Deep Neural Network Acceleration in Non-Volatile Memory: A Digital Approach

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

***Abstract*—Latest algorithmic development has brought competitive classification accuracy for neural networks despite constraining the network parameters to ternary or binary representations. These findings show significant optimization opportunities to replace computationally-intensive convolution operations (based on multiplication) with more efficient and less complex operations such as addition. In hardware implementation domain, processing-in-memory architecture is becoming a promising solution to alleviate enormous energy-hungry data communication between memory and processing units, bringing considerable improvement for system performance and energy efficiency while running such large networks. In this paper, we review several of our recent works regarding Processing-in-Memory (PIM) accelerator based on Magnetic Random Access Memory computational sub-arrays to accelerate the inference mode of quantized neural networks using digital non-volatile memory rather than using analog crossbar operation. In this way, we investigate the performance of two distinct in-memory addition schemes compared to other digital methods based on processing-in-DRAM/GPU/ASIC design to tackle DNN power and memory wall bottleneck.**

***Index Terms*—Depp Neural network acceleration, In-memory computing, Magnetic Random Access Memory**

I. Introduction

To reduce the massive MAC operations and memory requirement in Deep Convolutional Neural Networks (DNN), researchers have proposed various quantized/binarized DNNs by limiting inputs, weights or gradients to be quantized/ binarized specifically in forward propagation [1], [2].The DoReFa-Net achieves high accuracy over both SVHN and ImageNet data-sets with various low bit-width configurations after applying quantization methods [1]. Besides, the extremely-quantized Binary Neural Networks (BNNs) have obtained close accuracy to existing DNNs [2]. An identical yet updated version of binary training has been also applied for Generative Adversarial Networks (GANs) [3] to reduce the memory utilization, thus improving the hardware deployment efficiency.

From DNN/GAN accelerator architecture perspective, the isolated processing and memory units (i.e., GPU or CPU) are connected through buses which has dealt with a wide variety of challenges, i.e. congestion at I/Os, long memory access latency, limited memory bandwidth, huge data transfer energy and leakage power for saving network parameters in the volatile memory [4]. To alleviate such concerns, Processing-in-Memory (PIM) platforms has been widely explored in recent accelerators design [4]–[13] to bring a potential solution for memory wall challenge. The major idea of PIM is to realize computational logic units inside the memory to process data utilizing the inherent memory parallelism and huge internal memory bandwidth. Such mechanism is expected to bring significant efficiency in the off-chip data transfer latency and energy. Meanwhile, a lot of research efforts have been carried out in emerging technology based on Non-Volatile Memory (NVM), such as resistive RAM (ReRAM) [4], Magnetic RAM (MRAM) and etc. Spin-Transfer Torque Magnetic Random Access Memory (STT-MRAM) [14] and Spin-Orbit Torque Magnetic Random Access Memory (SOT-MRAM) [5] are energy-efficient and high performance candidates, mainly owning to superior endurance, low switching energy, CMOS technology compatibility, etc. By leveraging such intrinsic resistive property, the bulk bit-wise in-memory operations such as AND/OR/XOR have been implemented through the method of routed sensing [15]–[17], that opens a novel route to realize efficient PIM platforms. Although, this effort has been limited to fundamental logic operations so far. Such functions are not necessarily applicable to many tasks except by imposing multi-cycle operations [16], [18] to implement more complex combinational logics such as addition.

In this paper, we review several of our recent works about acceleration of in-memory addition operations that could be used in Binary-Weight DNNs (BWNNs) in an SOT-MRAM based in-memory computing platform, based on set of microarchitectural and circuit-level designs.

II. Binary-Weight Neural Networks

DNN performs in two distinct modes, in training mode, the configuration parameters of layers are computed by training the network on pre-classified images, and in inference mode, test images are fed to the network for examination. As mentioned earlier, Multiplication and Accumulation operations (MAC) accounts for the most arithmetic operations [19] used in both modes. To remove the need for massive multiplication operations and memory storage, researchers have developed various Binary-Weight DNNs (BWNN) by constraining the weights to binary values in forward propagation. BinaryConnect [2] achieves close results to full-precision networks on MNIST and CIFAR-10 data-sets by training the networks with binary weights (-1, +1). This method can potentially replace computationally-intensive multiplication operations with much simpler complement addition/subtraction operations [19]. Such replacement remarkably reduces weight storage requirements. The following equation [2] gives the binarization functions in deterministic and stochastic way for w_{fp} (floating point weights):

$$w_{b,De} = \begin{cases} +1, w_{fp} \geq 0 \\ -1, w_{fp} < 0 \end{cases}, w_{b,St} = \begin{cases} +1, & p = \sigma(w_{fp}) \\ -1, & 1-p \end{cases} \quad (1)$$

where σ denotes a hard sigmoid function that determines the distribution probability:

$$\sigma(x) = clip(\frac{x+1}{2}, 0, 1) = max(0, min(1, \frac{x+1}{2})) \quad (2)$$

Now, we present the MRAM platform to accelerate BWNNs in the PIM context.

