PXNOR-BNN: In/With Spin-Orbit Torque MRAM Preset-XNOR Operation-Based Binary Neural Networks

Liang Chang, *Student Member, IEEE*, Xin Ma, Zhaohao Wang, *Member, IEEE*, Youguang Zhang, *Member, IEEE*, Yuan Xie, *Fellow, IEEE*, and Weisheng Zhao, *Fellow, IEEE*

***Abstract*— Convolution neural networks (CNNs) have demonstrated superior capability in computer vision, speech recognition, autonomous driving, and so forth, which are opening up an artificial intelligence (AI) era. However, conventional CNNs require significant matrix computation and memory usage leading to power and memory issues for mobile deployment and embedded chips. On the algorithm side, the emerging binary neural networks (BNNs) promise portable intelligence by replacing the costly massive floating-point compute-and-accumulate operations with lightweight bit-wise XNOR and popcount operations. On the hardware side, the computing-in-memory (CIM) architectures developed by the non-volatile memory (NVM) present outstanding performance regarding high speed and good power efficiency. In this paper, we propose an NVM-based CIM architecture employing a Preset-XNOR operation in/with the spin–orbit torque magnetic random access memory (SOT-MRAM) to accelerate the computation of BNNs (PXNOR-BNN). PXNOR-BNN performs the XNOR operation of BNNs inside the computing-buffer array with only slight modifications of the peripheral circuits. Based on the layer evaluation results, PXNOR-BNN can achieve similar performance compared with the read-based SOT-MRAM counterpart. Finally, the end-to-end estimation demonstrates 12.3× speedup compared with the baseline with 96.6-image/s/W throughput efficiency.**

***Index Terms*— Binary neural networks (BNNs), computing-in-memory (CIM), magnetic random access memory (MRAM), preset, spin–orbit torque (SOT).**

Manuscript received March 13, 2019; revised June 3, 2019; accepted June 29, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61704005, Grant 61627813, and Grant 61571023; and in part by the Program of Introducing Talents of Discipline to Universities under Grant B16001. *(Corresponding authors: Zhaohao Wang; Yuan Xie; Weisheng Zhao.)*

L. Chang and Y. Zhang are with the School of Electronic and Information Engineering, Fert Beijing Research Institute, Beihang University, Beijing 100191, China (e-mail: liang.chang@buaa.edu.cn; zyg@buaa.edu.cn).

X. Ma and Y. Xie are with the SEAL Laboratory, University of California at Santa Barbara, Santa Barbara, CA 93106 USA (e-mail: xinma@ucsb.edu; yuanxie@ucsb.edu).

Z. Wang and W. Zhao are with the Beijing Advanced Innovation Center for Big Data and Brain Computing (BDBC), School of Microelectronics, Fert Beijing Research Institute, Beihang University, Beijing 100191, China, and also with the Beihang-Georgtek Joint Microelectronics Institute, Qingdao Research Institute, Beihang University, Beijing 100191, China (e-mail: zhaohao.wang@buaa.edu.cn; weisheng.zhao@buaa.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2019.2926984

I. Introduction

DEEP convolution neural networks (CNNs) have attained great success in solving a wide range of practical tasks in academia and industry [1]–[3]. Specifically, there is an increasing focus on the deployment of CNNs in mobile phones and embedded chips for the general public. Unfortunately, these systems typically have limited computing power and memory space. One challenge to widespread deployments of CNNs is their considerable demand for computation and storage capacity due to their requirement of millions of floating-point parameters and operations.

Various efforts have been put on improving the operational efficiency of CNNs. For instance, researchers have used fixed-point or bit-wise operations on field-programmable gate array (FPGA) and application specific integrated circuit instead of using 32-bit floating-point operations on GPU [4], [5]. Mixed-precision computation has been used to reduce the bits of weights, activations, and gradients, simultaneously [6]. Binary neural networks (BNNs), with binary weights and activations (such as -1 and $+1$), have been among the most promising techniques to meet the desired computation and memory requirements [7], [8]. BNNs can achieve an accuracy comparable to full precision nets, which may be the key to efficient deep learning on mobile and embedded systems [5], [9]. In addition, a computational benefit achieved by using BNNs in inference is to replace float-point multiply-and-accumulate operations with bit-wise XNOR and popcount operations.

Potentially, such bit-wise operations further take advantage of the revival of the computing-in-memory (CIM) architecture by performing the logic operations locally in memory [10]–[12]. The CIM architecture reduces the data movement between the host processor and memory, which saves the bandwidth and power consumption. Recently, the CIM architectures developed using emerging non-volatile memory (NVM) technologies, such as magnetic random access memory (MRAM), resistive RAM (RRAM), and phase change memory (PCM), have obtained significant performance improvements regarding high speed and good power efficiency [11]–[14]. The NVM-based CIM supports several bit-wise operations based on the modified read circuit [12], [15]. However, several bit-wise operations

provided by the NVM-CIM are inefficient since complex peripheral circuits are involved. In addition, it is not practical to use the typical approach of modifying the read circuit and reference cell in MRAM since the read margin is small and a huge reference tree is generally used for the memory array [16], [17]. Furthermore, the CIM architecture with complex CMOS logic increases the cost due to the different optimization objective of the logic and memory [15]. Many efforts have been done to reduce using the complex CMOS logic [12], [18].

In this paper, we present a preset-based XNOR (PXNOR) operation using a single memory-cell of spin–orbit-torque MRAM (SOT-MRAM). SOT-MRAM is a promising memory technology, providing separate read/write paths, low latency, and low write energy, which is a good candidate for the write-based computation [19]–[22]. In the previous NVM-based CIM architecture, the XNOR operation is performed in the memory array using the read-based computing operation [15]. Typically, two sense amplifiers (SAs) are required to calculate an XNOR result with two or three clock cycles [15]. In addition, the final XNOR result requires an additional write operation to write-back or cache the intermediate result for the next operation. Specifically, the energy-efficient PXNOR operation employs the write operation to simultaneously implement both the computation and storage processes within the same memory cell. For writing the bit-cell of the SOT-MRAM, a write current, which is larger than the critical current (threshold current), is passing through the bit-cell to switch the state of the bit-cell. Based on the proposed PXNOR operation, we develop a computing-in-SOT-MRAM architecture for accelerating the computation of BNNs, namely PXNOR-BNN. In PXNOR-BNN, the write-based computation operation overlaps the buffering operation to improve energy efficiency. Then, the computing results are moved to the digital process unit (DPU) for the reduction operations, such as popcount, partial-sum, and merge operations. Our contributions are as follows.

1) We propose a write-based computing operation, which uses a single SOT-MRAM bit-cell to implement the XNOR operation.
2) Based on the bit-cell, we present a computing-buffer structure with a modified write-driver (WD) and first-in-first-out (FIFO)-based address decoder logic performing the PXNOR operation. This operation can overlap with the dynamic mapping operation of the BNNs.
3) With the computing-buffer, we develop a BNN accelerator (PXNOR-BNN) and a mapping method for accelerating the CIFAR-10 BNN model. Through an extremely efficient write scheme, we achieve overall advantages in energy and speed. The network-wise evaluation demonstrates 12.3× speedup compared with the baseline. The final throughput efficiency can reach up to 96.6 image/s/W.

The remainder of this paper is organized as follows. Section II provides the background of BNNs and SOT-MRAM. Section III discusses the circuits for the PXNOR operation. The PXNOR-BNN architecture, the memory array structure, and the design of the modules are introduced in Section IV.

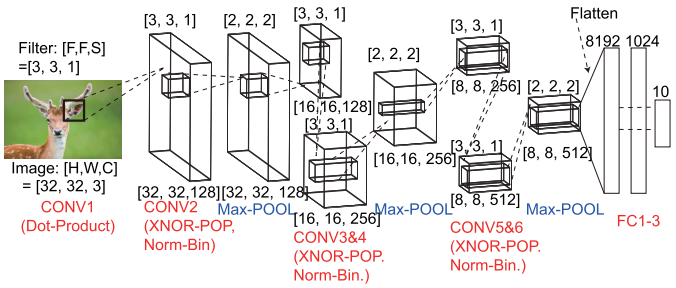


Fig. 1. Structure illustration of CIFAR-10 BNN, in which weights and activations are constrained to be the binary values of $\{-1, 1\}$. Parameters above the layer are the filter size, and the parameters under each layer are feature maps and the number of channels.

Afterward, we indicate the evaluation setup and results in Section V. Finally, Section VI concludes this paper.

II. Preliminary

This section provides the necessary information about the CNNs and BNNs. The CIFAR-10 BNN model is indicated with parameters of each layer. Also, we introduce the memory cell and basic operations of the SOT-MRAM.

