL4_N}$ $\overline{WL4_P}$ $\overline{WL5_N}$ $\overline{WL5_P}$ $\overline{WL6_N}$ $\overline{WL6_P}$ $\overline{WL7_N}$ $\overline{WL7_P}$ $\overline{WL8_N}$ $\overline{WL8_P}$
BL2
SL2
$\overline{\text{Strip 1}}$

Figure 6: Wang et al.'s complementary bitcell-strips structure using STT-MTJs for multibit computation with N=9

Keeping in mind that '0' actually represents '-1' in a BCNN and attending to Equation 1, if the majority of XOR results are '0', then the real XOR-BC result would be *negative* and after binarization it would be '-1' ('0'); if the majority are '1', the real XOR-BC result would be *positive* and its binary value would be '1'. This majority detection and binarization is performed by reading all of the bitcells simultaneously and comparing the accumulated current I_{N-XORs} to a properly set reference current I_{ref} in the SA. If $I_{N-XORs} > I_{ref}$ then the majority of XOR results were '0' and the XOR-BC result is '0', and if $I_{N-XORs} < I_{ref}$, then the majority were '1' and the XOR-BC result is '1'. Both the binary XOR-BC result and its complement XNOR-BC are produced by the SA.

B. Hardware-level optimization

The two-cycle process to write the weights and the simultaneous read operation for a single-bit XOR operation were previously shown in Figure 5.a and Figure 5.b, respectively, using the complementary bitcell structure proposed by Wang *et al.* [1] but implemented with STT-MTJs.

With Wang *et al.*'s structure, each bitcell in the complementary pair occupies a different BL, and *they don't share the WL* (as seen in Step 2, complementary gate voltages are used). This means that a single-bit XNOR operation requires 2 BLs and 2 WLs of the memory array; *N* parallel XNOR operations – which use two complementary bitcell strips – would require 2 BLs and 2×*N* WLs. Moreover, this structure requires a connection between the bitlines of complementary strips to a single SA.

I propose a hardware-level optimization that consists on locating each pair of complementary bitcells in the same bitline. Figure 7 shows this optimization.

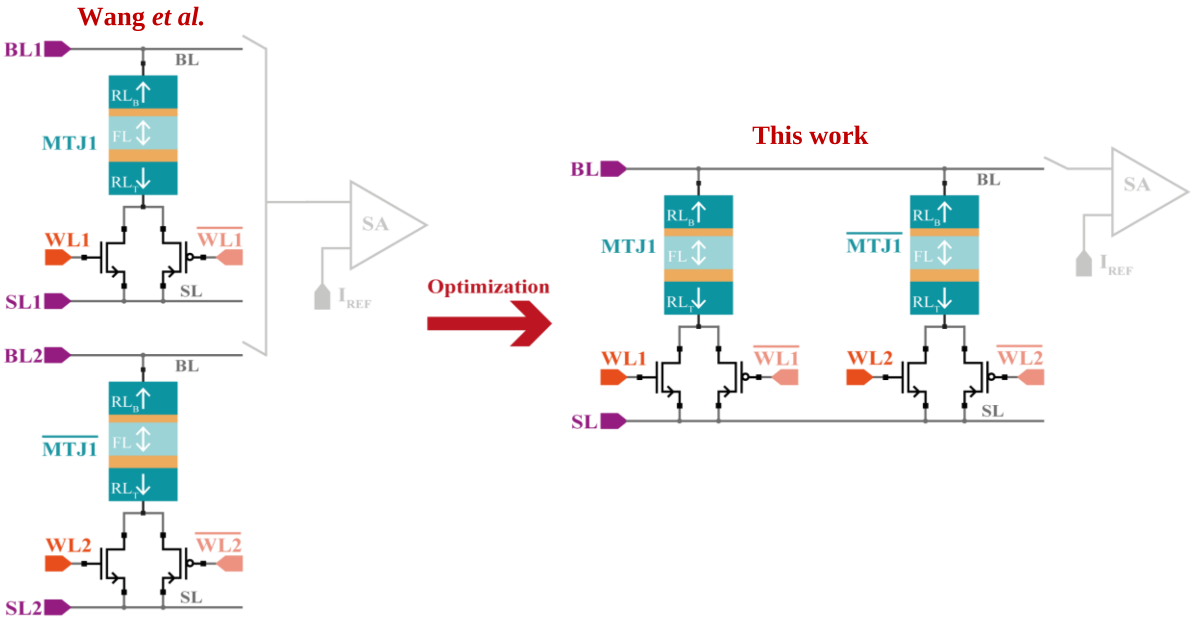

Figure 7: Hardware-level optimization for a pair of complementary bitcells

Regarding storage density, *N* parallel XNOR operations would require just 1 BL and 2×*N* WLs, as the complementary bitcell strips would be on the same bitline. Clearly, the BL would already be “connected” between both strips to the same SA.

This optimization implies a slight modification to the two-cycle weight-writing process: the same state must be written to both bitcells at the same time, while complementary access transistor gate voltages are applied. This modification can be taken further by writing ‘0’ to all the bitcells – “resetting” them – in Step 1, and writing the ‘1’ weights in the second step using $\boldsymbol{W}$ and $\overline{\boldsymbol{W}}$ as gate voltages. The first step implies an increased energy consumption; however, in multi-kernel operations in which the kernel weights are written sequentially into the memory, only one “write ‘0’/‘0’ step” would be necessary at the beginning of the weight-writing process followed by only “ write ‘1’/ ‘1’ ” steps, instead of sequential “ write ‘0’/ ‘0’ ” + “ write ‘1’/ ‘1’ ” steps for every kernel. Figure 8 shows the proposed complementary bitcell structure for the two-cycle write and simultaneous read operations.

