Energy Efficient In-Memory Binary Deep Neural Network Accelerator with Dual-Mode SOT-MRAM

Deliang Fan and Shaahin Angizi
Department of Electrical and Computer Engineering
University of Central Florida, Orlando, FL 32816
Email: {dfan, angizi}@ucf.edu

***Abstract*—In this paper, we explore potentials of leveraging spin-based in-memory computing platform as an accelerator for Binary Convolutional Neural Networks (BCNN). Such platform can implement the dominant convolution computation based on presented Spin Orbit Torque Magnetic Random Access Memory (SOT-MRAM) array. The proposed array architecture could simultaneously work as non-volatile memory and a reconfigurable in-memory logic (AND, OR) without add-on logic circuits to memory chip as in conventional logic-in-memory designs. The computed logic output could be also simply read out like a normal MRAM bit-cell using the shared memory peripheral circuits. We employ such intrinsic in-memory computing architecture to efficiently process data within memory to greatly reduce power-hungry and omit long distance data communication concerning state-of-the-art BCNN hardware.**

***Index Terms*—In-memory computing, SOT-MRAM, Convolutional Neural Network**

I. Introduction

Deep learning Convolutional Neural Network (CNN) has achieved world-wide attention due to close-to-human or even better than human performance in image recognition over large scale dataset such as ImageNet [1]–[3]. Following the trend, when going deeper in CNNs (e.g. ResNet employs 18-1001 layers), memory/computational resources and their communication have faced inevitable limitations. This can be interpreted as 'CNN power and memory wall' [1], leading to different approaches to improve CNN efficiency at either algorithm or hardware level. Recently, Binary Convolutional Neural Network (BCNN) has achieved almost similar accuracy as CNN on large datasets by breaking the network parameters' high precision limit [4]. It binarizes both CNN weights and inputs, providing a promising solution to mitigate storage and computational bottlenecks [5].

From Hardware implementation point of view, the isolated memory and computing units (GPU or CPU) interconnected via buses has faced serious challenges, such as long memory access latency, significant congestion at I/Os, limited memory bandwidth, and huge leakage power consumption for the neural network acceleration. To address these concerns, in-memory processing platform based on non-volatile memory can be an alternative solution to integrate memory and logic, leading to an energy-efficient information processing platform

Partial support of this research was provided by the National Science Fundation under Grant No. 1740126 and the Woodrow W. Everett, Jr. SCEEE Development Fund.

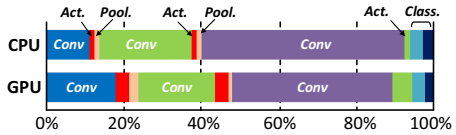


Figure 1: Execution time of a sample CNN for scene labeling on CPU and GPU [13].

[6]–[8]. Spin-transfer torque magnetic random access memory (STT-MRAM) [9] and recent Spin Orbit Torque Magnetic Random Access Memory (SOT-MRAM) [10] are very promising to pave a new way to realize such area and energy-efficient systems supporting in-memory processing with features like non-volatility, zero standby leakage, compatibility with CMOS fabrication process and excellent integration density [11], [12].