A. CNNs and BNNs

1) CNNs: CNNs are constructed by stacking multiple computation layers, including convolution (CONV) layers, pooling (POOL) layers, and fully connected (FC) layers. The primary computation of CNNs occurs in the CONV layer, which is governed by

$$\boldsymbol{Y_{out}}(\boldsymbol{i},\ \boldsymbol{j},\ \boldsymbol{k}) = f\left(\sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{c=1}^{C}\boldsymbol{X_{in}}(i+m,\ j+n,\ c) * \boldsymbol{W}_{m,n,c} + \boldsymbol{b}\right) \quad (1)$$

where M, N, and C are the dimensions of the input feature map ($\boldsymbol{X_{in}}$), $\boldsymbol{Y_{out}}(x,\ y,\ z)$ is the output feature map, and $\boldsymbol{W}_{m,n,c}$ and $\boldsymbol{b}$ are the weight kernel and bias of kth output feature map, respectively. f represents the activation function, such as a rectified linear unit (ReLU). POOL layers map the input feature maps to output feature maps whose pixel is the max/mean of $F \times F$ pooling window. FC layers take an input vector of 1 feature maps and perform a dot product with a filter.

2) BNNs: On the image classification benchmarks such as CIFAR-10, BNNs have been demonstrated to achieve nearly state-of-the-art accuracy [9], [23]. BNNs quantize the weights and activations to be the binary values of $\{-1, 1\}$, as shown in Fig. 1. During BNNs training, a real-valued master weight space is saved, while its binary version is also generated through a binarization function, for all the computations in the forward and backward pass. A number of approaches, such as deterministic rounding or probabilistic sampling [8], as well as the optimization method [24], can realize the binarizing quantization. Although BNNs introduce some noises into its original CNNs model, many reports evidenced that the noises can be averaged out after many training epochs, and

sometimes, BNNs even perform better because of the intrinsic regularization. In this paper, we adopt the probabilistic sampling for weight binarization and deterministic rounding for activation binarization in [8] to build the BNNs model and obtain the binary data. As for the activation case, the normalization and binarization layers (Norm-Bin) are attached after the CONV layers [9], [11]. The computation of CONV can be expressed as

$$\boldsymbol{Y^b_{i,j,k}} = \mathrm{Norm} - \mathrm{Bin}\left(\mathrm{popcount}\left(\mathrm{XNOR}\left(\mathrm{W^b_{c,k,f}}, \mathrm{X^b_{m,n,c}}\right)\right)\right) \tag{2}$$

where $Y^b_{i,j,k}$, $W^b_{c,k,f}$, $and X^b_{m,n,c}$ are the layer output, binary weight, and binary input parameters, respectively. The deterministic method is expressed as

$$\boldsymbol{y^b_n} = \begin{cases} 1, & \text{if} x^b_n > 0 \\ -1, & \text{otherwise} \end{cases} \tag{3}$$

where x^b_n and y^b_n are the input and output of Norm-Bin layers, respectively. Because all the activations and weights are binary values, the multiply-and-accumulate operations can be simplified as bit-wise XNOR and popcount operations, whose details can be found in [8] and [24].

3) CIFAR-10 BNN Model: Based on the introduction of the BNNs mentioned earlier, the CIFAR-10 BNN model is discussed. Fig. 1 provides the structure of the CIFAR-10 BNN model, which consists of six CONV layers, three FC layers, and three POOL layers [9]. The CONV layer employs 3×3 filters and edge adding, and the batch normalization is used before binarization. The POOL layer uses the 2×2 filters executing the max-pooling operation. Finally, the output of the third Max-POOL layer is flattened to the FC-1 layer. For FC layers, the weight is 1-bit data, which occupies most of the weight parameters. We use an open-source code and pre-trained weight parameters to train the CIFAR-10 BNN model and achieve an 11.46% error rate [8], [9]. In PXNOR-BNN, we mainly focus on the feedforward process. The first CONV and last layers are processed by the host processor (i.e., CPU) under non-binary (8 bit) parameters refer to [9].

B. SOT-MRAM

SOT-MRAM is a high-speed and energy-efficient MRAM. It was developed to overcome the drawbacks of spin-transfer torque-MRAM (STT-MRAM) with high write latency and substantial write energy dissipation [17], [25]. Generally, the storage element of SOT-MRAM consists of a magnetic tunnel junction (MTJ) above a heavy metal (HM) strip, which is a three-terminal device as shown in Fig. 2(b), rather than a two-terminal device as shown in Fig. 2(a). The key structure of MTJ is a tunnel barrier sandwiched between a ferromagnetic fixed layer and a free layer. The relative magnetization directions of the free layer and the fixed layer can be programmed to a parallel (P) or anti-parallel (AP) state, resulting in low or high tunnel resistance, respectively.

For programming the SOT-MTJ, an in-plane current can be applied to the HM to generate the spin accumulation and hence induce the SOT through the spin Hall effect (SHE) in the HM layer, or the Rashba effect at the HM/FM/oxide interfaces, or both, which is still under debate [19], [20]. The SHE and Rashba effect are mainly represented by the damping-like torque and the field-like torque, respectively. The ratio of damping-like torque to field-like torque is dependent on the device structure, fabrication process, material type, and so on. There are two types of SOT-MTJ in terms of in-plane anisotropy MTJ (i-MTJ) and perpendicular-anisotropy MTJ (p-MTJ). In this paper, we employ the p-MTJ since its high write efficiency of the p-MTJ [17], [20].

For the p-MTJ shown in Fig. 2(b), the polarization orientation of the spin accumulation is vertical to the anisotropy axis of the FL so that a larger spin torque can be generated to obtain faster magnetization reversal than in the case of the conventional STT, as shown in Fig. 2(a). However, an additional magnetic field is required to achieve deterministic switching. The existence of this magnetic field limits the application of the SOT p-MTJ in NVMs and logic circuits. Some solutions have been proposed to avoid the use of the magnetic field. For example, the structural mirror asymmetry was developed to replace the magnetic field, but it requires a set of non-standard process to fabricate the wedge film [26]. Spin-Hall-assisted STT (SHA-STT) was proposed to eliminate the incubation delay of the STT to accelerate the magnetization switching in the absence of the magnetic field [27]. In recent experiments, researchers have replaced the magnetic field with an exchange bias induced by antiferromagnet/ferromagnet bilayer [28], [29]. This approach does not require a non-standard process and shows good compatibility with the existing fabrication technology of the MTJ. In addition, another work has demonstrated that the appropriate field-like torque can induce field-free deterministic SOT switching [30]. In this paper, based on the principles of [28]–[30], we propose that the PXNOR operation can be implemented with a three-terminal SOT-p-MTJ under the actions of an exchange bias and a relatively large field-like torque.

III. PXNOR Operation

We have developed a SPICE model of the three-terminal SOT-p-MTJ, which is indispensable for simulating and evaluating the performance of the designed BNNs before fabricating the real architecture [27], [31], [32]. Based on the developed model, we design the memory cell and the read and write circuits for the SOT-MRAM, as shown in Fig. 2(c). In the memory cell, two nMOS transistors connected to T3 and T2 terminals are used to control the read and write operations, respectively. For the read circuit, we employ the pre-charge SA (PC-SA) for identifying the "AP" and "P" states of the memory cell [33]. The reference cell is used for comparison with the memory cell, where the resistance value of the reference cell is normally $(R_{\mathrm{P}} + R_{\mathrm{AP}})/2$ or $(R_{\mathrm{P}} \times R_{\mathrm{AP}})^{1/2}$, where R_{P} and R_{AP} are the resistances of the MTJ at P and AP states, respectively.

