CORN: In-Buffer Computing for Binary Neural Network

Liang Chang[1,2], Xin Ma[2], Zhaohao Wang[1], Youguang Zhang[1], Weisheng Zhao[1] and Yuan Xie[2]
[1]Fert Beijing Research Institute, BDBC, School of Electronic and Information Engineering, Beihang University
[2]Electrical and Computer Engineering, University of California, Santa Barbara

***Abstract*—Binary Neural Networks (BNNs) have obtained great attention since they reduce memory usage and power consumption as well as achieve a satisfying recognition accuracy on Image Classification. In particular to the computation of BNNs, the multiply-accumulate operations of convolution-layer are replaced with the bit-wise operations (XNOR and pop-count). Such bit-wise operations are well suited for the hardware accelerator such as in-memory computing (IMC). However, an additional digital processing unit (DPU) is required for the pop-count operation, which induces considerable data movement between the Process Engines (PEs) and data buffers reducing the efficiency of the IMC. In this paper, we present a BNN computing accelerator, namely CORN, which consists of a Spin-Orbit-Torque Magnetic RAM (SOT-MRAM) based data buffer to perform the majority operation (to replace the pop-count process) with the SOT-MRAM-based IMC to accelerate the computing of BNNs. CORN can naturally implement the XNOR operation in the NVM memory array, and feed results to the computing data buffer for the majority write operation. Such a design removes the pop-counter implemented by the DPU and reduces data movement between the data buffer and the memory array. Based on the evaluation results, CORN achieves 61% and 14% power saving with $1.74\times$ and $2.12\times$ speedup, compared to the FPGA and DPU based IMC architecture, respectively.**

***Index Terms*—MRAM, Spin Orbit Torque, Binary Neural Networks, Write Operation, Preset-XNOR**

I. Introduction

Convolutional neural network (CNN) has become the state-of-the-art machine learning engine for image classification, object detection, text understanding and artificial intelligence [1] [2]. However, modern CNNs suffer from significant resource and energy overhead, due to their requirement of millions of floating-point parameters and operations. A promising solution is the Binary CNNs (BNNs) using approximate binary weights and activations, which significantly reduce the computations without sacrificing too much accuracy in classification [3] [4]. Specifically, simple exclusive-negated-OR (XNOR) and pop-count operations are used in BNNs, instead of complex multipliers and accumulator trees, resulting in a notable reduction of energy and area overhead [5].

Potentially, such bit-wise operations in BNNs can further benefit from recent achievements on in-memory computing architecture (IMC) [6] [7]. By performing the logic operations locally in memory, the off-chip data communication energy and latency can be remarkably saved. Moreover, recent IMC designs show performance improvement with the magnetic random access memory (MRAM) family (Spin-Transfer Torque or STT, Spin-Orbit Torque or SOT, -MRAM),

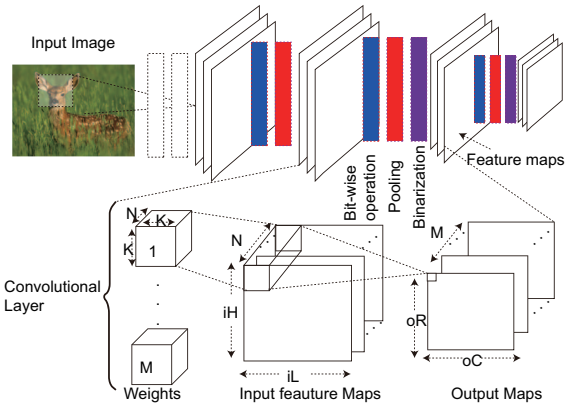


Fig. 1. The multiple layer of BNNs including convolution layer (CONV), pooling (POOL) and fully-connected (FC) layers.

which benefit from their high-density, low standby power, and resistance bit-cell [8]. However, challenges still exist when designing the MRAM-based IMC for the computing-intensive BNNs accelerator: firstly, a Digital Processing Unit (DPU) is used to implement the pop-counter, which increases the energy dissipation and data movement between the memory array and data buffer. Secondly, we should consider the data placement and mapping method between different memory arrays (as PE), which induce considerable programming energy.