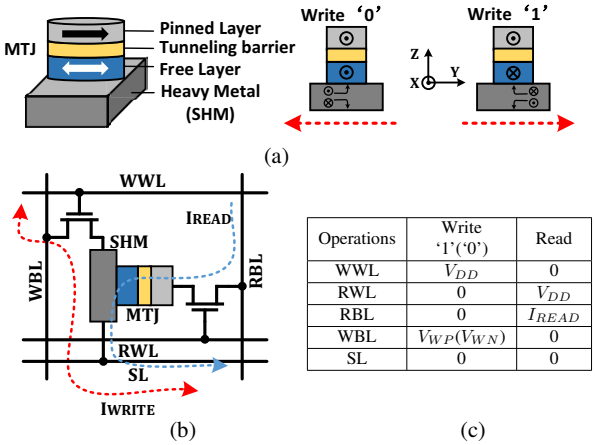
CNN (/BCNN) typically consists of multiple layers, namely convolution, activation, and pooling, where as depicted in Fig. 1 [13], convolutional layer always takes most fraction (over 90%) of execute time and computational sources in GPU, CPU, FPGA and ASIC implementations [1-4]. In this work, we present a convolution-in-memory engine (CIM) that can implement the dominant convolution computation within memory based on our presented dual-mode SOT-MRAM array architecture to greatly accelerate Binary CNN (BCNN). We show that performing the most computationally intensive convolution within non-volatile CIM can thrive three significant objectives: (1) With help of a simple digital processing unit (DPU), all BCNN computation can be implemented within the proposed in-memory BCNN accelerator, where input images are stored. It eliminates massive energy consumption of data communication in traditional architecture between memory and computing units (i.e. CPU/GPU); (2) Reducing the energy consumption of convolution layers through utilizing energy efficient intrinsic in-memory computation; (3) Accelerating the inference task by employing in-memory parallelism.

II. In-Memory Processing Platform

A. SOT-MRAM

Fig. 2a shows a Spin-Orbit Torque Magnetic Random Access Memory (SOT-MRAM) device structure with the composite structure of spin Hall metal (SHM) and Magnetic Tunnel Junction (MTJ). Here the flow of charge current ($\pm$y) through the SHM (Tungsten, $\beta - W$ [14]) will cause accumulation of opposite directed spin on both surfaces of SHM due to spin

1063-6404/17 $31.00 © 2017 IEEE
DOI 10.1109/ICCD.2017.107



Operations	Write '1'('0')	Read
WWL	V_{DD}	0
RWL	0	V_{DD}
RBL	0	I_{READ}
WBL	$V_{WP}(V_{WN})$	0
SL	0	0

Figure 2: (a) SOT-MRAM device structure and Spin Hall Effect, (b) Schematic and (c) biasing conditions of SOT-MRAM bit-cell.

Hall effect [11]. Thus, a spin current flowing in $\pm$z is generated and further produces spin-orbit torque (SOT) on the adjacent free magnetic layer, causing switch of magnetization. The bit-cell structure of 2T1R SOT-MRAM and its biasing conditions are shown in Fig. 2b and 2c, respectively.

B. Computational Sub-array Architecture

The presented SOT-MRAM sub-array architecture could work in dual mode that perform both memory read-write and AND/OR logic operations. Fig. 3e shows the architecture of 2×2 memory array. Each SOT-MRAM cell is associated with the Write Word Line (WWL), Read Word Line (RWL), Write Bit Line (WBL), Read Bit Line (RBL), and Source Line (SL) to perform typical memory operations. Moreover, in our design, any two cells in identical column (i.e. RBL) could be sensed simultaneously to implement an in-memory logic function. The peripheral decoders (active-high output) control the activation of current path through the array. Voltage drivers are used with the WBLs for providing the required write voltage. A voltage mode Sense Amplifier (SA) [11] is connected to the RBL for sensing the total resistance in the selected current path during Read or Computing mode.

Memory Write: To write a bit in any of the SOT-MRAM cells, for example in the cell of 1st row and 1st column, write current should be injected through the heavy metal substrate of SOT-MRAM. To activate this write current path, WWL1 should be activated by the Row Decoder and SL1 is grounded, while all the other word lines and source lines are kept deactivated. Now, in order to write '1' (/ '0'), the voltage driver (V1) connected with WBL1 is set to positive (/negative) write voltage. This allows sufficient charge current flows from V1 to ground (/ground to V1), leading to MTJ resistance programmed to High-R_{AP} (/Low-R_P)

Memory Read: For typical memory read, a read current flows from the selected SOT-MRAM cell to ground, generating a sense voltage at the input of SA, which is compared with memory mode reference voltage ($V_{sense,P}<V_{ref}<V_{sense,AP}$). This reference voltage generation branch is selected by setting the Enable values $(EN_M, EN_{AND}, EN_{OR})$= (1,0,0). Now, if the path resistance is higher (/lower) than R_{ref}, i.e. R_{AP} (/R_P), then the output of the SA produces High (/Low) voltage indicating logic '1' (/ '0').