III. Processing-in-MRAM Accelerator

The presented accelerator architecture is depicted in Fig. 1a with computational SOT-MRAM sub-arrays, kernel and image banks, and a Digital Processing Unit (DPU) with three sub-components as represented by Binarizer, Activation Function, and Batch Normalization. The platform is mainly controlled by Ctrl (located in each sub-array) to run whole BWNNs. In the first step, with Kernels (W) and Input feature maps (I) that are respectively saved in Kernel and Image Banks, W has to be instantly binarized for mapping into sub-arrays. Note that this doesn't apply to the first layer [1]. Moreover, binarized shared kernels will be used for different inputs. This operation is implemented through DPU's Binarizer-Bin. (shown in Fig. 1a) and then outputs are sent to the sub-arrays, developed to handle the computational load employing PIM methods. In the second and third steps, as will be throughly explained in the next subsections, the parallel computational sub-arrays perform to feature extraction employing combining and parallel compute. schemes. Eventually, the accelerator's DPU activates the resultant feature map to complete the fourth step by producing output feature map.

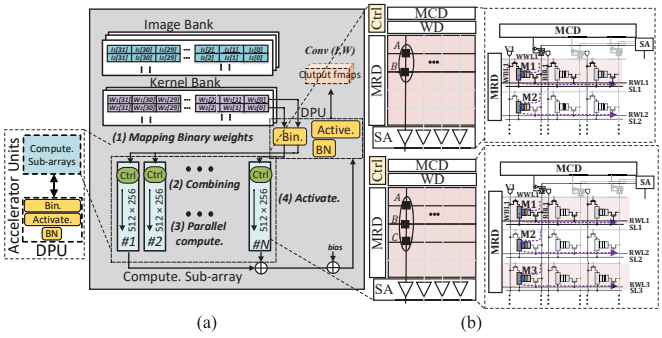


Fig. 1. (a) MRAM accelerator platform, (b) Computational sub-array design.

A. Sub-array with Reconfigurable Sense Amplifier

Fig. 1b illustrates the in-memory computing sub-array architecture implemented by SOT-MRAM. This sub-array is basically composed of a Memory Row Decoder (MRD), a Memory Column Decoder (MCD), a Write Driver (WD) and n Sense Amplifier (SA) ($n \in$ # of columns) that are adjustable by Ctrl to implement a dual mode computation i.e. memory write/read and in-memory logic operations. SOT-MRAM device is a composite device of a spin Hall metal (SHM) and a Magnetic Tunnel Junction (MTJ). Considering MTJ as the main storage element, the parallel magnetization resistance in both magnetic layers is represented by '0' and is lower than that of anti-parallel magnetization resistance ('1'). Every SOT-MRAM bit-cell in computational sub-arrays is connected to five controlling signals namely Write Word Line (WWL), Write Bit Line (WBL), Read Word Line (RWL), Read Bit Line (RBL), and Source Line (SL). The computational sub-array is mainly developed to realize the computation between in-memory operands using two different mechanisms as called two-row activation and three-column activation. These mechanisms will be later used to respectively implement bulk bit-wise in-memory AND and addition/subtraction (`add/sub`).

a) Bit-line Computation Mode: The presented SOT-MRAM sub-array is developed to realize the bulk bit-wise in-memory logic operations between every two or three operands positioned in the similar bit-line. Generally, in the 2-/ 3-input in-memory logic method, every two/three bits stored in the identical column are selected with the MRD [20] and simultaneously sensed by SA connected to the same bit-line. Accordingly, a programmable SA's reference selected by Ctrl will be compared with the equivalent resistance of such parallel connected bit cells and their cascaded access transistors after injecting a small amount of current over resistors. Through selecting different reference resistances e.g. R_{AND2} and R_{OR2}, the SA can perform basic 2-input in-memory Boolean functions AND and OR, respectively. For example, to realize AND operation, R_{ref} is set at the midpoint of $R_{AP}//R_P$ ('1','0') and $R_{AP}//R_{AP}$ ('1','1'). With the data organization shown in Fig. 1b, where A and B operands correspond to M1 and M2 memory cells, respectively, 2-input in-memory method outputs AB in only one memory cycle. The idea of voltage comparison between V_{sense} and V_{ref} is shown on Fig. 2 for different operations. The reconfigurable SA can also provide 2-input NOR, NAND functions through the complementary outputs. Here, we present two reconfigurable SAs [8], [21], as shown in Fig. 3, that can be used alternatively through a direct connection to the same computational sub-array BL to perform the computation.