In this paper, we propose a Non-Volatile Memory (NVM)-based BNNs accelerator, namely CORN, which uses the memory array of SOT-MRAM to implement the XNOR operation and to replace the pop-count operation of the BNNs model. The SOT-MRAM develops both the Process Engine (PE) and the data buffer to mitigate the data movement between the off-chip and host; in addition to reducing the data movement between different PEs. Our goal is to embrace the BNNs algorithm with the outstanding advantages of NVM-IMC architecture design. We note that our idea is also applicable to the typical STT-MRAM and other threshold resistance memory such as memristor [6], in which the write current and threshold will be much larger. Generally, SOT-MRAM can achieve good performance on the write operation owing to the high-efficiency of the SOT effect.

The rest of this paper is organized as follows: Section II provides an introduction to BNNs and the concept of in-memory computing and basics of SOT-MRAM. Section III

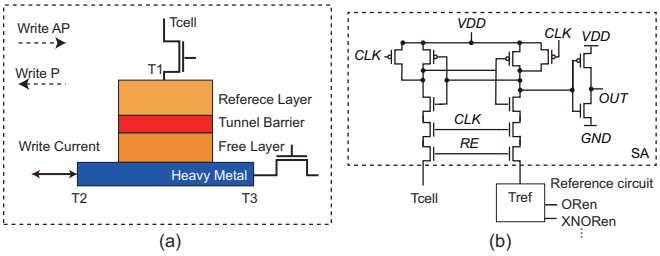


Fig. 2. The memory cell of SOT-MRAM (a) SOT-MTJ with two control transistors, T1, T2, T3 are three terminals of SOT-MTJ. The write current flowing through heavy metal (HM) to program the SOT-MTJ to anti-parallel (AP) and parallel (P), respectively. (b) The read circuit and reference circuit control logic [9], Tcell and Tref are connected to the bit-line (bit-cell) and reference circuit, respectively.

gives an overview of the CORN BNNs accelerator and the detailed architecture. Section IV describes the experimental setup and results. Finally, section V summarizes this paper.

II. Preliminaries

This section briefly reviews the Convolutional Neural Networks (CNNs) and Binary Neural Networks (BNNs) regarding layers, operations, and performance. Also, we introduce the basics of the SOT-MRAM and the concept of the NVM-based in-memory computing (IMC).

A. CNNs and BNNs

CNNs: Conventional CNNs are multi-layer networks, which typically provide the probabilities for a multi-channel image input to individual output classes [2] [10]. The CNNs consist of convolutional layers (CONV), pooling layers (POOL), and fully-connected layers (FC). Typically, the **CONV** layers convolve the input feature maps (ifMap) using a $K \times K$ weight-filter window to generate the output feature maps (ofMap), as shown by

$$\boldsymbol{y_n} = f(\boldsymbol{b} + \sum_{m=1}^{M} \boldsymbol{x_m} * \boldsymbol{W_{n,m}}). \quad (1)$$

The weight parameters of a CONV layer are $M \times N \times K \times K$. M and N are the number of ifMap and ofMap, respectively; $K \times K$ is the filter size. f represents the activation function, such as a rectified linear unit (ReLU). The **POOL** layers map the *ifMap* to *ofMap* whose pixel is the max/mean of a $L \times L$ pooling window. The dimension of *ofMap* of *POOL* is smaller than ifMap since pooling windows usually do not overlap with each other. The computation of **FC** layers is similar to that of the *CONV* layers but with 1×1 weight filter.

BNNs: BNN is an extreme case of CNN with binary constraints resulting in binary weights and activations [11] [3]. Fig. 1 shows the CONV layer of BNNs, which is similar to the typical CNNs; the input of each CONV is a 3D feature map with a size of $N \times iH \times iL$. The weight size is $N \times K \times K$ with stride S to filter the *ifmap* to produce a $M \times oR \times oC$ ofmap. In BNNs, both the weights and activations are constrained to a binary set to {-1, 1}. Therefore, we can use the bit-wise XNOR and pop-count operations to replace the complex multiplications in CONV, which significantly reduce the logic and memory resources in the hardware accelerator. Then, the normalization and binarization layers (Norm-Bin) are attached after the CONV layer [11] [6], which can be expressed as,

$$\boldsymbol{y} = \begin{cases} 1, if x > 0 \\ -1, otherwise \end{cases} \quad (2)$$

where the *x*, *y* are the input and output of Norm-Bin layer. The entire computing of the CONV layer can be defined as,

$$\boldsymbol{y_{nn}} = Norm - Bin(pop - Count(XNOR(w_b^{mm}, x_b^{mm}))). \quad (3)$$

Where $y_{nn}, w_b^{mm}, x_b^{mm}$ are the layer output, binary weight and binary input parameters, respectively.