Computing Mode: In this mode, every two bits stored in the identical column can be selected and sensed simultaneously as depicted in Fig. 3e. Note that, the row decoders are modified to support multi-line enable function, through combining two single-line enable decoder with their outputs connected to OR gates. Then, the equivalent resistance of such parallelly connected SOT-MRAMs and their cascaded access transistors are compared with a specific reference by SA. Through selecting different reference resistances $(EN_M, EN_{AND}, EN_{OR})$, the SA can perform basic in-memory Boolean functions (i.e. AND and OR). For AND operation, R_{ref} is set at the midpoint of $R_{AP}//R_P$ ('1','0') and $R_{AP}//R_{AP}$ ('1','1'). Thus only when both of the two selected SOT-MRAM bit-cells are in anti-parallel state (i.e. binary input: '1','1'), the output is high, whereas output is low. Similarly, for OR operation, R_{ref} is set at the midpoint of $R_P//R_P$ and $R_P//R_{AP}$.

III. Proposed BCNN Accelerator

BCNN typically imposes the least possible computational complexity to underlying hardware due to the extreme 1-bit quantization of weights/inputs compared to well-trained CNNs which use floating point operations. Interestingly, the proposed dual-mode SOT-MRAM array architecture can also support limited bit-width computation which well-suites to BCNNs. To show this, we take advantage of AlexNet architecture [15], which is a deep CNN successfully performing ImageNet classification task in relative high accuracy. AlexNet is a CNN architecture with 5 convolutional layers and 3 fully connected layers as depicted in Fig. 3a. In recent XNOR-NET [4], the authors have successfully shown the binarized AlexNet (AlexNet-BCNN) with last three fully-connected layers converted into convolutional layers as depicted in Fig. 3a. Thus, such AlexNet-BCNN consists of 8 convolutional layers. Note that, as thoroughly discussed in [4], this BCNN avoids binarization in the first and last layers.

A. Mapping

The binary computational blocks of AlexNet-BCNN (i.e. $Conv2$ to $Conv7$) mainly consist of four consecutive layers: 1-Batch Normalization, 2- Binary Activation, 3-Binary Convolution, and 4-Pooling (see Fig. 3a). Fig. 3b illustrates the split of computational tasks in each layer of block between DPU and CIM units. Similar colors are used to demonstrate tasks performed in one layer (e.g. Sign-Func and Scaling-Fact tasks belong to Bin Active layer). Initially, input (I) and weight (W) tensors are organized in ImageBanks and KernelBanks of memory as depicted in Fig. 3c. To perform the computation, they need to be preprocessed to efficiently map into CIM units.

In our proposed accelerator, we convert the original binary tensor in the form of +1, -1 to 1, 0. Thus the binary convolution of two vector $I(B)$ and $W(B)$ could be computed by AND logic and BitCount operations: $BitCount(and(I(B), W(B))$.