The truth table of the typical XNOR and XOR operations is shown in Fig. 2(d) (top), in which A and B are the input signals, and output is the output signal. We can achieve the same operation by using the SOT-MTJ, which we call PXNOR

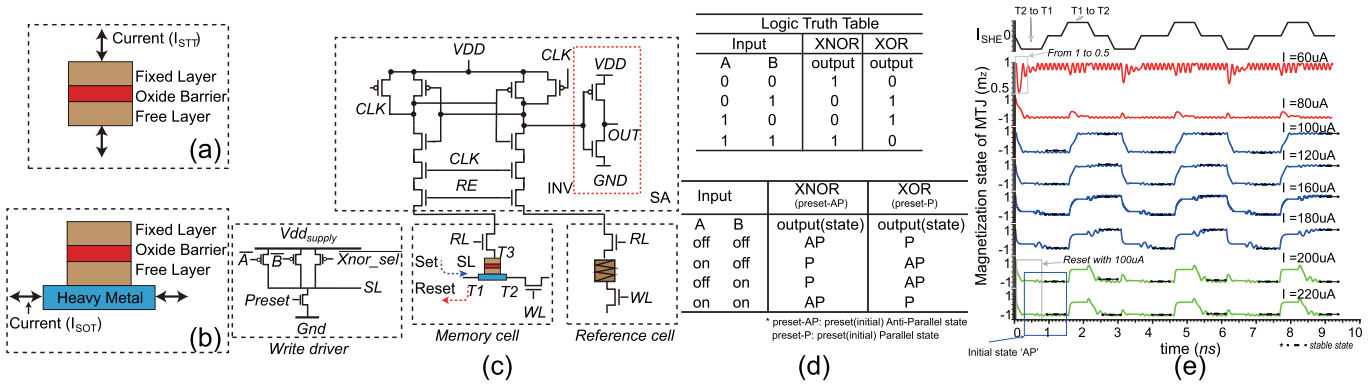


Fig. 2. Core element of SOT-MRAM and the PXNOR operation. (a) STT MTJ. (b) SOT MTJ. (c) Read and WD. The read circuit includes a PC-SA, an INV, and a reference cell. (d) Hardware mapping truth table from the logic to SOT-MTJ. A and B are the input signals, and output is the result depending on the values. ON and OFF present the activated and deactivated statuses of the control signal in WD. (e) Simulation results with an ideal current source applied to the HM of the SOT-MTJ in (c).

operation, as shown in Fig. 2(d) (bottom). Input signals A and B work as the control signals of the WD, as shown in Fig. 2(c). By changing the states of A and B, the current density flowing through the HM can be modulated to program the memory cell to different states. The output signal is represented by the output of the inverter (INV) for the SA, as shown in Fig. 2(c) (as well as the state of MTJ).[1] To implement the proposed PXNOR operation, two requirements need to be met, as mentioned in Section II-B: first, an exchange bias is induced through the antiferromagnet/ferromagnet structure; second, the ratio of field-like torque to damping-like torque is adjusted to be larger than 1 (1.4 in this paper). The detailed operation is explained as follows.

If the initial value of the device is AP state [i.e., z-component of the unit magnetization (m_z) is -1, as shown in Fig. 2(e)], the XNOR operation can be implemented: when both A and B are deactivated, the current flowing through the HM (i.e., I_{SOT}) is so small that the state of memory cell remains unchanged (i.e., 1 XNOR 1 = 1). If both of them are activated, I_{SOT} is excessively large to induce a huge SOT. In this case, the precession dominates the magnetization dynamics since the field-like torque is stronger than the damping-like torque in this paper. As a result, the memory cell returns to the initial state according to the principle of [30] (i.e., 0 XNOR 0 = 1). If only A or B is activated, an intermediate I_{SOT} could switch the memory cell to the opposite state (i.e., 1 XNOR 0 = 0), as expected by the classic SOT theory. On the contrary, if the initial state of the memory cell is P state, the XOR operation is obtained based on a similar principle. These results match the truth table, as shown in Fig. 2(d). As a result, we can obtain the XNOR and XOR operations by presetting the memory cell to different initial states. Therefore, we name it as PXNOR operation.

The above-mentioned PXNOR operation has been simulated and validated with our SPICE model. We employ an ideal current source, which can generate I_{SOT} with a tunable amplitude of 60 μA $\sim$ 220 μA and a duration of 0.5 $\sim$ 1.0 ns. The current is directly applied to the HM of a single memory cell. Simulation results show that the amplitude range of I_{SOT} for the successful switching is 82.6 $\sim$ 176.2 μA [the blue waveform indicated in Fig. 2(e)]. Outside this range, the switching fails, and thus, the memory cell remains in the initial state [as demonstrated by the red and green waveforms in Fig. 2(e)]. These simulation results are in agreement with the above-mentioned analysis. The proposed PXNOR operation can work at a wide range of field-like torque strength. In the following, we will use the proposed PXNOR operation to accelerate the computation of BNNs.

[1]Both the WD and the SA are located at the edge of the memory array for the bit-line (BL) in the array. Those peripheral circuits are the common modules of the memory.

IV. Overview of PXNOR-BNN Architecture

This section gives an overview of our proposed PXNOR-BNN architecture. The design of modules in terms of the memory array, the computing-buffer, and the interconnections of different memory arrays is introduced. After that, we present the control flow, computation pattern, operations, and the mapping method.

A. Accelerator Architecture

As shown in Fig. 3(a), the PXNOR-BNN architecture includes Chip, Bank, Sub-Bank, and Array levels. In the Chip level, the external interface can be used to communicate with the host processor, such as CPU or FPGA, for mapping data and configuring parameters. The global data buffer is used to store the intermediate weight parameters for configuring each Bank. In addition, the binarized output feature map parameters of the CONV1 are also stored inside the global data buffer before the running phase. An internal control logic is used to control the data communication between different Banks and global data buffer and monitor the status of each module. PXNOR-BNN can work standalone after configuring and writing back results to the global data buffer. During the running phase, the host processor can be used to handle other work.

The Bank is divided into Sub-Banks to improve the parallelism of the acceleration. One communication connection is

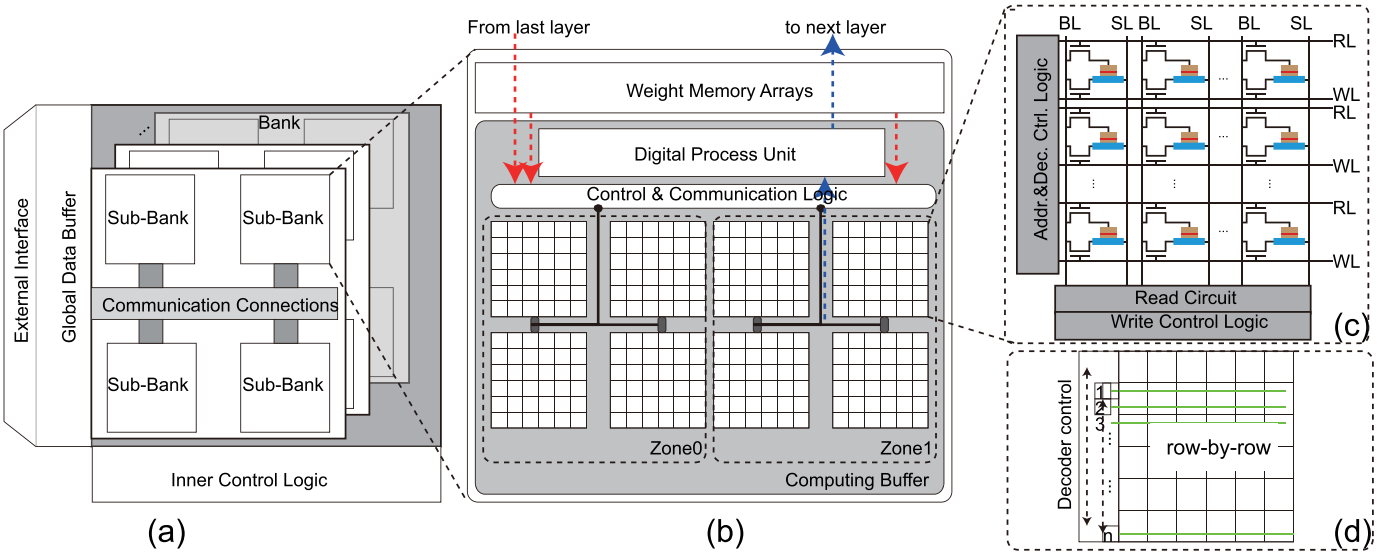


Fig. 3. PXNOR-BNN architecture. (a) Overview of the PXNOR-BNN architecture. (b) Sub-Bank architecture contains the SOT-MRAM memory array, buffers, control logic (Ctrl.), and DPU. (c) Memory array structure consists of BLs, SLs, RLs, WLs, and other peripheral circuits, such as address and decoder logic (Addr. and Dec.), read circuit, and write control logic. (d) Row-by-row mapping scheme and the FIFO-based decoder.

employed to support the communication among different Sub-Banks. Different Banks work in parallel and commit data to the global buffer under the control of the inner control logic.

The Sub-Bank contains weight memory arrays and computing-buffer, which performs the memory access and computing operations, respectively. The proposed PXNOR operation is implemented in the computing-buffer that is divided into two zones working in a "ping-pong" manner, as shown in Fig.3(b). The DPU is shared by the different zones for the reduction, partial-sum, and merge operations. The DPU can be customized according to the requirement of the acceleration.

Fig. 3(c) gives the structure of Array, including memory array, address and decoder logic, and write and read control logic, which provides the normal memory access and PXNOR operations. The read-line (RL) and write line (WL) control the read and write operations, respectively. The BL is shared by the read terminal and the write terminal. Both the source-line (SL) and BL are connected to the write control logic.