CIFAR-10 BNNs model: CIFAR-10 BNNs model consists of six CONV layers, three FC, and three POOL layers, as shown in Fig. 3 (a). All CONV layers use 3×3 filters and edge padding, and the batch normalization is employed before binarization. Both input and weight parameters of the CONV layers are binarized to -1, 1 except for the first CONV layer where the input is the image. For FC layers, the weight is 1 bit, which occupies most of the weight parameters. All POOL layers employ the max-pooling, the weight is 2×2 without filter overlapping. We use open-source code to train the CIFAR-10 BNN model and achieve an 11.46% error rate [4]. In this paper we mainly focus on accelerating the inference process; the detailed parameters of CIFAR-10 can be found in [12].

B. SOT-MRAM

SOT-MRAM utilizes the effects of SOT to write the memory cell. Similar to other MRAMs, the memory bit value is stored in the magnetic tunneling junction (MTJ) with a tunnel barrier (TB) sandwiched between a ferromagnetic reference layer (RL) and a free layer (FL) above HM. The "0" or "1" values are represented with the low or high tunneling magneto-resistance, which is governed by the Parallel (P) or anti-parallel (AP) magnetization alignments between the RL and FL. An electrical current passing through the HM layer can effectively switch the magnetization of the FL as well as the bit value, utilizing the SOT generated by spin Hall effect (SHE) or Rashaba effect [13]. A similar memory design is the STT-MRAM, where the write current passes through the MTJ and uses the STT effect to switch the magnetization (P or AP). In comparison, while the memory cell of SOT-MRAM needs one additional transistor (three-terminals) compared with that of STT-MRAM (two-terminals) as shown in Fig. 2 (a), the SOT-MRAM is more efficient in write operation [14] [15]. To date, the electrical model is used to evaluate the performance of SOT-MRAM before fabricating a real device and chip via the circuit simulation [16] [17]. In this paper, we employ the device model with the SPICE electrical netlist to develop the memory cell and memory array according to parameters from recent literature [14] [15].

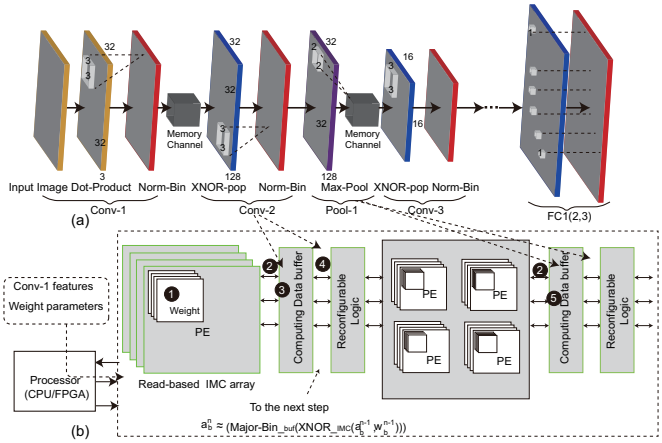


Fig. 3. Overview of CORN architecture. (a) The CIFAR-10 BNNs model with XNOR, Norm-Bin, and pop-count operation. (b) The architecture and control flow of CORN.

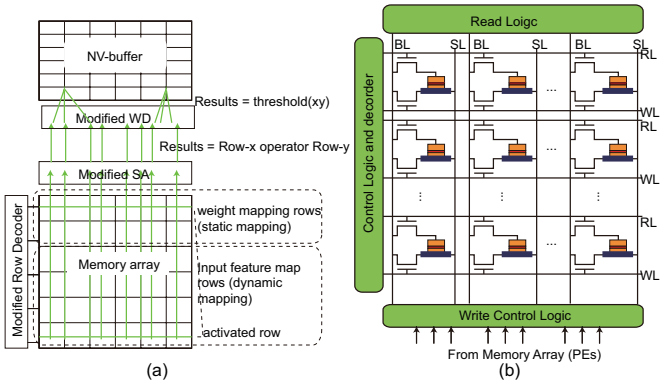


Fig. 4. The data communication between PEs and NV-buffer. (a) The data communication and data mapping for the memory array (as PEs). (b) The memory and data structure developed by the SOT-MRAM. In the PEs, we modify the read logic and decoder. In the buffer, we modify the write control logic. Glossary: bit-line (BL), read line (RL), source line (SL) and word line (WL).