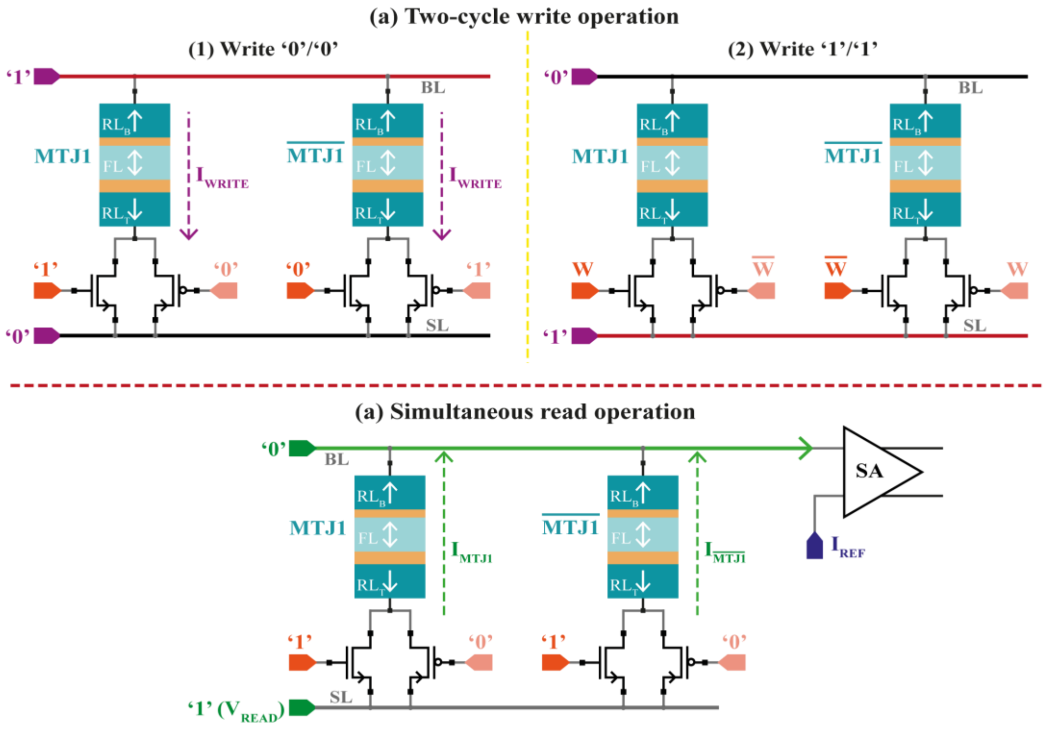

Figure 8: This work’s optimized complementary STT-bitcell structure: (a) Two-cycle process to write weights for single-bit XOR operation, (b) Simultaneous read operation for single-bit OR step

The modified two-cycle weight-writing process steps, shown in Figure 8.a, are:

a. Activate the access transistor of both bitcells; apply current to write ‘0’ to MTJ_1 and $\overline{MTJ_1}$.

b. Apply W and $\overline{W}$ to the gates of the access transistors to MTJ_1 and $\overline{MTJ_1}$, respectively; apply current to write ‘1’ in both bitcells.

Regarding the complementary bitcell strips structure, Figure 6 in Section IV. A showed it using the array structure proposed in [1]. Keep in mind that each pair of complementary MTJs does not share the same WL. This array would be used for a XNOR-bitcount operation of a 9-bit filter (3x3) with a 9-bit feature map portion. Figure 9 shows the complementary strips using this work’s hardware optimization, in which they share the same BL and SL.

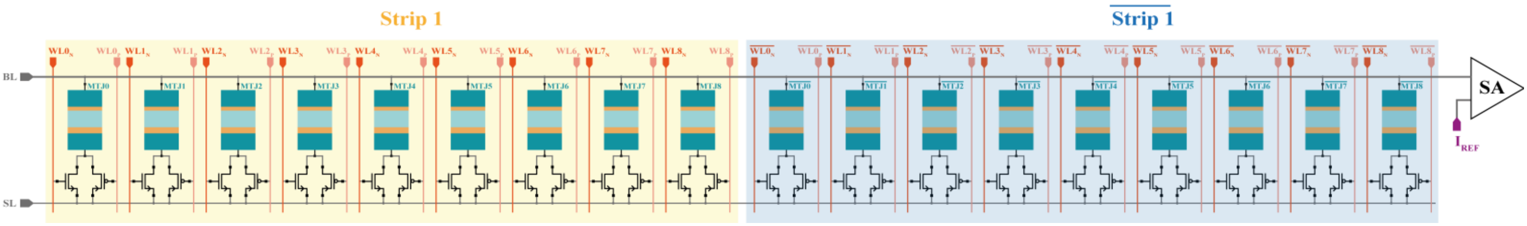
Strip 1
$\overline{\text{Strip 1}}$
BL
SL
$WL0_N$ $WL0_P$ $WL1_N$ $WL1_P$ $WL2_N$ $WL2_P$ $WL3_N$ $WL3_P$ $WL4_N$ $WL4_P$ $WL5_N$ $WL5_P$ $WL6_N$ $WL6_P$ $WL7_N$ $WL7_P$ $WL8_N$ $WL8_P$
MTJ0 MTJ1 MTJ2 MTJ3 MTJ4 MTJ5 MTJ6 MTJ7 MTJ8
$\overline{WL0_N}$ $\overline{WL0_P}$ $\overline{WL1_N}$ $\overline{WL1_P}$ $\overline{WL2_N}$ $\overline{WL2_P}$ $\overline{WL3_N}$ $\overline{WL3_P}$ $\overline{WL4_N}$ $\overline{WL4_P}$ $\overline{WL5_N}$ $\overline{WL5_P}$ $\overline{WL6_N}$ $\overline{WL6_P}$ $\overline{WL7_N}$ $\overline{WL7_P}$ $\overline{WL8_N}$ $\overline{WL8_P}$
$\overline{MTJ0}$ $\overline{MTJ1}$ $\overline{MTJ2}$ $\overline{MTJ3}$ $\overline{MTJ4}$ $\overline{MTJ5}$ $\overline{MTJ6}$ $\overline{MTJ7}$ $\overline{MTJ8}$
SA
I_{REF}

Figure 9: This work’s optimized complementary bitcell strips structure using STT-MTJs for multibit computation with N=9

C. Wang *et al.*’s XNOR-bitcount method with STT-MRAMs and optimized hardware

Figure 10 shows Wang *et al.*’s proposed method [1] using the optimized array structure for a single-bit XOR/XNOR operation, in which the complementary bit-strips share the BL.