B. Control Flow and Computation Pattern

The PXNOR-BNN is a memory-centric computing architecture. The control flow includes two phases: off-line configuration phase and running phase. In the off-line configuration phase, the host processor configures the global buffer and inner control logic. Weight parameters are mapped into the weight buffer following the mapping method. The output activation parameters of CONV1 layer are programmed into the global buffer. In the running phase, the host processor sends a start tag to the PXNOR-BNN. The PXNOR-BNN begins to execute the computation. The layers of BNNs are executed in parallel, and the current outputs are fed to the next layer as inputs. The inputs of each layer are programmed into the corresponding memory array overlapping with the computing operations.

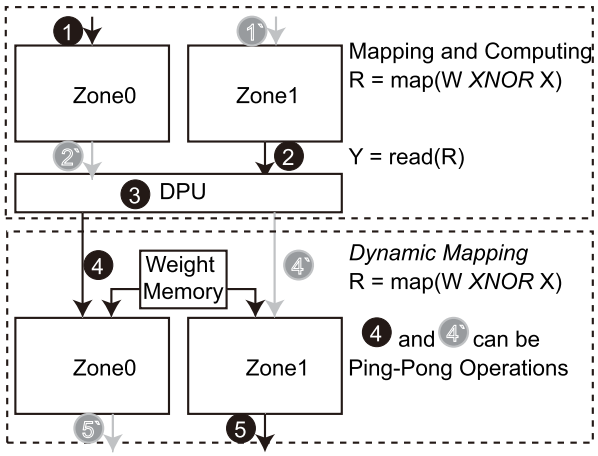


Fig. 4. Computation pattern of PXNOR-BNN. The mapping, read, DPU, dynamic mapping, and read operations form a streamed pipelined computation pattern.

A streamed pipeline computation pattern can be developed to improve the speed and energy efficiency. Fig. 4 gives the details of the computation pattern developed for the BNN acceleration. Zones can work in a "ping-pong" manner, which means that one zone is read for the DPU computing, while another zone can be map-computed to improve the parallelism. First, the mapping operation (①) is executed to write the XNOR results to one of the zones. In the meantime, the read operation (②) is performed to get the XNOR results of another zone for the computation inside the DPU. After the computation of DPU (③), the output data are dynamically mapped to the next Sub-Banks with the ready weight parameters (④). In each step, the two zones can work in parallel and execute different operations with an overlap between mapping operations and read operations. In addition, the same operation of two zones is performed in a "ping-pong" manner to develop a streamed computation pattern.

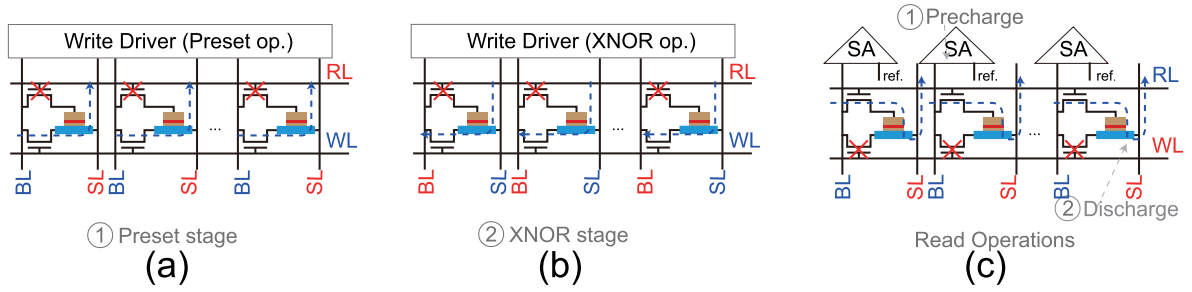


Fig. 5. Operations of the memory array. (a) Preset operation with a current flowing through the HM (from BL to SL), WL is activated, and RL is deactivated. (b) XNOR operation with the current flowing through the HM (from SL to BL), and RL is deactivated. (c) Read operation with the stable PC-SA, including two phases: first, pre-charge the SA; second, discharge to the memory-cell and reference cell (ref.).

C. Modules and Operations Design

In the PXNOR-BNN architecture, we modify the WD, address, and decoder control logic to support the PXNOR operation. In the following, we highlight those modified modules as well as introduce several necessary modules for organizing the PXNOR-BNN architecture.

1) Write Driver for Computing-Buffer: Fig. 2(c) reveals the control signals of the WD, including input signals A and B, the function control signal Xnor_sel, and Preset signals. The Preset signal is used to enable the nMOS transistor for the preset operation. Inputs A and B combined with Xnor_sel are used to control the pMOS transistor for providing the current flowing through the HM of the memory cell. If the Xnor_sel signal is deactivated, signals A and B are set to the same value to perform the standard write operation of the memory cell. In contrast, activating the Xnor_sel signal can enable the XNOR operation as the truth table [see Fig. 2(d)]. Compared with the read-based XNOR operation, which requires complex logic and multiple steps, our proposed XNOR operation is implemented with an SOT-induced write operation, resulting in a smaller latency and lower energy dissipation. Normally, the multiplexer is employed for the memory array to reduce area overhead of the SA and WD. In the computing-buffer, we remove the column multiplexer to enhance the buffer bandwidth and parallelism of the computing-buffer.

2) FIFO-Based Address and Decoder Control Logic: PXNOR-BNN performs operations in/with memory to reduce data movements. We map data to the computing-array row-by-row. This mapping method means that the previous random access address control logic is complicated and time-consuming for the row-by-row operation. Therefore, we employ an FIFO-based address and decoder control logic for the computing-array [34]. We can program the memory array row-by-row with the FIFO-based address scheme, as shown in Fig. 3(d). Note that the start point of the address and decoder control logic can be #1 or #n, which is decided by the mapping method and data placement.

3) Connection Consideration: Fig. 3(c) indicates the connection of the memory array. RL and WL are used to control the read and write operations, respectively. The BL is shared by two terminals of the memory-cell, which can remove an SL compared with the previous work [25]. SL is directly connected to the T2 [as shown in Fig. 2(a)] for providing write current to the memory cell. With this connection, we can observe that: during the XNOR operation (described below), the WD can provide enough write current without considering the source degeneration problem incurred by the nMOS transistor.[2]

4) Other Modules: We employ the PC-SA to read out the results of the memory array [33]. The proposed architecture also requires registers that serve for the configuration and programming. In addition, the DPU is a standard module used for the BNN accelerator, which provides the necessary function, such as pop-counter [11], [12].

5) Operations: Fig. 5 presents the detailed operations based on the above-mentioned modules. We indicate the details connections of each control line for the bit-cell.

1) *Preset Operation:* As shown in Fig. 5(a), first, the WL is activated, and RL is deactivated. Meanwhile, BL is connected to V_{dd}, and SL is grounded. After that, the write current flows through the HM to provide sufficient SOT to initialize the memory cells within the same row of the memory array. The preset operations of the computing buffers can be finished before the running phase of PXNOR-BNN and during the idle clock cycle of corresponding computing-arrays. Therefore, the cost of the preset operation can be ignored for the PXNOR operation.
2) XNOR *Operation:* As shown in Fig. 5(b), first, the WL is activated, while the RL is deactivated. BL is connected to GND or VSS. The programming current flows through the HM of the memory-cell through the SL and is tuned by the WD. The XNOR operation is the critical stage of the PXNOR operation since this stage performs the actual computation. In addition, the write operation of the SOT-MTJ is ultra-fast and energy-efficient.
3) *Read Operation* As shown in Fig. 5(c), in the pre-charge stage, the SA is pre-charged to V_{dd}. The control logic activates the selected RL of the selected row. In the discharge stage, the discharge current passes through the memory-cell from BL to the SL. After that, the SA

[2]In our designed memory cell, the write current flows through the HM from SL to BL. Thus, the source of the nMOS transistor is close to the ground, which means a large gate–source bias and an enough channel current. In contrast, if the current direction is reversed, the transistor channel cannot be sufficiently open due to a low gate–source bias, which is a source degeneration effect.

can identify the value of each memory-cell. In PXNOR-BNN, we perform the popcount, merge, and partial-sum operations with the DPU.

6) Discussion: Several advantages can be observed in the developed streamed-computation pattern. First, the write-based computing scheme improves the buffering efficiency through only minor modifications of the WD. In the conventional read-based computing solution, the buffering operation is required after the computing operation. Second, the streamed-computation pattern enhances the parallelism of the PXNOR-BNN by performing the "ping-pong" operation. Finally, we use the conventional SA and do not modify the reference cell of the read circuit, which improves the stability of the read operation. Moreover, a new computing scheme is provided and added to the CIM architecture rather than replacing the conventional read-based computing scheme. It is our future work to explore the possibility to combine the write-based with read-based computation patterns. In the proposed PXNOR-BNN, we manage to validate the function of the PXNOR operation and achieve an acceptable performance for the accelerator design.