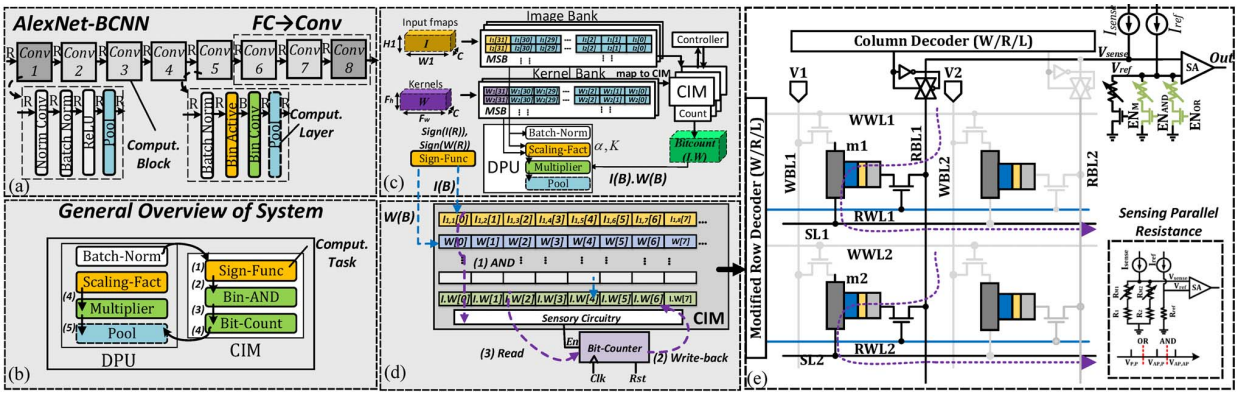


Figure 3: (a) Computational blocks of AlexNet-BCNN [4]. The input/output type of each computational layer is indicated by R (Real-valued) or B (Binary), (b) Mapping of AlexNet-BCNN into the proposed accelerator (CIM) and DPU units, (c) DPU and memory organization with Image/Kernel Banks, (d) Convolution-in-memory engine based on (e) The proposed dual-mode SOT-MRAM array architecture with in-memory logic functions.

As shown in Fig. 1d, such bit-wise AND logic could be intrinsically implemented within CIM unit based on presented SOT-MRAM array without add-on logic circuits, which differentiates from other logic-in-memory designs. As shown in Fig. 1d, assuming tensor-I and tensor-W are stored in different words, in order to implement in-memory logic, two bits stored in the identical column (i.e. Bit-Line) will be selected and sensed simultaneously. Then, the equivalent resistance of such parallelly connected SOT-MRAMs is compared with a specific reference by SA to realize bulk AND operation. Thus, it can be seen that such binary convolution computation could be intrinsically implemented within memory with no need to read out data from memory. Such design will directly read-out the logic output and sends to the following 'bit-counter' (see Fig. 3d) to finish the convolution computation. It is worth to note that, even a DPU is added to take care of miscellaneous computation in one computational block, like batch normalization, scaling and pooling. The mapping method of each computational task is summerized in Table I.

IV. Results and Discussion

A. Main Memory Storage

The breakdown of memory storage required for AlexNet is shown in Fig. 4 under different input/weight bit-levels. AlexNet-BCNN [4] discussed here requires 39.7MB memory storage which is 12× and 6× less compared to double precision (DP) and single precision (SP) CNN implementations, respectively.

B. Energy and Area Estimation

In this subsection, the energy and area of different implementations of AlexNet on ImageNet is estimated. To do the experiment, we have set up a device-to-architecture evaluation platform. The circuit level simulation is initially implemented in Cadence Spectre with NCSU 45nm CMOS PDK [16]. SOT-MRAM resistive model of Fig. 2a is used in the circuit simulation. The MTJ resistance (R_{MTJ}) is obtained from the NEGF approach [11], while the heavy metal resistance (R_{SHM}) is calculated based on the resistivity and device dimension. Accordingly, we massively modified the system level memory evaluation tool NVSim [17] to co-simulate with an in-house developed C++ code based on circuit level results.

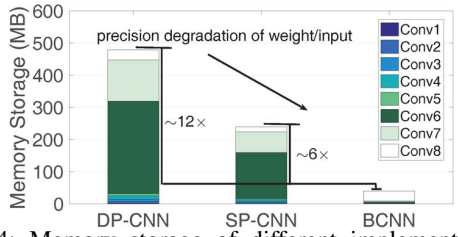


Figure 4: Memory storage of different implementations of AlexNet architecture (Double Precision (DP), Single Precision (SP), and BCNN herein).