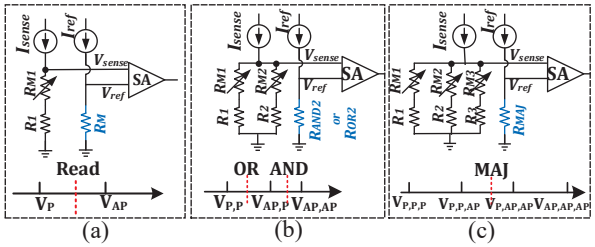


Fig. 2. Voltage comparison between V_{sense} and V_{ref} for (a) read, (b) 2-input, (c) 3-input in-memory logic operation.

b) SA-I: The SA-I, as shown in Fig. 3a, has 2 sub-SAs and 4 reference-resistance branches that could be enabled by Enable bits ($EN_M, EN_{OR2}, EN_{MAJ}, EN_{AND2}$) by the Ctrl to implement both memory and computation operations based on Table I. The memory and single-threshold logic functions could be readily realized only by activating one enable bit at a time. As an instance, by enabling EN_{AND2}, 2-input `AND/NAND` function is carried out on the bit-line between two operands. Besides, two enables can be activated at a

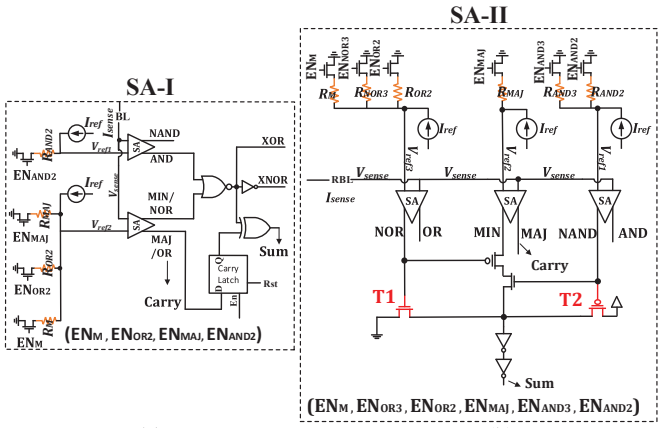


Fig. 3. Sense amplifier designs: (a) SA-I [8], (b) SA-II [21]

time e.g. EN_{OR2}, EN_{AND2}, to generate two separate logic functions, which is in particular useful to output double-threshold logic functions like XOR/XNOR. Additionally, 3-input majority/minority functions (MAJ/MIN) in a single SA's sensing cycle can be realized by selecting every three cells located in a same bit-line by MRD and sensing simultaneously. Assuming the data organization in Fig. 1b having A, B and C operands respectively mapped to M1, M2 and M3 cells, the sub-array performs logic $AB + AC + BC$ function easily by enabling EN_{MAJ}. Fig. 2c shows that R_{MAJ} is considered at the midpoint of $R_P//R_P//R_{AP}$ ('0','0','1') and $R_P//R_{AP}//R_{AP}$ ('0','1','1') to realize MAJ operation.

TABLE I
CONFIGURATION OF ENABLE BITS FOR SA-I.

In-memory Operations	read	OR2/ NOR2	AND2/ NAND2	MAJ/ MIN	XOR2/ XNOR2
EN_M	1	0	0	0	0
EN_{OR2}	0	1	0	0	1
EN_{MAJ}	0	0	0	1	0
EN_{AND2}	0	0	1	0	1

Besides the aforementioned single-cycle operations, the accelerator's sub-arrays can perform addition/subtraction (add/sub) operation very efficiently in two cycles. In the first cycle, Carry-out bit of the Full-Adder (FA) Boolean function can be produced by MAJ function (Carry in Fig. 3a) by enabling EN_{MAJ}. Accordingly, we devised a latch after SAs to save intermediate data for carry such that it can be applied in summation of next bits. In the second cycle, by inserting a 2-input XOR gate in reconfigurable SA, Sum output can be obtained through XOR3 operation. Thus, considering A, B and C operands, the 2- and 3-input in-memory mechanisms could efficiently generate Sum(/Difference) and Carry(/Borrow) bits in two memory cycles.

c) SA-II: The SA-II, as illustrated in Fig. 3b, has 3 sub-SAs and totally 6 reference-resistance branches that could be enabled by enable bits (EN_M, EN_{OR3}, EN_{OR2}, EN_{MAJ}, EN_{AND3}, EN_{AND2}) by the Ctrl to implement the memory and computation modes based on Table II. SA-II realizes memory read and single-threshold logic operations through activating one enable in either branches at a time. Additionally, by activating more than two enables at a time, more thatn on logic functions could be simultaneously realized with SAs that could be employed to generate complex multi-threshold logic functions such as XOR3/XNOR3.