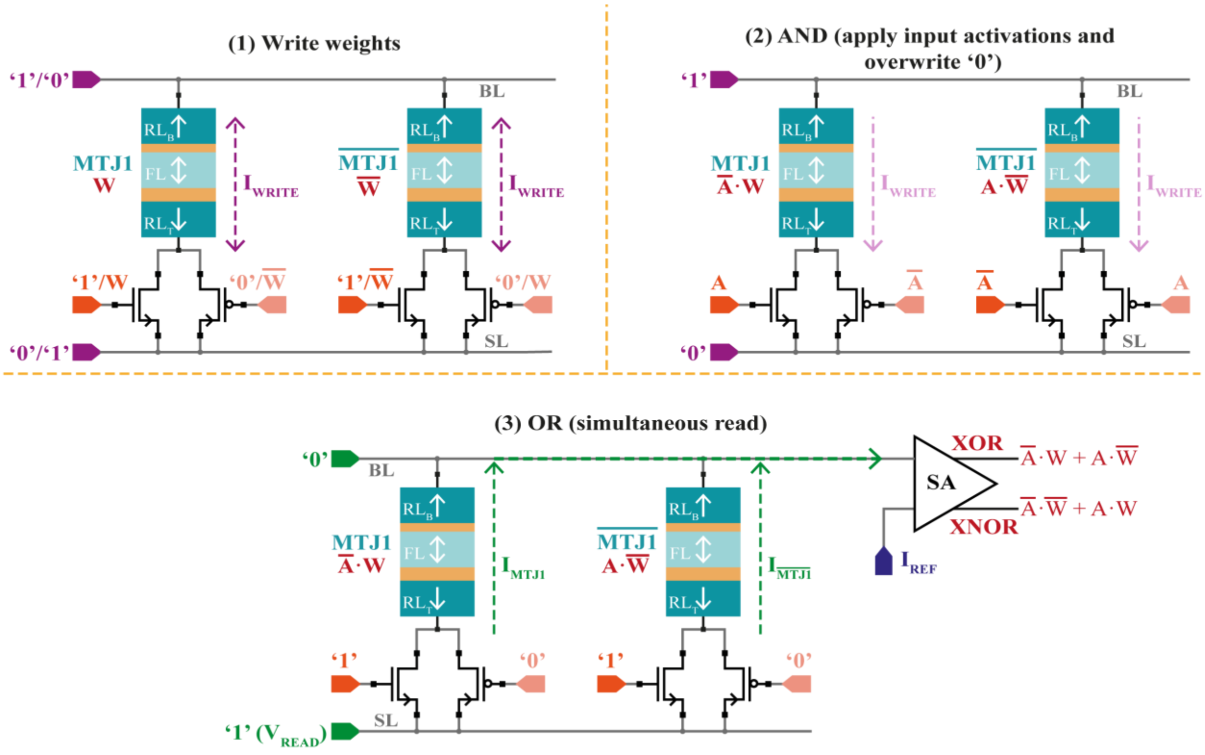

Figure 10: Wang et al.'s single-bit XOR/XNOR operation using STT-MTJs and optimized array structure

Figure 10.1 shows the first step in which the weights are written into the bitcells' MTJs, Figure 10.2 shows the AND operation, and Figure 10.3 shows the OR operation in which all the bitcells are simultaneously read to obtain the accumulated current representative of the complete XOR-BC operation.

D. Optimized XNOR-bitcount method with STT-MRAMs and optimized hardware.

Single-bit XNOR-bitcount operation using algorithmic optimization

Beyond the memory array structure optimization, this work proposes a more important algorithmic optimization to the XNOR-BC method proposed by Wang *et al*. [1]: the second (AND operation) and third step (OR operations and analog majority) of the XNOR-BC method can be merged together into a single read step based on the input activations. I consider the expression of the XNOR operation through two AND operations and one OR operation, as shown in Equation 2**:**

$$XNOR \longrightarrow \overline{A \oplus W} = \mathrm{A} \cdot W + \bar{A} \cdot \bar{W} \tag{2}$$

This methodology allows the direct computation of the XNOR-BC result, instead of the XOR-BC. The optimized single-bit process is illustrated in Figure 11, in which the exact same optimized hardware as in Figure 10 is used.

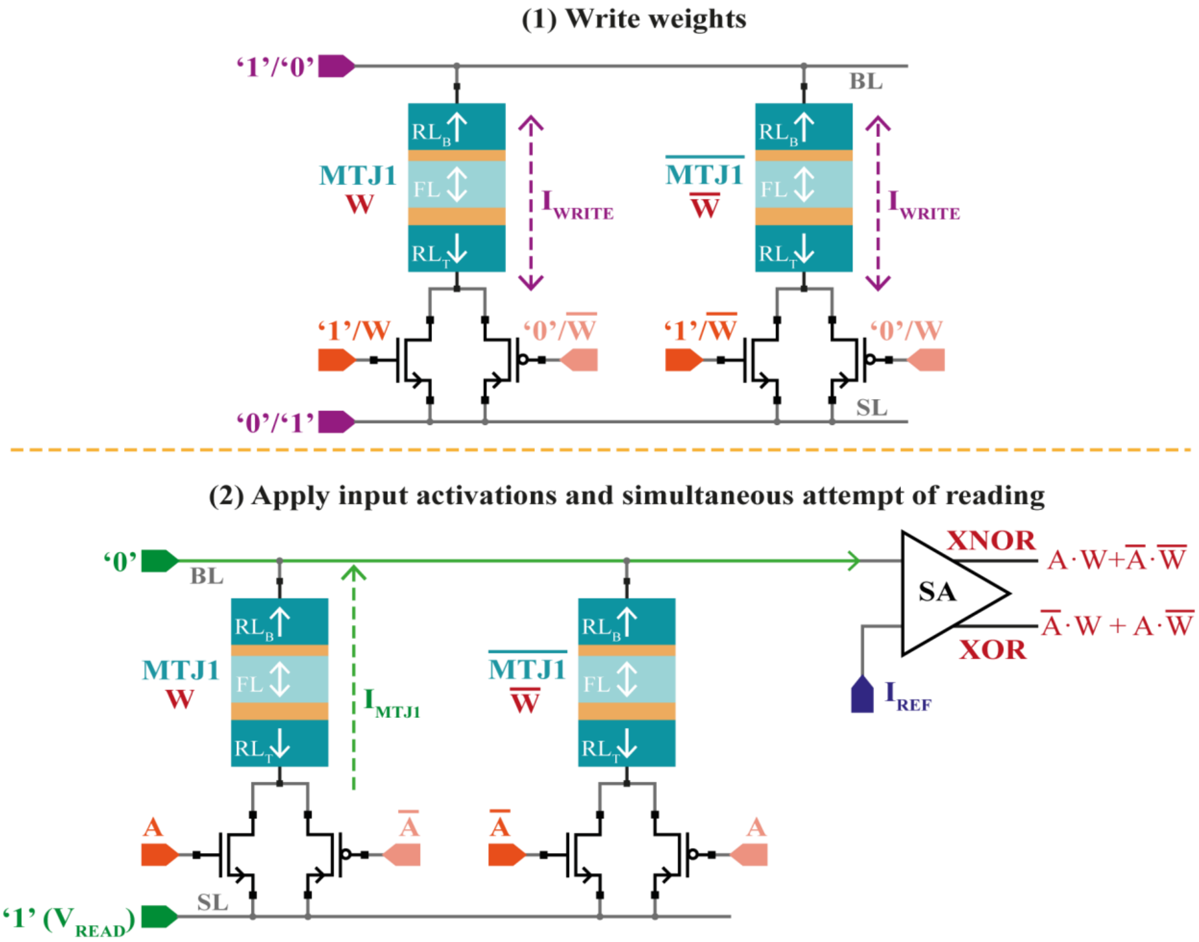

Figure 11: This work's single-bit XOR/XNOR operation using STT-MTJs and optimized array structure

The optimized method’s basic steps are (see Figure 11):

1) *Write weights.* Write kernel weights W and $\overline{W}$ to MTJ_1 and $\overline{MTJ_1}$, respectively, in two cycles (see Figure 8.a).