C. NVM-based In-memory Computing

The concept of In-Memory Computing (IMC) attracts ever-increasing research interests, starting from the 1990s with the proposed Processing-in-Memory [18]. Here, we define the IMC the same as [2] where the computing can be completed inside the memory array with very few control-circuit modifications. For the NVM-based IMC, the bit-wise operation can be obtained with the memory array architecture by configuring the reference circuit [7]. In the memory array, two or more word lines are activated (on), and the Sense Amplifier (SA) is used to sense the value of the bit line as shown in Fig. 2 (b). The different configuration of the SA controls the value of the reference cell to decide the output of the SA. The NVM-based IMC can provide several advantages to the BNNs acceleration. First, the computation can efficiently execute in the memory array reducing the data communication between the host and off-chip memory with minor modification (SA and decoder).

Moreover, NVM technology with ultra-low leakage current can significantly reduce the leakage power of the accelerator thus improving the power efficiency [8]. However, the NVM memory array has difficulty in implementing the pop-count operation of the CONV layer. Therefore, a DPU should be employed to perform the pop-count operation. The DPU is both an area-consuming and a power-consuming digital circuit degrading the performance of IMC. In this paper, we observe that the pop-count process can be replaced with a majority write operation to remove the DPU of the pop-count operation.

III. Architecture Design

This section provides an overview of CORN architecture and the write-based in-buffer computing engine. Also, the write-driver (WD) for the buffer, data communication between PEs and buffer, and mapping method are discussed.

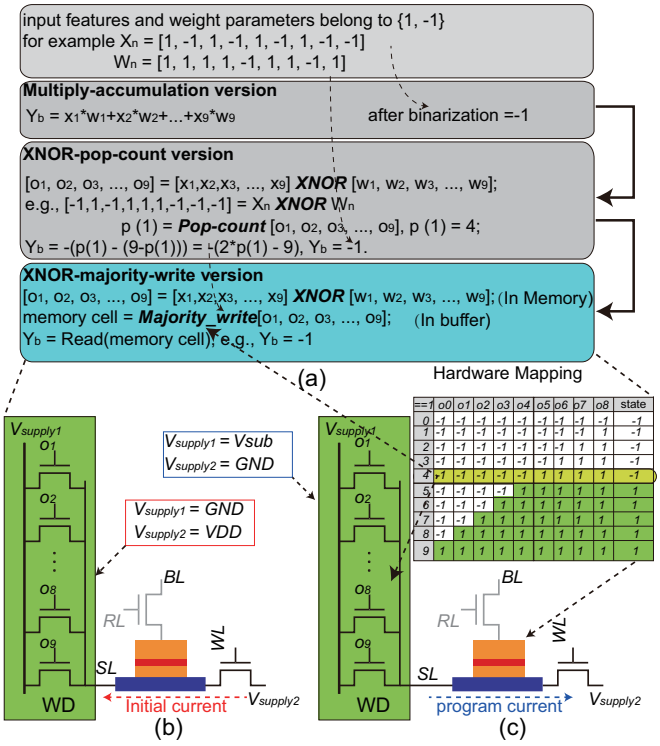
A. Overview of the Architecture

Fig. 3 reveals the architecture of CORN containing several computations: the first CONV layer to abstract the input image (yellow color), the binary CONV layer with XNOR and pop-count operations (XNOR-pop) followed by Normalization and Binarization (Norm-Bin) for the other CONV and FC layers, and the binary max-pooling operation. The first CONV layer is a fixed-point convolutional kernel that is handled by the **host processor** such as a Center Process Unit (CPU) or Field Program Gate Array (FPGA) (classification accuracy loss of $< 0.5\%$). All of the binarized CONV layers are accelerated in the CORN architecture. **PEs** accelerate the XNOR operation using the memory read operation with activating XNORen (Fig. 2) (b). **Computing data buffer** achieves the raw data (XNOR results), and executes the pop-count (typically) and binarization operations simultaneously, which we use as a majority write operation for the data-buffer controlled by the write control logic shown in Fig. 4 (b). An **NVM-based reconfigurable logic** provides the necessary function for the CONV layer such as Norm-Bin operations. The computing data buffer can also execute the max-pooling operation to remove some DPU resources. Therefore, the capacity of the reconfigurable logic is much smaller than that of the DPU in the previous literature [7].