Here, we concisely explain our experimental results. The energy consumption breakdown for memory read/write, CIM, and DPU is plotted in Fig. 5a. It can be seen that for the first and last un-binarized layers, DPU handles the computational load. While, CIM takes care of dominant convolution computations for the rest of layers. We also profile the area distribution of different convolutional layers of AlexNet-BCNN and floating-point implementation of AlexNet (CNN) in Fig. 5b. The experimental results show that last three convolutional layers (converted from fully connected layers) take up the most part of the area due to high number of weight parameters and accordingly number of subarrays in either BCNN or CNN implementations. However, the area overhead of BCNN is obviously less than floating-point implementation

The computation energy, area, and execution time of different accelerators for AlexNet-BCNN/CNN on ImageNet dataset are tabulated in Table II. The hardware mapping results show that our accelerator can process BCNN on ImageNet favorably with 310.42 μJ/Img, where ~7× and 1.7× lower energy and area are achieved, respectively, compared to recent RRAM-based BCNN accelerator [3]. Such significant improvements mainly come from: (1) RRAM based convolution computing

Table I: Hardware Mapping of BCNN Computational Tasks to the Proposed Platform

Computational Task	Mapping
Batch-Norm	It alleviates the information loss during binarization by normalizing the input batch to have zero mean and unit variance. $I_o(R) = \frac{I_i(R)-\mu}{\sqrt{\sigma^2+\varepsilon}}\gamma + \beta$
Sign-Func	Sign-Func task can be readily accomplished by reading the sign bit of input and weight tensors from ImageBank and KernelBank (Fig. 3c), and writing it back to an CIM sub-array to make them prepared for next computational task. $I(B) = sign(I(R)), W(B) = sign(W(R))$
Scaling-Fact	Accurate binary convolution between weight kernels W with input fmap I requires calculating two important parameters called scaling factors [4]. The optimum scaling factor for weight kernels can be obtained by the average of absolute weight values $\alpha = \frac{\|W(R)\|_{l1}}{n}$
Bin-AND	After binarizing the inputs and weights, the results are restored in consecutive rows of CIM sub-arrays as depicted in Fig. 3d. Binary weights are inserted in specific rows of CIM's sub-arrays, where inputs can be written in following rows. Then, CIM can efficiently perform bit-wise convolution operations thanks to the following equation which computes the dot-product of two vectors $I(B)$ and $W(B)$ by AND and $BitCount$ operations: $I \circledast W = BitCount(and(I(B), W(B))$
Bit-Count	The output of sensory circuitry is then connected to a CMOS Bit-counter. Only when the read data is binary '1', counter counts upward.
Multiplier	Once the bit-counting outputs are derived and scaling factors α and K are calculated for the weights and inputs, respectively, the convolution between input I and weight kernel W can be approximated by DPU mainly using following equation: $I * W = BitCount(and(I(B), W(B)) \cdot K\alpha$

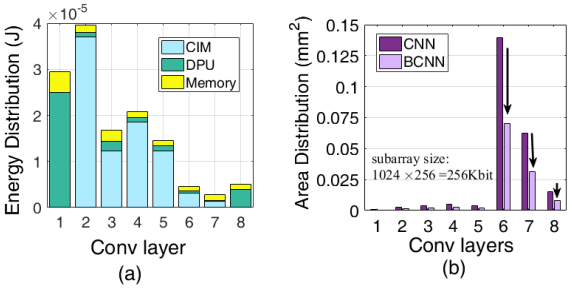


Figure 5: (a) Energy and (b) Area distribution of AlexNet-BCNN in different convolutional layers in inference mode.

unit has to employ matrix splitting due to small array limitation, while the commercialized MRAM chip demonstrates sub-array size of 1024 by 256, which is the array size used in our work; (2) RRAM-based crossbar peripheral circuit's overhead, such as buffers and DAC/ADC, contributes more than 85% of area and energy consumption. While, in our design, the only logic overhead for convolution computation is a digital bit-counter circuit since logic-AND could be fully implemented within memory with shared memory peripheral circuits. We also compare our accelerator with various platforms including NVIDIA Jetson TK1 GPU (GPU1), server class NVIDIA Tesla K40 GPU (GPU2), and Xilinx Zynq-7000 SOC FPGA [2]. As shown in Table II, the proposed accelerator herein along with ASIC design [1] consume least energy per image.