2) *AND & OR operation.* Apply input activations A and $\bar{A}$ to the gates of the access transistors for MTJ_1 and $\overline{MTJ_1}$, respectively, and read both bitcells simultaneously (see Figure 8.b). Only one bitcell is read because of the complementary activations used as V_g. The reading of the bitcell with MTJ_1 results in the AND operation $\mathrm{A} \cdot W$, and the reading of the bitcell with $\overline{MTJ_1}$ results in the AND operation $\bar{A} \cdot \overline{W}$. The OR operation between them occurs because they share the same bitline and the complementary bitcells are operated simultaneously, so the resulting current represents the XNOR result $\mathrm{A} \cdot W + \bar{A} \cdot \overline{W}$. Table 4, in the fourth and fifth columns, shows the state that is read from the bitcell with MTJ_1 and the bitcell with $\overline{MTJ_1}$, respectively, during this step for all the possible activation A and weight W combinations. The last column shows the accumulated read current values that represent the XNOR result.

A	W	**XNOR**	State read from MTJ_1 $(\bar{A} \cdot W)$	State read from $\overline{MTJ_1}$ $(A \cdot \overline{W})$	**Total read current**
0	0	1	*Not read*	1	$I_{read(1)}$
0	1	0	*Not read*	0	$I_{read(0)}$
1	0	0	0	*Not read*	$I_{read(0)}$
1	1	1	1	*Not read*	$I_{read(1)}$

Table 4: Stored states read from MTJ_1 and $\overline{MTJ_1}$ during last step (simultaneous reading) and total read current according to activations A and weights W using this work’s XNOR-BC method

Multi-bit XNOR-bitcount operation using algorithmic optimization

To compute the MAC or XNOR-bitcount result between a $k \times k$ kernel with N weights and a $k \times k$ feature map portion with N input activations, the N XNOR operations are computed in parallel using the same two complementary bitcell strips as in Wang *et al.*'s method. In Step 1 the weights $\boldsymbol{W} = [W_1, W_2, \ldots, W_n]$ and $\overline{\boldsymbol{W}}$ are parallelly written into the complementary strips. In Step 2 the input activations $\boldsymbol{A} = [A_1, A_2, \ldots, A_n]$ and $\overline{\boldsymbol{A}}$ are parallelly applied to the bitcells' access transistor gates and they are all simultaneously read (although only half of the bitcells in the two complementary strips are really read); using the same analog majority approach, the currents produced by each pair of complementary bitcells (representative of the XNOR results) are accumulated resulting in $I_{N-XNORs}$, and when compared to a proper reference signal in a SA, the binarized XNOR-bitcount result is obtained. In this method, if $I_{N-XNORs} > I_{ref}$, then the majority of XNOR results were '0' and the XNOR-BC result is '0', and if $I_{N-XNORs} < I_{ref}$, then the majority were '1' and the XNOR-BC result is '1'.

The advantages of this optimized XNOR-bitcount method are the following:

- Only two write operations (write '0' and write '1' when writing weights) are needed instead of three.
- The weights stored in the bitcells are *not overwritten* during the process. This means that if consecutive XNOR-bitcount operations are performed using the same filter (weights) but different feature map portions (activations), the weights do not have to be re-written and only the reading step (analog majority) has to be performed. This means that when performing a convolution between a filter and a feature map, the first MAC or XNOR-bitcount is performed as mentioned earlier, but the XNOR-BCs that follow

– between the same filter and other feature map portions –, consist of only the simultaneous read operation (that computes the AND and OR operations) with specific access transistor activations (Step 2).

- In Step 2, only half of the *2N* bitcells are activated instead of all of them (assuming a *N*-bit filter). This means less read current, and therefore less energy consumption.
- The direct result of the SA is XNOR-bitcount, instead of XOR-bitcount.

V. DESIGN OF STT-MRAM AND SIMULATION RESULTS OF XNOR-BITCOUNT METHODS

A. STT-MRAM bitcell design

As mentioned earlier, the DMTJ device is described by macrospin-based Verilog-A compact models [16] and the access transistors belong to TSMC's commercial 65-nm technology node. The Synopsys Custom Compiler tool was used for simulations.

VDD selection and access transistor sizing

STT-MRAMs have the advantage of being able to operate at relatively low voltages [9]. Therefore, a polarization voltage of 0.8V was chosen in this work, which can be considered to be in the upper limit of the low-voltage range:

$$V_{DD} = 0.8\,V$$

Lower voltages were not chosen because they required the access transistors to be too large in order for them to ensure a proper I_{write}/I_c ratio and an adequate sense margin in the reading operation [17]. Bear in mind that in STT-MRAM bitcells, most of the bitcell area is occupied by the access transistors, not the MTJ [19], so minimizing the transistor sizes is necessary.

Write operation constraints for transistor sizing

To ensure a robust write operation, a write current ratio I_{write}/I_c greater than 2 is desired [15]. The write currents ratios of each switch are set to the same value. The write current is set to ensure a write-error-rate (WER) of 10^{-7} [9]. For ‘1’ to ‘0’ switching, the n-MOS transistor is basically the only driver. For ‘0’ to ‘1’ switching, the p-MOS transistor is the main driver, but the n-MOS transistor contributes to the write current. Therefore, the sizing process consists on setting w_p to the minimum width allowed ($w_{min} = 120\ nm$), and preforming a sweep to find w_n. With w_n set, a sweep is performed to find w_p.

Read operation constraints for transistor sizing

For the read operation, both read disturbance and read decision failures must be avoided [20] [21]. To avoid read disturbance failures, which are accidental switchings of the MTJs states by too-large read currents [17], an appropriate V_{read} should be calculated. Reference [21] presents the following equation from which I_{read} can be determined by setting the read disturbance rate (RDR) to the common value of $RDR = 10^{-9}$:

$$RDR = 1 - \exp\left[-\frac{t_{read}}{\tau}\exp\left(-\Delta\left(1-\frac{I_{read}}{I_{c0}}\right)\right)\right] \tag{3}$$

t_{read} is the read pulse width, τ is the attempt period ($\sim 1\ ns$), and Δ is the thermal stability factor (see Table 1). I_{read} is the current that flows through the DMTJ from the SL to the BL when a sensing voltage V_{read} is applied to the SL. I_{c0} is the critical switching current of the DMTJ (see Table 1). From Equation 3 we get the following result:

$$I_{read} = 9.9868\ [uA]$$

The largest read current that flows through the DMTJs is the read current for the ‘0’ state $I_{read(0)}$, given that R_0 corresponds to the lowest resistance state and a constant V_{read} is applied. Therefore, the following condition must be achieved in order to prevent read disturbances:

$$I_{read(0)} \leq I_{read} \tag{4}$$

To avoid read decision failures, which occur when the sensing circuitry cannot distinguish between the two states of the MTJ [20], the reading operation should be robust enough. This robustness can be quantified using the read-error-ratio (RER), that refers to the amount of wrongfully read bits from the total. The RER can be evaluated through the amount of standard deviations that fit the space between the median $\mu_{I_{read(0)}}$ of the $I_{read(0)}$ distribution and I_{ref}, and between the median $\mu_{I_{read(1)}}$ of the $I_{read(1)}$ distribution and I_{ref} [17]. A 3σ sensing margin (SM), which allows a RER of approximately 10^{-3} [21], is chosen for the design. This means that the distance between $\mu_{I_{read(0)}}$ and $\mu_{I_{read(1)}}$ (the nominal SM) should be:

$$\mu_{I_{read(0)}} - \mu_{I_{read(1)}} \geq 3\sigma_{I_{read(0)}} + 3\sigma_{I_{read(1)}} \tag{5}$$

Access transistor sizing

$I_{write}/I_c \approx 2.6$ was chosen. Lower write current ratios allowed smaller transistor widths, but the associated process variations resulted in worsened sense margins – lower that 3σ. Larger access transistor sizes are necessary to minimize said variations [21]. Larger widths also allow higher write currents and shorter write delays, and the latter allow lower write energy. However, the area occupied by the transistors should be as small as possible.

Keeping this trade-off in mind, a careful sizing methodology was followed that resulted in the following access transistor widths:

$w_n\ [nm]$	310
$w_p\ [nm]$	310

Table 5: Access transistor widths

The corresponding current I, delay t and energy E nominal results for the writing operation ($WER = 10^{-7}$) and the reading operation ($RDR = 10^{-9}$, $3\sigma\ SM$) are shown in Table 6 and Table 7, respectively.

State switch	$I_{write}\ [\mu A]$	$t_{write}\ [ns]$	$E_{write}\ [fJ]$
$(\mathbf{0} \rightarrow \mathbf{1})$	37.22	2.577	76.80
$(\mathbf{1} \rightarrow \mathbf{0})$	37.19	2.580	76.80

Table 6: Write operation current, delay and energy results at bitcell level

$V_{read} = 95\ [mV]$			
State	$I_{read}\ [\mu A]$	$t_{read}\ [ns]$	$E_{read}\ [fJ]$
‘0’	7.853	1	0.7461
‘1’	4.599		0.4369

Table 7: Read operation current, delay and energy results at bitcell level

The Monte Carlo analysis results for the reading operation are shown in Figure 12. For DMTJs, Gaussian-distributed process variations are considered for the area, FL and top/bottom oxide layers [9]. For the transistors, the technology node’s Monte Carlo library was used.

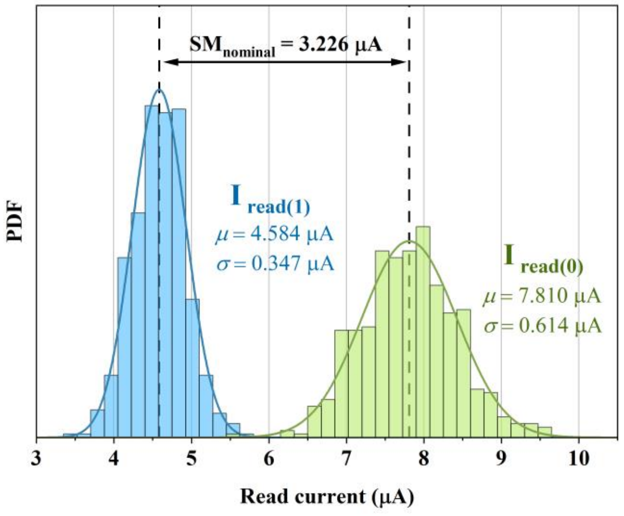

Figure 12: Monte Carlo results for read operation considering process variations

Considering the read current means and standard deviations depicted in Figure 12, the spacing between read currents medians – nominal SM – is:

$$\mu_{I_{read(0)}} - \mu_{I_{read(1)}} = 3.226\ \mu A$$

And the minimum spacing is:

$$3\sigma_{I0} + 3\sigma_{I1} = 2.883\ \mu A$$

Even though there is a slight overlap between $I_{read(0)}$ and $I_{read(1)}$ distributions, the chosen transistor sizes allow a reliable differentiation between the parallel and antiparallel MTJ states. To improve the reading operation robustness, the transistor sizes can be increased [21], albeit larger bitcell area, or peripheral circuitry (SAs) with specific designs can be used [22].

B. Implementation of XNOR-bitcount methods and simulation results

To test the XNOR-bitcount method proposed in [1] implemented with STT-MTJs and optimized hardware, as well as this work's optimized XNOR-bitcount method, Synopsys' Custom Compiler was used to build the hardware and simulate said methods using vector files.

Logic verification of methods

Using the $I_{read(0)}$ and $I_{read(1)}$ values in Table 7, and considering the total read currents produced in each XOR/XNOR operation shown in Table 3 and Table 4, the following tables can be built that show the expected accumulated current I_{9-XORs} and $I_{9-XNORs}$ for each number of '1' XOR and XNOR results in a 9-bit XOR/XNOR-bitcount operation (*i.e.* 3x3 filter and 3x3 feature map portion). Table 8 shows the expected results using Wang *et al.*'s method. Keep in mind that the number of stored '1's in the 18 MTJs is the number of '1's of the XOR-BC operation, which is the number of '0's in the XNOR-BC operation. Table 9 shows the expected results using this work's method. The approximate reference currents I_{ref} that could be used are shown in both tables, calculated using the 3σ rule presented in Section V. A.

$I_{ref} \approx 126.5\ [\mu A]$

# '1' XORs	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
$\boldsymbol{I_{9-XORs}\ [\mu A]}$	141.36	138.11	134.8523	131.60	128.34	125.09	121.84	118.58	115.33	112.07
XOR-BC result	0	0	0	0	0	1	1	1	1	1
XNOR-BC result	1	1	1	1	1	0	0	0	0	0

Table 8: Accumulated currents using Wang's method corresponding to each XNOR-bitcount result

$I_{ref} \approx 55.8\ [\mu A]$

# '1' XNORs	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
$\boldsymbol{I_{9-XNORs}\ [\mu A]}$	70.68	67.42615	64.17	60.92	57.66	54.41	51.16	47.90	44.65	41.39
XOR-BC result	1	1	1	1	1	0	0	0	0	0
XNOR-BC result	0	0	0	0	0	1	1	1	1	1

Table 9: Accumulated currents using this work's method corresponding to each XNOR-bitcount result

Multi-kernel XNOR-bitcount simulation

To test the correct functioning of both methods applied to STT-MRAMs and to show their parallelism and high-throughput, the same three XNOR-bitcount operations as in a example provided in [1] were used: three 3x3 filters are operated with a single 3x3 feature map portion (activations). Table 10 presents the weights and activations. XNOR-BC 1 has 4 '1' XNOR results, XNOR-BC 2 has 5 '1's, and XNOR-BC 1 has 2 '1's.