B. XNOR-Majority-Write to Replace XNOR-pop-count

Fig. 5 illustrates the write-based majority operation and modification of the write driver (WD).

1) Operation replacement: Firstly, we analyze the multiply-accumulation and XNOR-pop-count operation for the binary parameters (9 bits in our experiment, which is suitable for the CIFAR-10 model). In the multiply-accumulation version, the final result is *-1* after multiple-addition and binarization as indicated in Fig. 5 (a). Similarly, the XNOR-pop-count version



==1	o0	o1	o2	o3	o4	o5	o6	o7	o8	state
0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
2	-1	-1	-1	-1	-1	-1	-1	1	1	-1
3	-1	-1	-1	-1	-1	-1	1	1	1	-1
4	-1	-1	-1	-1	-1	1	1	1	1	-1
5	-1	-1	-1	-1	1	1	1	1	1	1
6	-1	-1	-1	1	1	1	1	1	1	1
7	-1	-1	1	1	1	1	1	1	1	1
8	-1	1	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1

Fig. 5. The majority operation process and hardware implementation. (a) From multiply-accumulation operation; the pop-count operation to the write-based majority operation. (b) The PRESET stage of the write-based majority operation. (c) The program-stage of the write-based majority operation with the truth table (consider 0 as -1).

can achieve the same result (*-1*) taking the same x_n and w_n [1].

Secondly, we use a majority-write operation to replace the pop-count operation. Fig. 5 (a) reveals the proposed XNOR-majority-write version of computation, where the output *-1* is obtained since *-1* is the majority of the XNOR results (e.g. [-1, 1, -1, 1, 1, 1, -1, -1, -1]). The benefit of the *majority-process* is that we can replace the DPU with the write-based in-buffer computing which is more energy-efficient, and reduce the data communication.

2) Hardware mapping for majority-write operation: In CORN, we use CIM to accelerate the *XNOR* operation (In Memory). After that, the buffer is used to implement the *Majorit-write* operation. Fig. 5 provides the hardware mapping and the control circuit for the majority-write operation.

The SOT-MTJ is a current-density threshold device, which means the memory cell can be programmed (the state from P to AP or AP to P) only by providing enough current (via WD) [2]. For a fixed width of the transistor, more transistors can provide more current to the source line. We use nine control transistors (we use NMOS rather than PMOS) for the WD; the output of the PE controls each NMOS transistor (in computing mode). **PRESET mode:** Firstly, We **PRESET** the SOT-MTJ to *P* state (low resistance of SOT-MTJ or binary -1) with an initial current from terminal T3 to T2 of the SOT-MTJ (as shown in Fig. 2 (a)). **PROGRAM mode:** Then, the $V_{supply1}$ is down to *Vsub*, and several NMOS transistors are activated to provide current for the HM as shown in Fig. 5 (c). The state of the SOT-MTJ can be programmed (from P to AP state) only when more than five NMOS transistors are activated. This writing process matches the truth table described in Fig. 5 (c) and the number of "-1" presents deactivation of the NMOS transistor (can also be considered as 0; in this paper we assume it is -1.). With the proposed WD and write scheme, we can develop the XNOR-read-Majority-write operation for the computation of BNNs with faster speed and lower energy dissipation as shown in Fig. 4 (a).

Threshold operation for the max pooling: The threshold write operation (of the buffer) also can be used to implement the max-pooling operation (2*2 to 1 bit). For instance, a max-pooling window consists of x = {x1, x2, x3, x4}, if only all elements of x equal to -1, the $max - pool(x) = -1$, and $max - pool(x) = 1$ if x contains +1 element (e.g, $max - pool[1, -1, 1, -1] = 1$). We use the same WD as shown in Fig. 5 (b) and (c); we set the [o1, o2, o3, o4, o5] = [11110] and four bits of the max-pooling window connected to other four bits of the WD. Therefore, the SOT-MTJ can be successfully written only when a "1" exists in the max-pooling window. In this process, we can reduce the comparator developed by the DPU.

C. Control Flow and Data Mapping

Fig. 3 (b) discusses the control flow of the CORN architecture. In *Step1*, we program weight parameters into the designated PEs before the execution to reduce the communication between the host processor and CORN architecture (①). The host processor is also used to compute the CONV-1 layer and map its results to corresponding PEs. Therefore, the host processor is ideal or can be used to compute other input images during the acceleration. In *Step2*, PEs of CONV2 start to compute, and a read operation is used to get the XNOR results (②) following a write operation of the computing data buffer to obtain the majority-binarized result (③). Afterward, the reconfigurable logic achieves the data from the computing data buffer ④) and maps the data into PEs for the next layer. Another read operation is executed, and the data is transferred to the computing data buffer for the max-pooling (⑤) by the threshold write operation. At the end of *Step2*, the final results are written to the final buffer (such as a global buffer).