D. Mapping Method

The mapping method is an essential factor that influences the performance of the PXNOR-BNN. In the mapping phase, the following factors should be considered: the size of the memory array and the static and dynamic mapping data [35]. First, the size of the memory array determines the number of memory arrays used for one layer. It also influences the parallelism of the Sub-Bank. Second, some of the mapping data will not be modified during the running phase, which we call static data, such as input parameters of the first layer and all weight parameters. However, some data are generated during the running phase and dynamically mapped to the destination memory array, which are called the dynamic data. We separate the mapping to configuration phase and running phase.

1) Configuration Phase: The weight parameters (static data) are configured into the weight memory array, and the input parameters of CONV2 are stored inside the global data buffer. All configuration information is programmed into each Bank and Sub-Bank for preparing the acceleration. These processes are completed under the off-line mode of the PXNOR-BNN.

2) Running Phase: After the configuration phase, the PXNOR-BNN works in the running phase. The XNOR results are generated and stored locally inside the computing-buffer row-by-row (dynamic data). The DPU performs the reduction operation by reading the intermediate data and dynamically maps the results into the computing-buffers.

For the detailed dynamic data mapping, we take the convolution layer-3 (CONV-3) of the CIFAR-10 BNN model as an example. The parameters of the layer are indicated in Fig. 6(a). Parameters I and O are the numbers of input channels and output channels, respectively. W is the weight parameters, in which $W1-1$ stands for the input channel 1 and output channel 1. $K \times K$ is the size of the filter that is 3×3 in the CIFAR-10 BNN model. Without considering the memory size, we unfold the input feature maps and weight parameters, as shown in Fig. 6(b). In Fig. 6(b), there are 128 (row) $\times$ 256 (column) weight parameters and 128 (row) $\times$ 196 (column) input parameters ($196 = (|I| - K + 1)^2$ for each input feature map). Each row of weights should iterate with input features and map the results into the memory array. After that, we consider to use a memory size of $W \times L$ to partition the entire parameter to the corresponding memory arrays. A fine partition and mapping method can also improve the parallelism of the memory array for the PXNOR-BNN [35]. The weight parameters can be duplicated to several Sub-Banks to further improve the parallelism. In addition, the dataflow model can be used to enhance the parallelism of the different memory arrays and Sub-Banks.

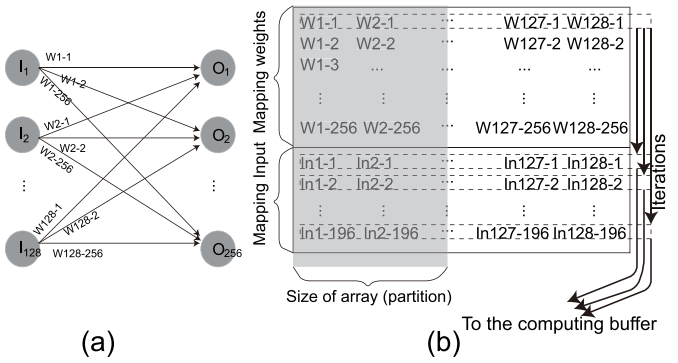


Fig. 6. Example of the mapping method for convolution layer-3 of the CIFAR-10 BNN model. (a) Parameters of convolution layer-3. (b) Mapping method for the computing buffer.

V. Experimental Results and Analyses

This section introduces the experimental setup and the validation results of the PXNOR operation. The evaluation results of different layers and network are demonstrated based on the proposed PXNOR-BNN architecture.

A. Experiment Setup

In the evaluation, we employ the SOT-MRAM to build the memory array. We develop a device-to-circuit framework to validate the functional behavior of the SOT-MTJ. In the device level, we employ a Verilog-A model of the SOT-MTJ to validate the PXNOR operation in terms of a single bit and memory row [27], [36]. According to the theories of [30], the field-like torque is enhanced in the Verilog-A model to enable the field-free ultra-fast switching. Based on this setting, the PXNOR operation could be achieved. It is important to mention that recent experiments have demonstrated the large field-like torque in appropriate device structures [37]–[39], which further validates the feasibility of our proposal. In the circuit level, we build the memory array with two-transistor-one-MTJ memory-cell and perform the SPICE simulation in Cadence virtuoso under a CMOS 28-nm technology. For validating the PXNOR operation, a wide range of current is added to the bit cell by adjusting the transistor width of the WD.

In the memory array and Bank levels, we employ a modified NVSim simulator to estimate area, latency, and energy [40].

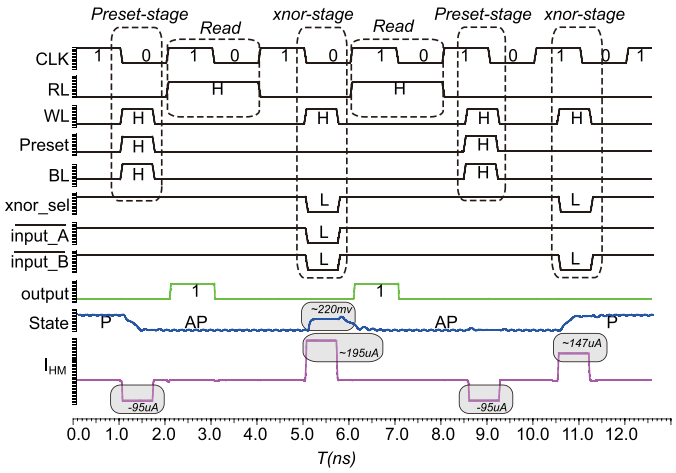


Fig. 7. Simulation results of a single memory-cell. Clock (CLK), RL for enabling read operation, word-line for enabling write operation, preset active signal (Preset), BL, the enable single of XNOR (xnor_en), and the WD control signals A and B. The invert output signal (output), the state signal of the MTJ (state), and I_{HM} the current added on the T1 terminal of Fig. 2(c).

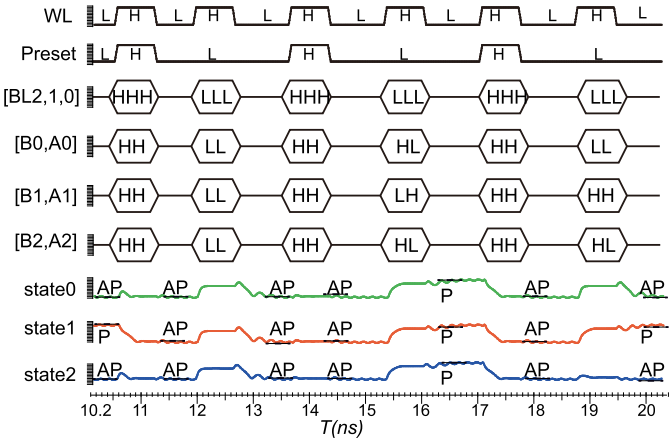


Fig. 8. Simulation results of the memory array. Three memory cells are selected to demonstrate the parallel XNOR operations with different configurations. BL2, BL1, and BL0 are the BLs for these three cells. [B0,A0], [B1,A1], and [B2,A2] are the control signals for the three cells. State0, state1, and state2 are the state signals for the memory cells.

The DPU and control logic are evaluated with the Synopsys Design Compiler under standard library. For the layer network evaluation, we develop an in-house simulator based on the NVSim simulator by combining our PXNOR-BNN architecture and the mapping method. We employ the pre-trained weight parameters and the open-source code to train the CIFAR-10 BNN model [9]. We can obtain 10.46% error rate with ∼1.5% accuracy degradation compared with the CNNs counterpart. The computation and performance models can be found in the previous works [9], [35], [41].

B. Functional Validation of PXNOR Operation

In Fig. 2(e), the PXNOR operation has been validated in the SPICE simulation by applying an ideal current source to the single SOT-MTJ. In this section, we use the WD [see Fig. 2(c)] and control logic, instead of the ideal current source, to perform the more comprehensive simulation. The corresponding simulation results of a single bit are demonstrated in Fig. 7, which is consistent with Fig. 2(e) and validates the PXNOR operation.