	W (filter weights)	**A** (feature map portion)	XNOR	XNOR-bitcount
XNOR-BC 1	010100001	010001110	111010000	0
XNOR-BC 2	101011110		000101111	1
XNOR-BC 3	101010101		000100100	0

Table 10: Weights and activations to evaluate XNOR-bitcount methods

The three filters presented in Table 10 are simultaneously operated with a single set of 9 activations. The memory array section (using this work's optimized structure) that would allow this multi-kernel parallel computation is presented in Figure 13.

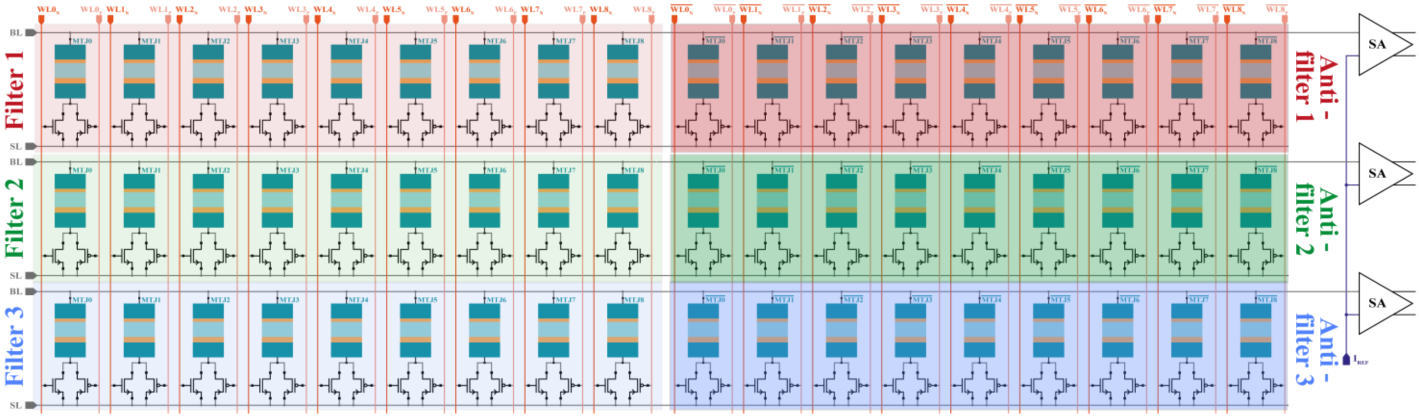

Figure 13: Memory array section for 9-bit multi-kernel parallel XNOR-BC computations

The simulation waveforms, that show the complete parallel XNOR-bitcount operations of the three examples given in Table 10, are presented in Figure 14 and Figure 15.

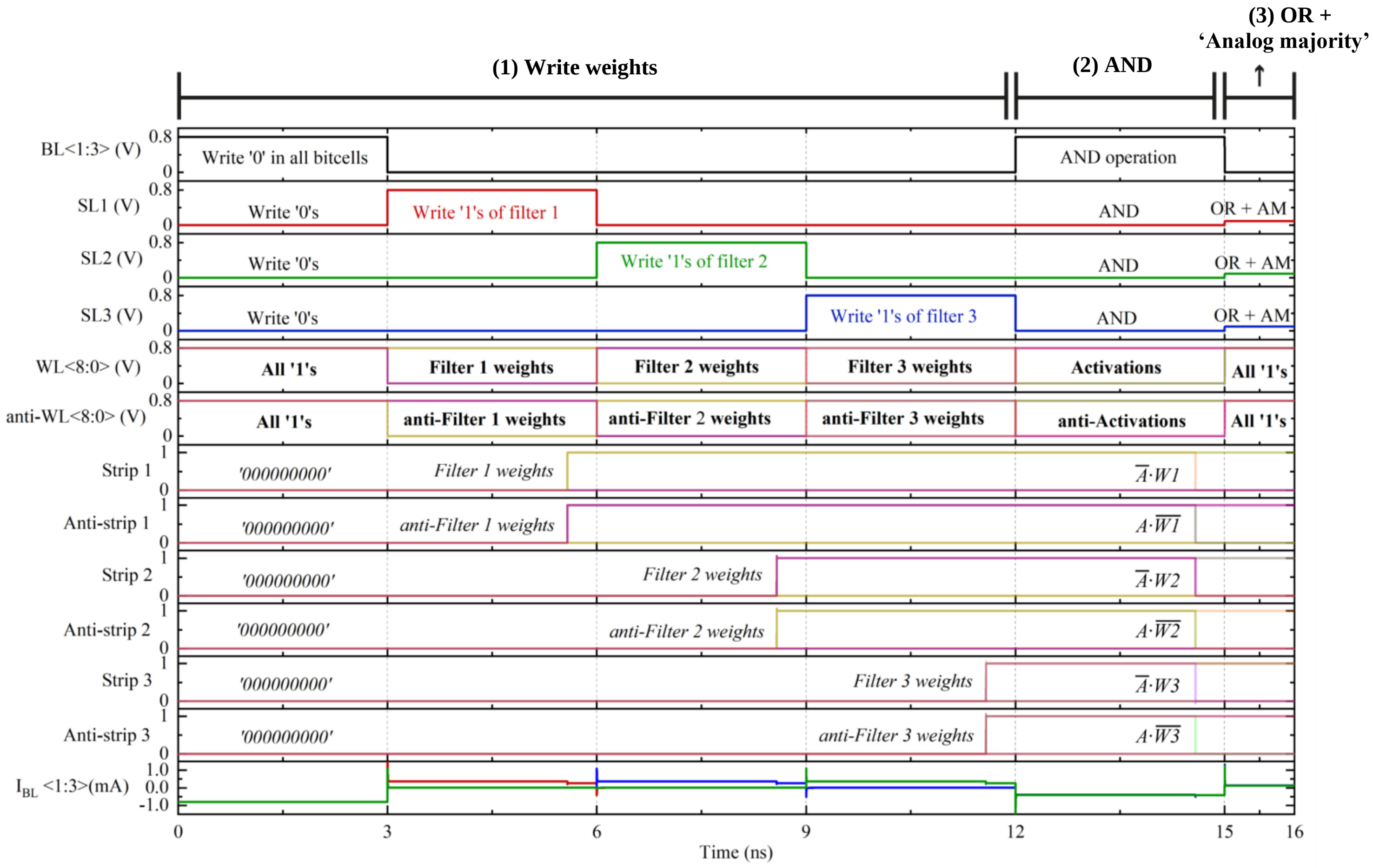

Figure 14: Simulation waveforms using Wang's method of the three parallel XNOR-bitcount operations in Table 10