Table II: Performance estimation of BCNNs accelerators for AlexNet on ImageNet. Metrics not reported by prior works are indicated by (-).

Networks		Energy (μJ/img)	Area (mm^2)	Execution time/img (ms)
CNN	RRAM [3]	5444.85	21.25	-
BCNN	RRAM [3]	2275.34	9.19	-
	FPGA [2]	27918	-	5.94
	GPU1 [2]	324000	-	90
	GPU2 [2]	237250	-	0.73
	ASIC [1]	352	1.9	-
	our work	310.42	5.28	10.7

V. Conclusion

This paper presents a Convolution-in-Memory engine (CIM) design based on dual-mode SOT-MRAM array architecture, which could efficiently implement convolution computation within memory to greatly reduce power-hungry and long distance data communication concerning state-of-the-art CNNs. The hardware mapping results shows that CIM can process the Binarized AlexNet [4] on ImageNet favorably with 310.42 μJ/Img, where $\sim$7$\times$ and 1.7$\times$ lower energy and area are achieved, respectively, compared to recent BCNN-RRAM accelerator [3].

References

[1] R. Andri *et al.*, "Yodann: An architecture for ultra-low power binary-weight cnn acceleration," *IEEE TCAD*, 2017.

[2] R. Zhao *et al.*, "Accelerating binarized convolutional neural networks with software-programmable fpgas," in *FPGA*. ACM, 2017.

[3] T. Tang *et al.*, "Binary convolutional neural network on rram," in *22nd ASP-DAC*. IEEE, 2017, pp. 782–787.

[4] M. Rastegari *et al.*, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[5] S. Zhou *et al.*, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606*, 2016.

[6] S. Angizi *et al.*, "Rimpa: A new reconfigurable dual-mode in-memory processing architecture with spin hall effect-driven domain wall motion device," in *ISVLSI*. IEEE, 2017, pp. 45–50.

[7] P. Chi *et al.*, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *ISCA*, vol. 43, 2016.

[8] S. Angizi *et al.*, "Energy efficient in-memory computing platform based on 4-terminal spin hall effect-driven domain wall motion devices," in *GLSVLSI*. ACM, 2017, pp. 77–82.

[9] Y. Kim *et al.*, "Write-optimized reliable design of stt mram," in *Proceedings of the 2012 ACM/IEEE ISLPED*. ACM, 2012.

[10] G. Prenat *et al.*, "Beyond stt-mram, spin orbit torque ram sot-mram for high speed and high reliability applications," in *Spintronics-based Computing*. Springer, 2015, pp. 145–157.

[11] X. Fong *et al.*, "Spin-transfer torque devices for logic and memory: Prospects and perspectives," *IEEE TCAD*, vol. 35, 2016.

[12] S. Angizi *et al.*, "Composite spintronic accuracy-configurable adder for low power digital signal processing," in *ISQED*. IEEE, 2017.

[13] L. Cavigelli *et al.*, "Accelerating real-time embedded scene labeling with convolutional networks," in *DAC, 2015*. IEEE, 2015.

[14] C.-F. Pai *et al.*, "Spin transfer torque devices utilizing the giant spin hall effect of tungsten," *Applied Physics Letters*, vol. 101, 2012.

[15] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[16] (2011) Ncsu eda freepdk45. [Online]. Available: http://www.eda.ncsu.edu/wiki/FreePDK45:Contents

[17] X. Dong *et al.*, "Nvsim: A circuit-level performance, energy, and area model for emerging non-volatile memory," in *Emerging Memory Technologies*. Springer, 2014, pp. 15–50.