In Fig. 7, the pulsewidth of WL (and BL) is 0.7 ns with 0.05-ns signal edge, and the pulsewidth of RL should be 1 CLK cycle for the pre-charge and discharge stages. Both input_A and input_B are active-low to control the WD. Preset and xnor_sel control the preset operation and XNOR operation, respectively. During the preset-stage, the pulsewidth can be further decreased by using the recent achievement of the SOT-MTJ experiment [42]. Therefore, we can achieve an ultra-fast write operation thanks to the high write efficiency of SOT-MTJ. The signal output is the result of the SA after an INV. We probe the inner state of the SOT-MTJ with state signal, and I_{HM} denotes the current flowing through HM, as shown in Fig. 2(b). First, the preset operation programs the state of the MTJ from P to AP, and the read operation correctly senses the result to "1." In the first XNOR-stage, WL and xnor_sel are activated; meanwhile, both A and B are low activated. The instantaneous current can reach 195 μA. As a result, the state of the MTJ returns to AP. In the second XNOR-stage, only the input signal B is low activated so that the current is reduced to 95 μA. In this case, the MTJ is successfully switched to P state. To sum up, the simulation results of a single bit correctly implement the PXNOR operation.

We further validate the parallel PXNOR operations in a memory row. The simulation results are demonstrated (captured from 10.2 ns) in Fig. 8. The operation process, in order, includes preset operation, XNOR operation (with control signals $[B_x, A_y]$ = LL), preset operation, XNOR operation (with control signals $[B_x, A_y]$ = HLorLH), preset operation, and XNOR operation (random). The states of all three memory cells indicate that preset and XNOR operations are correctly performed.

The above-mentioned simulation results validate the PXNOR operation with both a single memory-cell and a memory row. In addition, the simulation results indicate ultra-fast operation speed, owing to high-efficiency SOT. Furthermore, using the FIFO-based address control logic and the simple WD, we can achieve ultra-low energy dissipation for the computation.

C. Layer-Wise Evaluation

The size of the computing-array for the computing-buffer influences the performance efficiency of the PXNOR-BNN. Two major factors are considered, which influences the performance of each CONV layer, including the parallelism and the utilization ratio of the computing-array. We perform the computation onto the different sizes of the computing-array, such as width W (256, 512) and length L(1024, 2048). The CONV2, CONV3, and CONV4 layers of the CIFAR-10 BNN model are write-computing data onto the different size of the computing-arrays. For CONV2, CONV3, and CONV4 layers, the numbers of input and output channels are *128* and *128*, *128* and *256*, and *256* and *256*, respectively. The corresponding sizes of input features are 32, 16, and 16, respectively. The

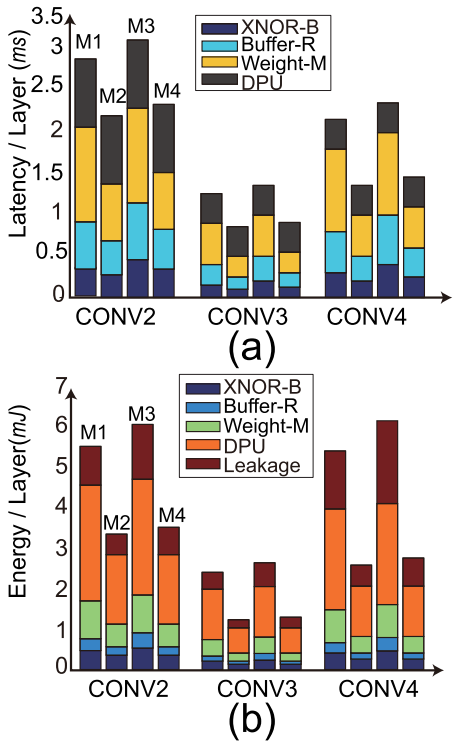


Fig. 9. Performance of the computation for the CONV2, CONV3, and CONV4 layers of the CIFAR-10 BNN model onto different sizes of computing-array. (a) Latency cost on CONV2, CONV3, and CONV4 layers for mapping and computing on different sizes of computing-array. (b) Energy dissipation per CONV layer for different sizes of computing-array. M1: array-1024 × 256, M2: array-1024 × 512, M3: array-2048 × 256, and M4: array-2048 × 512.

3 × 3 filter window is used to slide the input features as suggested in Fig. 1. Fig. 9 demonstrates the evaluation results in terms of latency and energy per layer.

Fig. 9(a) provides the latency evaluation results of CONV2, CONV3, and CONV4 layers. In each layer, the latency includes the latencies of XNOR-map, read, buffer, and DPU. The XNOR-map latency is caused by the data mapping and write-based computation. The read and buffer latency are the amount of time that the data are read, dynamic mapped, and intermediately stored, respectively. In addition, the DPU latency is the time spent in the popcount, move, merge, and partial-sum operations. The different size of memory array determines the number of memory arrays used for the computation, as shown in Fig. 6. More memory arrays increase the latency cost on each layer, in particular to the buffer operation and the move and merge operation of DPU. The computation using M2 (memory-1024 × 512) and M4 (memory-2048 × 512) costs smaller latency than using the M1 (memory-1024 × 256) and M3 (memory-2048 × 256) since the width (i.e., W) of the memory array determines the number of the memory arrays used for the computation according to the mapping method. We also notice that the latencies of the DPU and the buffer are considerably large, which becomes the bottleneck of the acceleration. The CIM operation (XNOR-map and read) could be faster thanks to the fact that the proposed PXNOR operation is performed inside a single memory-cell. It is beyond the scope of this paper to explore the powerful DPU and buffer, which will be our future work.

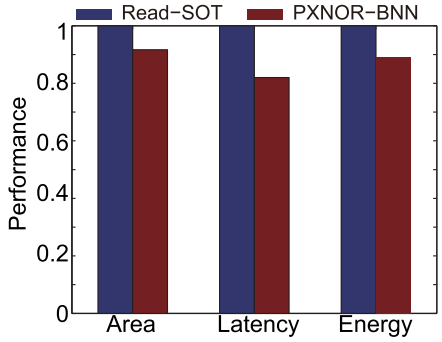


Fig. 10. Performance normalized to read-based in SOT-MRAM computation for the CONV3 and CONV4 layers.

Fig. 9(b) gives the energy evaluation results for those three layers. The computation using memory arrays M2 and M4 consumes less energy than M1 and M3 due to less memory array used for the computation. The width of the memory array is a critical factor influencing the energy efficiency and mapping partitions. The length of the memory array also affects the energy consumption for the CIM operations. Longer memory column increases the energy consumption on each access operation. In addition, the leakage current of the memory array and DPU may be the disadvantage to the energy-efficient CIM architecture. Despite the low leakage current of the SOT-MRAM, the control logic and digital logic still generate non-negligible leakage power consumption. Based on the above-mentioned analyses and the evaluation results, we select memory array M2 for the following evaluation.

D. Network-Wise Evaluation

Based on the streamed computation pattern, we compare the proposed PXNOR-BNN with the read-based computation pattern by performing the BNN computation [12]. First, we use the memory array size of 1024 × 512 for both the read-based and write-based computing memory arrays. In addition, PXNOR-BNN employs the same number of memory arrays for the read-based computation. Second, we consider the energy and latency cost on peripheral circuits, CIM operations, buffer, and DPU operations. We perform the computation of CONV3 and CONV4 layers from mapping parameters to the output dynamic mapping operations. Fig. 10 demonstrates the results regarding the area, latency, and energy. The PXNOR-BNN architecture slightly outperforms the read-based solution in the tree metrics in terms of area, latency, and energy consumption. For the area, we only make a minor modification on the WD and simplify the decoder and address logic, as shown in Fig. 3(c). In the read-based solution, both the read circuit and decoder logic are required to add more transistors, which results in large area overhead. For the latency and energy consumption, the PXNOR-BNN also achieves a similar performance compared with the read-based solution. In the read-based solution, both the weight parameters and input feature maps are required to be partitioned

TABLE I
PERFORMANCE OF PXNOR-BNN COMPARED WITH THE STATE-OF-THE-ART DESIGNS FOR THE CIFAR-10 BNN MODEL

	Execution time per image (ms)					
Item	**mGPU***	**CPU***	**GPU***	**FPGA*** [9]	**Read-SOT** [12]	**This work**
Conv1	-	0.68	0.01	1.13	0.68	0.68
Conv2-5	-	13.2	0.68	2.68	5.35	5.83
FC1-3	-	0.92	0.04	2.13	1.23	0.87
Total	90	14.8	0.73	5.94	7.26	7.31
Speedup	1×	6.1 ×	123×	15.1 ×	12.4 ×	12.3 ×
Power(W)	3.6	95	235	4.7	2.1	1.41
Throughput-efficiency (Image/sec/W)	3.09	0.71	5.83	35.8	68.1	96.6

* Results are sourced from data sheets and reference [9].
- a value we could not measure.

into several memory arrays. Each memory array performs the sequential operations since there is only one DPU inside each Sub-Bank. The “ping-pong" operations supported by PXNOR-BNN fully improve the parallelism and reduce the waiting time for the sequential model.