The computational sub-array can also perform add/sub operation based on SA-II but in a single cycle. With an observation on the FA truth table, it can be observed that for 6 out of 8 possible input combinations, Sum output equals inverted Carry signal. Besides, as discussed in SA-I, FA's Carry is directly generated by MAJ function. Based on this, SA-II can realize such Sum and Carry outputs readily by MIN and MAJ functions, respectively. Fig. 3b shows that the Sum bit is achieved from the SA's MIN output. But, in order to consider two extreme cases namely (0,0,0) and (1,1,1), we need to disconnect the MIN signal from Sum or else wrong result is achieved. It turns out that for these two cases, Sum signal can be implemented respectively by NOR3 (when T1:ON, T2:OFF $\rightarrow$ Sum=0) and NAND3 functions (when T1:OFF, T2:ON $\rightarrow$ Sum=1). This is implemented by inserting four additional pass transistors (two in the MIN function path for disconnecting the MIN signal and two to connect the Sum output to the corresponding GND or Vdd). It is worth pointing out that Sum output is the XOR3 function, thus the SA-II can realize 2-/3-input XOR/XNOR functions as well, without additional XOR gates like state-of-the-art designs [7], [18]. Considering A, B and C as input operands, 3-input in-memory logic with SA-II design outputs Sum(/Difference) and Carry(/Borrow) bits in only one memory cycle. The presented accelerator with SA-II could be considered as the first processing-in-MRAM platform capable of performing bulk in-memory addition in a single cycle, where for example processing-in-DRAM platforms such as Ambit [18] require over 10 cycles.

TABLE II
CONFIGURATION OF ENABLE BITS FOR SA-II.

Ops.	read	OR2/ NOR2	AND2/ NAND2	MAJ/ MIN	OR3/ NOR3	AND3/ NAND3	Add/ XOR3/XNOR3 XOR2/XNOR2
EN_M	1	0	0	0	0	0	0
EN_{OR2}	0	1	0	0	0	0	0
EN_{AND2}	0	0	1	0	0	0	0
EN_{OR3}	0	0	0	0	1	0	1
EN_{AND3}	0	0	0	0	0	1	1
EN_{MAJ}	0	0	0	1	0	0	1

B. Reliability Analysis

To analyze the impact of process variation and the variation tolerance of the presented sensing circuits, we ran Monte-Carlo simulation with 10000 iterations. We add $\sigma = 2\%$ variation to the RA_P (Resistance-Area product) and a $\sigma = 5\%$ variation to the TMR (Tunneling MagnetoResistive). Fig. 4 depicts the sense voltage simulation result for the sense margin for single-cell memory read, 2 fan-in in-memory logic and 3 fan-in in-memory operations. We observe that voltage sense margin gradually diminishes by increasing the number of selected cells. To avert the logic failure and enhance the reliability of SA outputs, the number of sensed cells are limited to maximum three. Our simulations show that such sense margin could be further enhanced by increasing either the MTJ's oxide

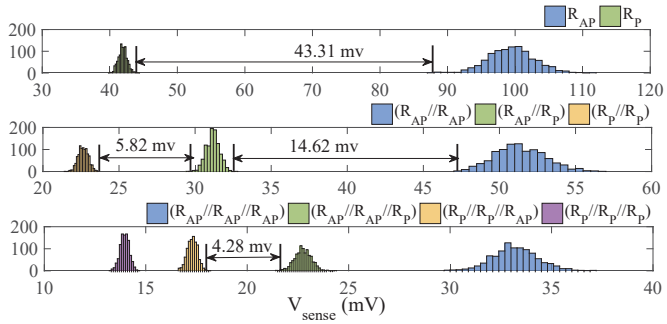


Fig. 4. Monte-Carlo simulation of V_{sense} for (top) read operation, and bit-line computation with (middle) two selected SOT-MRAM bit-cells (down) three selected SOT-MRAM bit-cells.

thickness or the sense current, but by sacrificing the energy-efficiency of the platform.