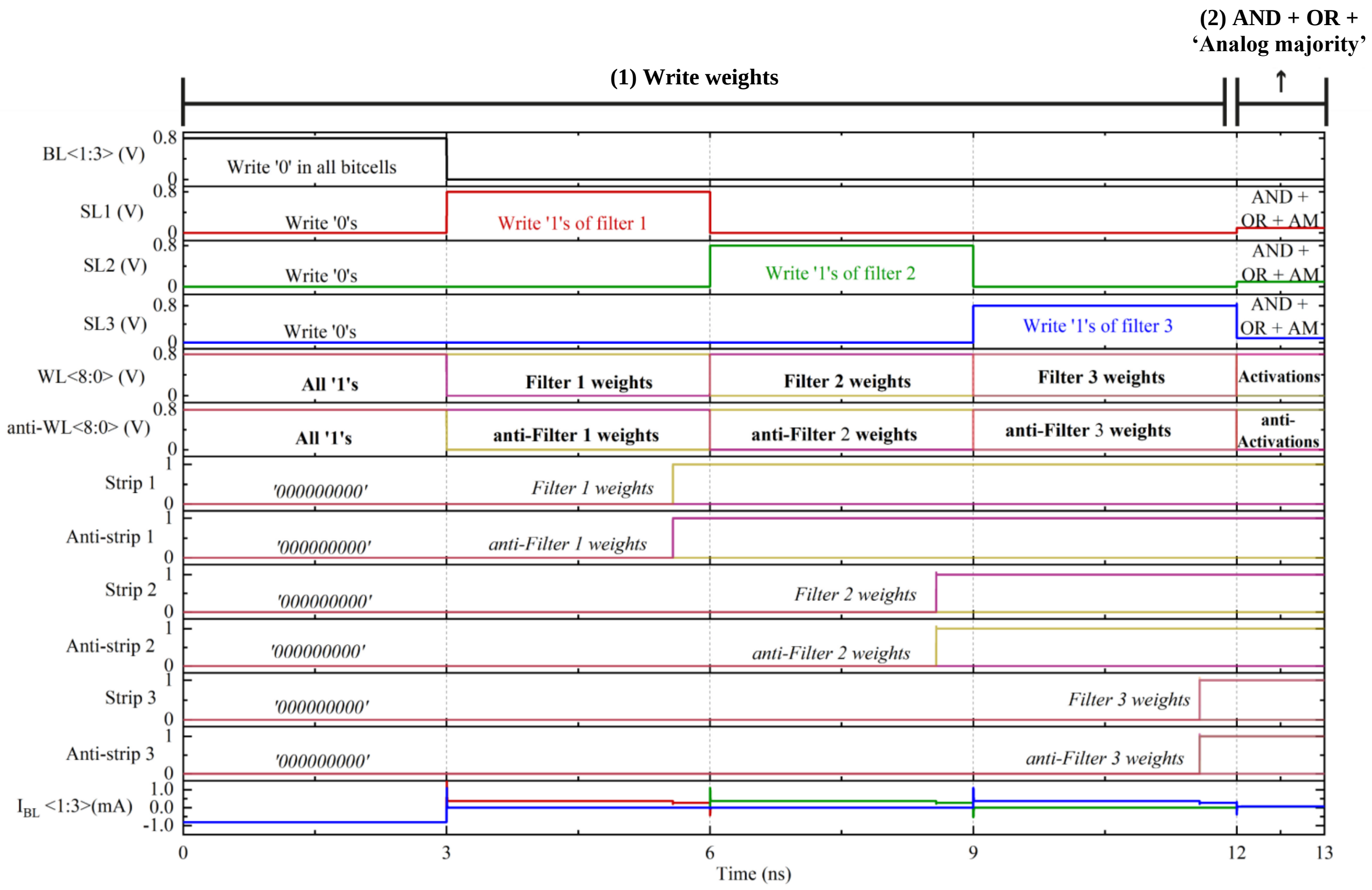

Figure 15: Simulation waveforms using this work's method of the three parallel XNOR-bitcount operations in Table 10

The write pulse is 3 ns long (to approximate t_{write} in Table 6) and the read pulse is 1 ns long. By measuring the accumulated current in the last step of each XNOR-BC for both methods, Table 11 is built.

	Expected XNOR-BC result	**Wang's I_{9-XORs} [μA]**	**Wang's XNOR-BC result**	**This work's $I_{9-XNORs}$ [μA]**	**This work's XNOR-BC result**
XNOR-BC 1	0	125.09	0	57.66	0
XNOR-BC 2	1	128.34	1	54.41	1
XNOR-BC 3	0	118.58	0	64.17	0

Table 11: Accumulated currents measured for each XNOR-BC operation

If the measured currents are compared with the I_{9-XORs} and $I_{9-XNORs}$ values in Table 8 and Table 9, the XNOR-BC result obtained by each method can be found. As can be observed in Table 11, the obtained results by Wang *et al.*'s method and this work's method are correct. This verifies the correct functioning of this work's proposed method and the parallelism that it allows for multi-kernel operations.

Energy consumption and execution time comparison between XNOR-bitcount methods

Worst-case energy and time comparison for single XNOR-BC with 3x3 filter

In order to compare the methods in terms of execution time and energy consumption, the necessary time to execute the whole operation and the worst case energy $Energy_{WC}$ corresponding to a XNOR-bitcount operation between a 3x3 kernel/filter and a 3x3 portion of a feature map are calculated from the results presented in Table 6. These results are shown in Table 12 and Table 13.

	$Energy_{WC}$ [fJ]			
	STEP 1 **Write weights**	STEP 2 **AND (write '0')**	STEP 3 **Read & analog majority**	TOTAL **XNOR-BC**
Wang *et al.*	2355.96	832.39	10.65	**3199.00**
This work	2355.96	-	6.71	**2362.68**

Table 12: Worst case energies for each XNOR-bitcount step and total energy consumption

	Execution time [ns]			
	STEP 1 **Write weights**	STEP 2 **AND (write '0')**	STEP 3 **Read & analog majority**	TOTAL **XNOR-BC**
Wang *et al.*	6	3	1	**10**
This work	6	-	1	**7**

Table 13: Execution times for each XNOR-bitcount step and total execution time

Given that overwriting 0 in half of the 18 bitcells is not performed in this work's method (step 2), and that only 9 bitcells are read instead of 18 in the last step, there is a clear advantage in energy consumption and execution time.