We consider using both the weight-stationary (reuse weight parameters in the memory array or PEs) and output-stationary (partial results locate in buffers) in the CORN architecture to remove the weight mapping and merge operations [10]. Fig. 4 (a) shows the process of the data mapping and computing between the memory array and NV-buffer. As aforementioned,

[1]The result can be verified via a truth table. The source code and the process of the computation in the link: https://github.com/itayhubara/BinaryNet.tf and https://github.com/MatthieuCourbariaux/BinaryNet.

[2]Of course, the thermal effect influences the SOT-MTJ, but the threshold current is still vital for writing the SOT-MRAM. Thermal effect influence is not the primary concern in this paper. Also, we can replace the SOT-MTJ with other threshold devices such as RRAM.

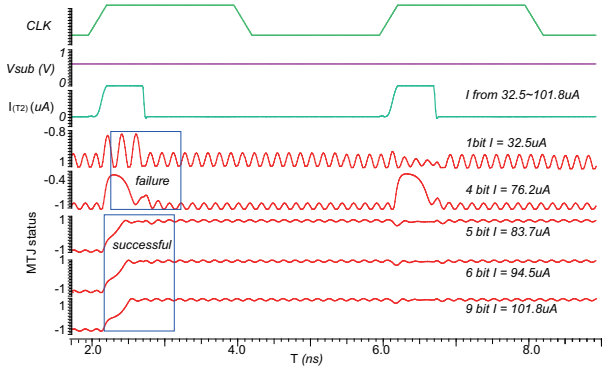


Fig. 6. Validation of the WD and the single bit-cell. *CLK*, *Vsub*, *and I* are the clock signals, the voltage provided in the programming stage (lower than VDD), and the current added on the HM of SOT-MTJ, respectively. MTJ status: AP (-1) and P (1).

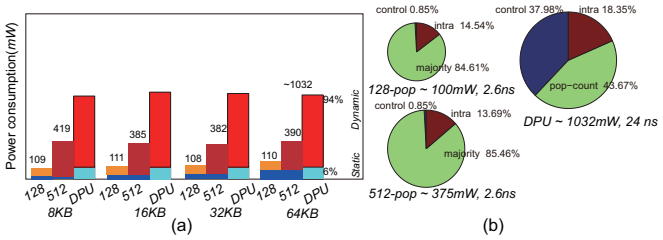


Fig. 7. The power consumption for the buffer with majority operation and the DPU with the pop-count operation. (a)The power consumption of different size data buffers. (b) The breakdown of dynamic power consumption.

weight parameters are programmed into each memory array in advance (static mapping) and the input features map to memory array according to the data mapping method (dynamic mapping). The modified SA is used to read two activated memory rows (horizontal green line) and transfer raw data (XNOR results) to the write driver (WD) of the NV-buffer that is developed by the SOT-MRAM array shown in Fig.4 (b). The WD executes the majority-write operation which is used to replace the pop-count and multiple operations by the DPU.

IV. Experiment and Results

This section provides the experimental setup and validation results of the threshold based write operation. Also, we demonstrate the results of the efficiency of the XNOR-majority operation and performance of different layers for the CIFAR-10 with the CORN architecture.

A. Experimental Setup

We validate the behavior of the majority-write operation and WD in the Cadence virtuoso (with Spectre) under STMicroelectronics CMOS 28 nm technology [19] [15]. A modified NVSim simulator obtains parameters of the memory array and data buffer [20]. The reconfigurable and control logic are evaluated by the Xilinx's Vivado and Synopsys's design compiler. The base-line and parameters of BNNs (source code) are from [12]. We write a C++ based simulator to evaluate the layer performance based on the mapping method and computing model [10]. The size of the PE is 1024×512.