Based on the above-mentioned analyses, we estimate the network performance for the PXNOR-BNN, implementing the CIFAR-10 BNN model. Table I reveals the estimated results based on the simulator. We compared the performance of CONV1, CONV2-5, and FC layers with an Intel Xeon E5-2640 processor (CPU), an NVIDIA Tesla K40 GPU (GPU), the FPGA-based solution, and read-based computing SOT-MRAM for performing the CIFAR-10 BNN model [9], [12].[3] The NVIDIA Jetson TK1 embedded GPU board (mGPU) is also taken as the baseline for the comparison [9]. As demonstrated in Table I, PXNOR-BNN achieves 12.3× speedup compared with mGPU. Moreover, PXNOR-BNN outperforms other architectures on the power consumption thanks to in/with memory computing and overlap between the computation and buffering operations. Another benefit is that such a solution can obtain 96.6-image/s/W throughput efficiency, which is much higher than the CPU, GPU, and FPGA solutions. Finally, the proposed PXNOR-BNN achieves better throughput efficiency compared with the read-based solution for the SOT-MRAM thanks to the high-parallelism of the write-based computing architecture. These results indicate that PXNOR-BNN is a power-efficient architecture for the BNN model.

E. Discussion

The PXNOR-BNN architecture accelerates the computation of BNNs by the PXNOR operation. Two techniques help to improve the parallelism. First, the computing-buffer is capable of large parallel columns computing since all the data are inside the same Sub-Bank to reduce the data movement. Second, the “ping-pong" operations improve the parallelism by performing the write-computing and read-out data operation in parallel. Similar to the many-core processor, the parallelism is very important to the CIM architecture in the big data era [43].

[3]We only add the read-based computing SOT-MRAM, and the recent literature has provided the comparison between SOT-MRAM and RRAM solutions [12].

The parallelism of PXNOR can be further improved by using the dataflow model and building more computing pattern for different algorithms [35], [44].

VI. CONCLUSION AND FUTURE WORK

In order to maximize the execution efficiency of CNNs, we propose a PXNOR-BNN by combining the advanced quantization algorithm of BNNs and the emerging SOT-MRAM memory technology. In the proposed PXNOR-BNN, the complex floating-point multiply-and-accumulate operations are replaced with simple bit-wise XNOR and popcount operations, and the XNOR operation is implemented with a high-efficiency write-based solution instead of the conventional read-based solution. Significant improvement in the power consumption has been demonstrated by the cross-layer simulation. A study on the CIFAR-10 BNN model has demonstrated that the proposed architecture obtained a 12.3× speedup compared with the baseline mGPU board and 96.6-image/s/W throughput efficiency, which is much larger than the CPU, GPU, and FPGA solutions. Our PXNOR-BNN promises new application space for SOT-MRAM and powerful intelligence on portable devices thanks to the low-power consumption.

Our goal of this paper is to find a write-scheme-based solution for the in/with memory computing. The write-scheme is also suitable for the STT-MTJ to implement a bit-wise operation. However, the XNOR operation is one of the unique features for the SOT-MTJ. Many emerging technologies and devices can also support the write-scheme, such as the magic operation of RRAM and write-scheme for carbon nanotube [45]–[47]. Therefore, those devices and technologies can extend the application areas of this paper. With the write-based scheme and previous studies, we can seek more energy-efficient operations and control flow for the in/with memory computing, which is our future work. We believe that the PXNOR-BNN is just an opening act for the write-based in/with memory computing.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions.

REFERENCES

[1] S. Yu, S. Jia, and C. Xu, "Convolutional neural networks for hyperspectral image classification," *Neurocomputing*, vol. 219, pp. 88–98, Jan. 2017.

[2] J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. Schwartz, and J. Makhoul, "Fast and robust neural network joint models for statistical machine translation," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, 2014, pp. 1370–1380.

[3] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "DeepDriving: Learning affordance for direct perception in autonomous driving," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 2722–2730.

[4] K. Guo *et al.*, "Angel-Eye: A complete design flow for mapping CNN onto embedded FPGA," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 37, no. 1, pp. 35–47, Jan. 2018.

[5] S. Davidson *et al.*, "The celerity open-source 511-core RISC-V tiered accelerator fabric: Fast architectures and design methodologies for fast chips," *IEEE Micro*, vol. 38, no. 2, pp. 30–41, Mar./Apr. 2018.

[6] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2016, *arXiv:1606.06160*. [Online]. Available: https://arxiv.org/abs/1606.06160

[7] S. Zhu, X. Dong, and H. Su, "Binary ensemble neural network: More bits per network or more networks per bit?" 2018, *arXiv:1806.07550*. [Online]. Available: https://arxiv.org/abs/1806.07550

[8] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to $+1$ or -1," 2016, *arXiv:1602.02830*. [Online]. Available: https://arxiv.org/abs/1602.02830

[9] R. Zhao *et al.*, "Accelerating binarized convolutional neural networks with software-programmable FPGAS," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, 2017, pp. 15–24.

[10] B. Yan, F. Chen, Y. Zhang, C. Song, H. Li, and Y. Chen, "Exploring the opportunity of implementing neuromorphic computing systems with spintronic devices," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 109–112.

[11] T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang, "Binary convolutional neural network on RRAM," in *Proc. 22nd Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2017, pp. 782–787.

[12] S. Angizi, Z. He, and D. Fan, "PIMA-logic: A novel processing-in-memory architecture for highly flexible and energy-efficient logic computation," in *Proc. 55th Annu. Design Autom. Conf.*, 2018, p. 162.

[13] R. Mochida *et al.*, "A 4M synapses integrated analog ReRam based 66.5 TOPS/W neural-network processor with cell current controlled writing and flexible network architecture," in *Proc. IEEE Symp. VLSI Technol.*, Jun. 2018, pp. 175–176.

[14] S. R. Nandakumar, M. Le Gallo, I. Boybat, B. Rajendran, A. Sebastian, and E. Eleftheriou, "Mixed-precision training of deep neural networks using computational memory," 2017, *arXiv:1712.01192*. [Online]. Available: https://arxiv.org/abs/1712.01192

[15] S. Li, C. Xu, Q. Zou, J. Zhao, Y. Lu, and Y. Xie, "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in *Proc. 53rd Annu. Design Autom. Conf.*, Jun. 2016, pp. 1–6.

[16] K. Rho *et al.*, "A 4Gb LPDDR2 STT-MRAM with compact 9F2 1T1MTJ cell and hierarchical bitline architecture," in *IEEE Int. Solid-State Circuits Conf. ISSCC Dig. Tech. Papers*, Feb. 2017, pp. 396–397.

[17] L. Chang *et al.*, "PRESCOTT: Preset-based cross-point architecture for spin-orbit-torque magnetic random access memory," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2017, pp. 245–252.

[18] L. Chang, X. Ma, Z. Wang, Y. Zhang, W. Zhao, and Y. Xie, "CORN: In-buffer computing for binary neural network," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 384–389.

[19] Z. Wang *et al.*, "High-density NAND-like spin transfer torque memory with spin orbit torque erase operation," *IEEE Electron Device Lett.*, vol. 39, no. 3, pp. 343–346, Mar. 2018.

[20] M. Wang *et al.*, "Current-induced magnetization switching in atom-thick tungsten engineered perpendicular magnetic tunnel junctions with large tunnel magnetoresistance," *Nature Commun.*, vol. 9, p. 671, Feb. 2018.

[21] M. Wang *et al.*, "Field-free switching of a perpendicular magnetic tunnel junction through the interplay of spin–orbit and spin-transfer torques," *Nature Electron.*, vol. 1, no. 11, pp. 582–588, 2018.

[22] Z. Wang *et al.*, "Proposal of toggle spin torques magnetic RAM for ultrafast computing," *IEEE Electron Device Lett.*, vol. 40, no. 5, pp. 726–729, May 2019.

[23] X. Sun, S. Yin, X. Peng, R. Liu, J.-S. Seo, and S. Yu, "XNOR-RRAM: A scalable and parallel resistive synaptic architecture for binary neural networks," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 1423–1428.

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

[25] F. Oboril, R. Bishnoi, M. Ebrahimi, and M. B. Tahoori, "Evaluation of hybrid memory technologies using SOT-MRAM for on-chip cache hierarchy," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 3, pp. 367–380, Mar. 2015.

[26] G. Yu *et al.*, "Switching of perpendicular magnetization by spin–orbit torques in the absence of external magnetic fields," *Nature Nanotechnol.*, vol. 9, no. 7, pp. 548–554, 2014.

[27] Z. Wang, W. Zhao, E. Deng, J.-O. Klein, and C. Chappert, "Perpendicular-anisotropy magnetic tunnel junction switched by spin-Hall-assisted spin-transfer torque," *J. Phys. D, Appl. Phys.*, vol. 48, no. 6, pp. 065001-1–065001-7, Jan. 2015.