Sequential XNOR-BC operations: energy and time comparison

These advantages are enhanced when multiple consecutive XNOR-bitcounts are performed between the same 3x3 filter and multiple 3x3 portions of a feature map (representing multiple steps of the convolution process between said filter and the feature map). This is because this work's method doesn't require re-writing the filter weights in the bitcell in each XNOR-bitcount operation if the same filter is being used. The results are shown in Table 14, assuming that 5 consecutive XNOR-bitcount operations are performed using the same filter (which means that the same pair of complementary 9-bitcell strips are used).

	$Energy_{WC}$ [fJ]			**Execution time [ns]**		
	1st XNOR-BC	**Following XNOR-BCs**	**Total of 5 consecutive XNOR-BCs**	**1st XNOR-BC**	**Following XNOR-BCs**	**Total of 5 consecutive XNOR-BCs**
Wang *et al.*	3199.00	3199.00	15995.00	10	10	50
This work	2362.68	6.71	2389.54	7	1	11

Table 14: Energy and execution time results for 5 consecutive XNOR-bitcount operations

This work's method allows almost one order of magnitude reduction to the total energy consumption of 5 consecutive XNOR-bitcount operations, and almost a 78% reduction in total execution time.

VI. CONCLUSIONS

In this work, an optimized method to implement the XNOR-bitcount operation of BCNNs using compute-in-memory with STT-MRAMs was proposed. The STT-MRAM bitcells were carefully designed to allow optimal energy consumption and performance, as well as robust writing and reading operations. The XNOR-bitcount method presented in [1] was optimized algorithmically and at hardware-level. The hardware-level optimization of the memory array structure allowed a decrease in the storage density necessary to compute the XNOR-BC operation. This structure was used to implement both the original XNOR-BC method and the proposed algorithmically optimized method using STT-MRAM. The algorithmic optimization allowed a 30% reduction in execution time and a 26.1% reduction in energy consumption for a single XNOR-BC operation when using a 3x3 (9-bit) filter. Considering 5 sequential XNOR-BC operations with a single filter, the optimized method allowed a 78% decrease in execution time and a 85.1% reduction in energy consumption.

BIBLIOGRAPHY

[1] 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 Transactions on Electron Devices,* vol. 68, no. 10, pp. 4944-4950, Oct. 2021.

[2] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, "Binary Neural Networks: A Survey," *Pattern Recognition,* vol. 105, no. 107281, September 2020.

[3] A. Biswas and A. P. Chandrakasan, "Conv-RAM: An Energy-Efficient SRAM with Embedded Convolution Computation for Low-Power CNN-Based Machine Learning Applications," *2018 IEEE International Solid-State Circuits Conference (ISSCC),* 2018.

[4] M. Rastegari et al., "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," *Computer Vision – ECCV 2016,* vol. 9908, 2016.

[5] M. Courbariaux et al., "Binarized Neural Networks: Training Neural Networks with Weights and Activations Constrained to +1 or −1," 2016.

[6] O. Mutly et al., "Processing data where it makes sense: Enabling in-memory computation," *Microprocessors and Microsystems,* vol. 67, pp. 28-41, 2019.

[7] G. Santoro et al., "New Logic-In-Memory Paradigms: An Architectural and Technological Perspective," *Micromachines,* vol. 10, no. 368, 2019.

[8] P. Barla, V. Kumar and S. Bhat, "Spintronic devices: a promising alternative to CMOS devices," *Journal of Computational Electronics,* 2021.

[9] E. Garzón et al., "Ultralow Voltage FinFET-Versus TFET-Based STT-MRAM Cells for IoT Applications," *Electronics,* vol. 10, no. 15, p. 1756, 2021.

[10] X. Fong et al., "Spin-Transfer Torque Memories: Devices, Circuits, and Systems," *Proceedings of the IEEE*, vol. 104, no. 7, pp. 1449-1488, 2016.

[11] K. Asifuzzaman, R. Sánchez and P. Radojkovic, "Enabling a Reliable STT-MRAM Main Memory Simulation".

[12] H. Wang, W. Kang, L. Zhang, H. Zhang, B. K. Kaushik, and W. Zhao, "High-Density, Low-Power Voltage-Control Spin Orbit Torque Memory with Synchronous Two-Step Write and Symmetric Read Techniques," *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE),* pp. 1217-1222, 2020.

[13] N. Maciel et al., "Magnetic Tunnel Junction Applications," *Sensors*, vol. 20, no. 1, p. 121, 2020.

[14] G. Hu et al., "STT-MRAM with double magnetic tunnel junctions," *2015 IEEE IEDM*, p. pp. 26.3.1–26.3.4, 2015.

[15] E. Garzón et al., "Assessment of STT-MRAM performance at nanoscaled technology nodes using a device-to-memory simulation framework," *Microelectronic Engineering*, vol. 215, p. 111009, 2019..

[16] R. De Rose et al., "Compact modeling of perpendicular STT-MTJs with double reference layers," *IEEE Transaction on Nanotechnology*, vol. 18, p. 1063–1070, 2019.

[17] A. V. Khvalkovskiy et al., "Basic principles of STT-MRAM cell operation in memory arrays," *Journal of Physics D: Applied Physics*, vol. 46, no. 7, January 2013.

[18] Y. Pan et al., "A Multilevel Cell STT-MRAM-Based Computing In-Memory Accelerator for Binary Convolutional Neural Network," *IEEE Transactions on Magnetics*, vol. 54, no. 11, pp. 1-5, November 2018.

[19] S. Arcaro et al., "Integration of STT-MRAM model into CACTI simulator," *2014 9th International Design and Test Symposium (IDT),* pp. 67-72, 2014.

[20] R. Bishnoi et al., "Read Disturb Fault Detection in STT-MRAM," *2014 International Test Conference,* pp. 1-7, 2014.

[21] K. T. Quang, S. Ruocco and M. Alioto, "Boosted sensing for enhanced read stability in STT-MRAMs," *2016 IEEE International Symposium on Circuits and Systems (ISCAS),* pp. 1238-1241, August 2016.

[22] Z. Bian, X. Hong, Y. Guo, L. Naviner, W. Ge, and H. Cai, "Investigation of PVT-Aware STT-MRAM Sensing Circuits for Low-VDD Scenario," *Micromachines,* vol. 12, no. 5, 2021.

[23] Y. Wang, H. Tang, Y. Xie, et al., "An in-memory computing architecture based on two-dimensional semiconductors for multiply-accumulate operations," *Nature Communications,* vol. 12, no. 3347, June 2021.