B. Validation of the Majority-write Operation

Fig. 6 illustrates the simulation result based on the circuit shown in Fig. 5 (b) and (c). A *PRESET* operation programs the SOT-MTJ to "-1" (P state). Firstly, only O_1 is activated to provide 32.5 *uA* which is under the critical current. Secondly, four NMOS transistors are activated to provide 76.2 *(*uA) which still cannot modify the state of the SOT-MTJ. When five NMOS transistors (and more) are activated, the state of the SOT-MTJ is modified from P to AP state (AP represent 1 in this paper). Both the latency and energy of this process are much lower than the pop-count process. Note that, the size of the NMOS transistor is 3F (F presents the feature size of the technology) under $V_{supply2} = 0.5V$ voltage.

C. Data Buffer Evaluation

For a 512 width memory row, we require 56 (512/9) 9-bit majority WD for the proposed NV-buffer. However, 512 bits width pop-counter (9-in-1-out) is considerably large as evaluated by the previous literature, which requires 376 look-up-tables (LUTs) and 29 flip-flops (FFs) (even worse using the bipolar-accumulate in which this number is 759 LUTs and 84 FFs) [21]. We build 128-bit and 512-bit width data buffers including the sizes of 8 KB, 16 KB, 32 KB, and 64 KB compared to the DPU counterpart. Fig. 7 demonstrates the comparison results, the power per operation of 128-bit width data buffer significantly outperforms that of the DPU based pop-counter design. Firstly, the static power is lower since the NVM technology is used to perform computing. Secondly, the dynamic power of the write-based majority solution is much lower than the DPU, thanks to the efficient write operation and data communication. Besides, results of the majority operation are directly inside the buffer, while the DPU's results should execute another write operation to program into the buffer. Even with a 512 width NV-buffer, our proposed majority solution is still much better than the pop-count solution. Also, the in-data-buffer based computing is power-efficiency since computation consumes more than 80% of the dynamic power, while the control logic and intra-communication of DPU based pop-counter consume a large proportion of the dynamic power as shown in Fig. 7 (b).

D. Layer-wise Estimation

We employ the 512-bit width data buffer to estimate the performance of CONV layer following the CORN's control flow and mapping method. We compare the performance of CORN to the state-of-the-art FPGA design for the CAFIR-10 BNN moel [12]. Besides, a DPU based IMC design is developed as the comparison following the method demonstrated in [7] (We add the mapping latency and energy between different layers). We choose the CPU as the host processor and show the result in table I. The CORN architecture outperforms the FPGA and IMC-DPU design both in power consumption and throughput thanks to the majority and threshold max-pooling

TABLE I
THE LAYER-WISE PERFORMANCE ESTIMATION OF CORN COMPARED TO THE STATE-OF-THE-ART DESIGN FOR CARFIA-10 BNN MODEL.

	Execution time per image (ms)		
Item	**FPGA** [12]	**IMC-DPU** [7]	**CORN(this work)**
Conv1	1.13	0.68	0.68
Conv2-5	2.68	5.35	2.21
FC1-3	2.13	1.23	0.89
Total	5.94	7.26	3.42
Power(*W*)	4.7	2.1	1.83
Throughput *(Image/sec/W)*	35.8	68.1	159.4

operations. The estimate power saving of CORN is 61% and 14% with $1.74\times$ and $2.12\times$ speedup, compared to the FPGA and IMC-DPU, respectively. As a result, CORN can achieve much higher throughput (Image/second/Watt) than that of the other two designs.

V. CONCLUSION

BNNs can reduce the memory usage and multiple-accumulate computing of CNNs without sacrificing much of the accuracy, which provides a possibility for the IMC with simple logic operations. In this paper, we proposed an IMC based computing accelerator, namely CORN, which performs the XNOR operation in the memory-based PEs (read operation); in addition, to replace the pop-count process with the efficient majority process with an NV-buffer (threshold write operation). Overall, based on the estimation, CORN achieves better performance regarding power consumption and throughput, compared to the FPGA and IMC-DPU counterpart. To date, we only support the 9-bit filter window in the CORN, which is suitable for some BNNs model. If we use a much higher threshold device such as RRAM (e.g., 25 bits, 49bit), the idea of CORN is also suitable for other networks.

ACKNOWLEDGEMENT

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions. Thanks to Lolly Contreras for helping improve the paper. L. Chang, Z. Wang, Y. Zhang and W. Zhao are supported by the National Natural Science Foundation of China under Grant No.: 61571023, 61501013 and 61627813. X. Ma and Y. Xie are supported by NSF grant 1730309, 1725447, 1719160, 1500848, CCF 1740352 and SRCnCORE NC-2766-A.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] A. Biswas and A. P. Chandrakasan, "Conv-ram: An energy-efficient sram with embedded convolution computation for low-power cnn-based machine learning applications," in *Solid-State Circuits Conference-(ISSCC), 2018 IEEE International*. IEEE, 2018, pp. 488–490.