[28] S. Fukami, C. Zhang, S. DuttaGupta, A. Kurenkov, and H. Ohno, "Magnetization switching by spin–orbit torque in an antiferromagnet–ferromagnet bilayer system," *Nature Mater.*, vol. 15, no. 5, pp. 535–541, 2016.

[29] Y.-W. Oh *et al.*, "Field-free switching of perpendicular magnetization through spin–orbit torque in antiferromagnet/ferromagnet/oxide structures," *Nature Nanotechnol.*, vol. 11, no. 10, pp. 878–884, 2016.

[30] W. Legrand, R. Ramaswamy, R. Mishra, and H. Yang, "Coherent subnanosecond switching of perpendicular magnetization by the fieldlike spin-orbit torque without an external magnetic field," *Phys. Rev. Appl.*, vol. 3, Jun. 2015, Art. no. 064012.

[31] A. Jaiswal, X. Fong, and K. Roy, "Comprehensive scaling analysis of current induced switching in magnetic memories based on in-plane and perpendicular anisotropies," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 6, no. 2, pp. 120–133, Jun. 2016.

[32] M. Kazemi, G. E. Rowlands, E. Ipek, R. A. Buhrman, and E. G. Friedman, "Compact model for spin–orbit magnetic tunnel junctions," *IEEE Trans. Electron Devices*, vol. 63, no. 2, pp. 848–855, Feb. 2016.

[33] W. Zhao, C. Chappert, V. Javerliac, and J.-P. Noziere, "High speed, high stability and low power sensing amplifier for MTJ/CMOS hybrid logic circuits," *IEEE Trans. Magn.*, vol. 45, no. 10, pp. 3784–3787, Oct. 2009.

[34] H. Esmaeilzadeh, P. Saeedi, B. N. Araabi, C. Lucas, and S. M. Fakhraie, "Neural network stream processing core (NnSP) for embedded systems," in *Proc. ISCAS*, May 2006, p. 4.

[35] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," 2018, *arXiv:1807.07928*. [Online]. Available: https://arxiv.org/abs/1807.07928

[36] K. Jabeur, G. Di Pendina, G. Prenat, L. D. Buda-Prejbeanu, and B. Dieny, "Compact modeling of a magnetic tunnel junction based on spin orbit torque," *IEEE Trans. Magn.*, vol. 50, no. 7, pp. 1–8, Jul. 2014.

[37] J. Kim *et al.*, "Layer thickness dependence of the current-induced effective field vector in Ta|CoFeB|MgO," *Nature Mater.*, vol. 12, no. 3, pp. 240–245, 2013.

[38] M. Akyol *et al.*, "Effect of the oxide layer on current-induced spin-orbit torques in Hf|CoFeB|MgO and Hf|CoFeB|TaO_X structures," *Appl. Phys. Lett.*, vol. 106, no. 3, 2015, Art. no. 032406.

[39] Y. Ou, C.-F. Pai, S. Shi, D. C. Ralph, and R. A. Buhrman, "Origin of fieldlike spin-orbit torques in heavy metal/ferromagnet/oxide thin film heterostructures," *Phys. Rev. B, Condens. Matter*, vol. 94, no. 14, 2016, Art. no. 140414(R).

[40] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 7, pp. 994–1007, Jul. 2012.

[41] H. Kwon, A. Samajdar, and T. Krishna, "Maeri: Enabling flexible dataflow mapping over DNN accelerators via reconfigurable interconnects," in *Proc. 23rd Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, 2018, pp. 461–475.

[42] M. Cubukcu *et al.*, "Ultra-fast perpendicular spin–orbit torque MRAM," *IEEE Trans. Magn.*, vol. 54, no. 4, Apr. 2018, Art. no. 9300204.

[43] S. Angizi, J. Sun, W. Zhang, and D. Fan, "GraphS: A graph processing accelerator leveraging SOT-MRAM," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 378–383.

[44] L. Chang *et al.*, "DASM: Data-streaming-based computing in nonvolatile memory architecture for embedded system," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, to be published.

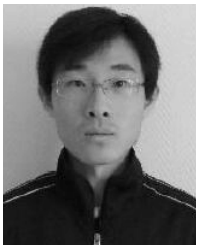
[45] H. Ji, L. Song, L. Jiang, H. H. Li, and Y. Chen, "ReCom: An efficient resistive accelerator for compressed deep neural networks," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 237–240.
[46] H.-S. P. Wong and D. Akinwande, *Carbon Nanotube and Graphene Device Physics*. Cambridge, U.K.: Cambridge Univ. Press, 2011.
[47] P. Junsangsri, J. Han, and F. Lombardi, "Design and comparative evaluation of a PCM-based CAM (content addressable memory) cell," *IEEE Trans. Nanotechnol.*, vol. 16, no. 2, pp. 359–363, Mar. 2017.

Liang Chang (S'15) received the B.S. degree from Chengdu Information and Technology University, Chengdu, China, in 2011, and the M.S. degree from Beihang University, Beijing, China, in 2014, where he is currently working toward the Ph.D. degree in spintronics at the Beijing Advanced Innovation Center for Big Data and Brain Computing (BDBC), Fert Beijing Institute, and at the School of Electronic Information and Engineering.

His current research interests include reconfigurable circuit design and advanced computer architectures based on emerging non-volatile devices.

Xin Ma received the B.S. degree in physics from the University of Science and Technology of China, Hefei, China, in 2009, and the Ph.D. degree in physics from The College of William and Mary, Williamsburg, VA, USA, in 2014.

He is currently a Postdoctoral Researcher with the Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA, USA. His current research interests include designing in-memory computing with emerging non-volatile memory technologies.

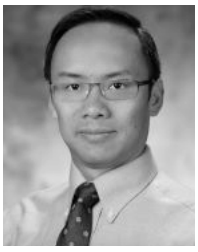


Zhaohao Wang (S'12–M'16) received the B.S. degree from Tianjin University, Tianjin, China, in 2009, the M.S. degree in microelectronics from Beihang University, Beijing, China, in 2012, and the Ph.D. degree in physics from the University of Paris-Sud, Orsay, France, in 2015.

He is currently an Assistant Professor with the School of Microelectronics, Beihang University. His current research interests include the modeling of emerging non-volatile nanodevices, the design of new non-volatile memories and logic circuits, and advanced computer architectures.

Youguang Zhang (M'13) received the M.S. degree in mathematics from Peking University, Beijing, China, in 1987, and the Ph.D. degree in communication and electronic systems from Beihang University, Beijing, in 1990.

He is currently a Professor with the Beijing Advanced Innovation Center for Big Data and Brain Computing (BDBC), Fert Beijing Institute, Beihang University, where he is also with the School of Electrical and Information Engineering. His current research interests include microelectronics and computer architectures.



Yuan Xie (M'02–SM'09–F'14) received the Ph.D. degree from the Electrical Engineering Department, Princeton University, Princeton, NJ, USA, in 2002.

From 2002 to 2003, he was with IBM, Armonk, NY, USA. From 2012 to 2013, he was with the AMD Research China Laboratory, Beijing, China. From 2003 to 2014, he was a Professor with Pennsylvania State University, State College, PA, USA. He is currently a Professor with the Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA, USA. His current research interests include computer architecture, electronic design automation, and VLSI design.

Dr. Xie served as the TPC Chair for HPCA 2018. He is currently the Editor-in-Chief of the *ACM Journal on Emerging Technologies in Computing Systems* (JETC), the Senior Associate Editor (SAE) of the *ACM Transactions on Design Automation for Electronics Systems* (TODAES), and an Associate Editor of the IEEE TRANSACTIONS ON COMPUTERS. He is an Expert in computer architecture who has been inducted to ISCA/MICRO/HPCA Hall of Fame.

Weisheng Zhao (M'07–SM'14–F'19) received the M.S. degree in electrical engineering from the ENSEEIHT Engineering School, Toulouse, France, in 2004, and the Ph.D. degree in physics from the University of Paris-Sud, Orsay, France, in 2007.

From 2008 to 2009, he was with the Embedded Computing Laboratory, CEA, Université Paris-Saclay, Saint-Aubin, France. From 2009 to 2013, he was a tenured Research Scientist with CNRS, Université Paris-Saclay. He is currently a Professor with the Beijing Advanced Innovation Center for Big Data and Brain Computing (BDBC), Fert Beijing Institute, Beihang University, Beijing, China, where he is also with the School of Microelectronics. He has authored or coauthored more than 120 scientific papers. His current research interests include the spintronics and its related device, circuit, and architecture investigations.

Dr. Zhao is an Associate Editor of the IEEE TRANSACTIONS ON NANOTECHNOLOGY and the *IET Electronics Letters*.