[3] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[4] 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," *arXiv preprint arXiv:1602.02830*, 2016.

[5] L. Jiang, M. Kim, W. Wen, and D. Wang, "Xnor-pop: A processing-in-memory architecture for binary convolutional neural networks in wide-io2 drams," in *Low Power Electronics and Design (ISLPED, 2017 IEEE/ACM International Symposium on*. IEEE, 2017, pp. 1–6.

[6] T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang, "Binary convolutional neural network on rram," in *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*. IEEE, 2017, pp. 782–787.

[7] S. Angizi, Z. He, F. Parveen, and D. Fan, "Imce: Energy-efficient bit-wise in-memory convolution engine for deep neural network," in *Proceedings of the 23rd Asia and South Pacific Design Automation Conference*. IEEE Press, 2018, pp. 111–116.

[8] M. Wang, W. Cai, K. Cao, J. Zhou, J. Wrona, S. Peng, H. Yang, J. Wei, W. Kang, Y. Zhang *et al.*, "Current-induced magnetization switching in atom-thick tungsten engineered perpendicular magnetic tunnel junctions with large tunnel magnetoresistance," *Nature communications*, vol. 9, no. 1, p. 671, 2018.

[9] 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 Transactions on Magnetics*, vol. 45, no. 10, pp. 3784–3787, 2009.

[10] F. Tu, S. Yin, P. Ouyang, S. Tang, L. Liu, and S. Wei, "Deep convolutional neural network architecture with reconfigurable computation patterns," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 8, pp. 2220–2233, 2017.

[11] Y. Li, Z. Liu, K. Xu, H. Yu, and F. Ren, "A 7.663-tops 8.2-w energy-efficient fpga accelerator for binary convolutional neural networks," in *FPGA*, 2017, pp. 290–291.

[12] R. Zhao, W. Song, W. Zhang, T. Xing, J.-H. Lin, M. Srivastava, R. Gupta, and Z. Zhang, "Accelerating binarized convolutional neural networks with software-programmable fpgas," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2017, pp. 15–24.

[13] Z. Wang, L. Zhang, M. Wang, Z. Wang, D. Zhu, Y. Zhang, and W. Zhao, "High-density nand-like spin transfer torque memory with spin orbit torque erase operation," *IEEE Electron Device Letters*, vol. 39, no. 3, pp. 343–346, 2018.

[14] Y.-W. Oh, S.-h. C. Baek, Y. Kim, H. Y. Lee, K.-D. Lee, C.-G. Yang, E.-S. Park, K.-S. Lee, K.-W. Kim, G. Go *et al.*, "Field-free switching of perpendicular magnetization through spin–orbit torque in antiferromagnet/ferromagnet/oxide structures," *Nature nanotechnology*, vol. 11, no. 10, p. 878, 2016.

[15] L. Chang, Z. Wang, A. O. Glova, J. Zhao, Y. Zhang, Y. Xie, and W. Zhao, "Prescott: Preset-based cross-point architecture for spin-orbit-torque magnetic random access memory," in *Computer-Aided Design (ICCAD), 2017 IEEE/ACM International Conference on*. IEEE, 2017, pp. 245–252.

[16] 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 Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 2, pp. 120–133, 2016.

[17] M. Wang, W. Cai, D. Zhu, Z. Wang, J. Kan, Z. Zhao, K. Cao, Z. Wang, Y. Zhang, T. Zhang *et al.*, "Field-free switching of a perpendicular magnetic tunnel junction through the interplay of spin–orbit and spin-transfer torques," *Nature Electronics*, vol. 1, no. 11, p. 582, 2018.

[18] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, and K. Yelick, "A case for intelligent ram," *IEEE micro*, vol. 17, no. 2, pp. 34–44, 1997.

[19] T. Quemerais, D. Gloria, D. Golanski, and S. Bouvot, "High-q mos varactors for millimeter-wave applications in cmos 28-nm fdsoi," *IEEE Electron Device Letters*, vol. 36, no. 2, pp. 87–89, 2015.

[20] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 994–1007, 2012.

[21] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "Finn: A framework for fast, scalable binarized neural network inference," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2017, pp. 65–74.