C. Convolution Schemes

a) Bit-wise Adder: As mentioned earlier, `add/sub` accounts for the major computational load of BWNNs as it is iteratively used in different layers and takes up the majority of the run-time in these networks. As the most crucial unit of the accelerator, `add/sub` unit must show resource-efficiency and high throughput while handling various input bit-widths at run-time. Here, we present a parallel in-memory adder (/subtractor) scheme on top of 2-/3-input in-memory logic mechanisms discussed in previous subsections to accelerate multi-bit `add/sub` operations.

Fig. 5a shows the required data organization and also computation steps for binary-weight layers of BWNNs. Fig. 5b gives a straightforward example only assuming `add` operations based on SA-I design. Obviously `sub` could be implemented based on `add`.

(1) First, the PIM accelerator selects c channels (here, 4) from input batch with the size of $kh \times kw$ (here, 3×3) and outputs a batch (called combined batch) according to the binary kernel batch {-1,+1}. This operation is easily performed by altering the sign-bit of input according to kernel data ($f1 * -1 = -f1$). (2) Now, as depicted in Fig. 5a, channels of the combined batch are transposed and mapped to different sub-arrays. From computational throughput standpoint, assuming n-activated sub-arrays ($x \times y$), `add/sub` operation of maximum x elements of m-bit ($3m + 2 \leq y$) could be performed in parallel in each sub-array. Thus, the accelerator could process $n \times x$ elements. (3) To generate the output feature maps in parallel, the accelerator's in-memory adder now operates. Fig. 5a R.H.S. shows the sub-array organization for such parallel in-memory operation. We considered m (here, 4) reserved rows for Sum and 2 reserved rows for Carry that are preset to zero. Furthermore, we show the SA latch's current state (Q) and the next state (Q*). In Fig. 5b, we take the `add` operation of two matrices (Ch1 and Ch2) with 4-bit data to explain how the platform operates. We use bit-line alignment technique for `add` operation where two corresponding operands in different matrices have to be aligned in the same bit-line before performing the operation.

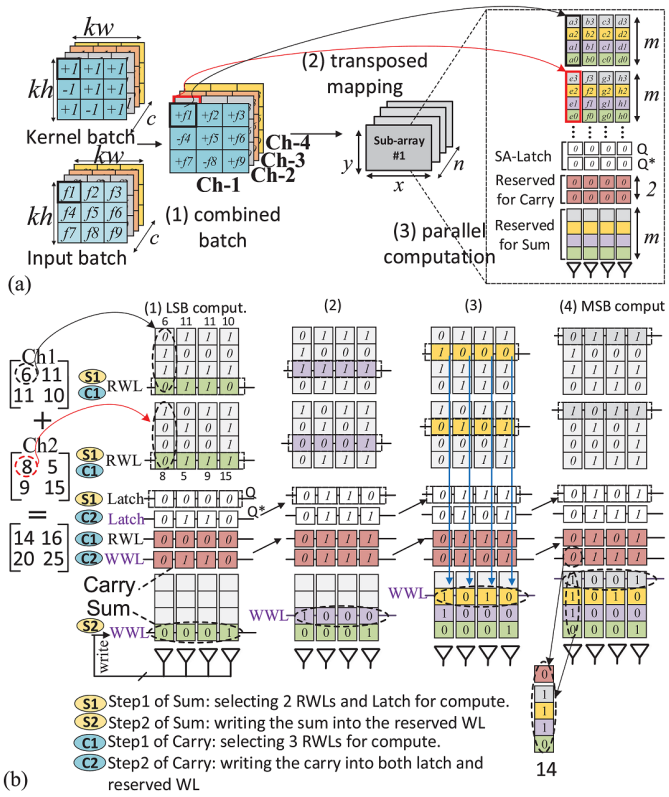


Fig. 5. (a) Memory organization, mapping and computing steps of binary-weight layers, (b) Parallel in-memory addition operation for sum and carry-out logic.

As shown, Ch1 and Ch2 elements are aligned in the same sub-array where the elements are consecutively stored within 8 rows.

The `add` operation begins with the LSBs of the two elements and goes to MSBs. Two cycles are considered for every bit-position computation with four steps as shown by S1, S2, C1 and C2 in Fig. 5b. In S1, 2 RWLs (LSBs) and Latch (preset to zero) are activated to produce the sum according to the mechanism presented in the previous subsections. During S2, a WWL is enabled to store back the resultant Sum. During C1, 2 operands and one of the reserved Carry rows are activated to produce the carry-out again using bit-line computation mode of the platform. In C2, a WWL is enabled to store back such carry-out result simultaneously into a reserved row and in the latch. This carry-out bit every time overwrites the carry latch data and is considered as the carry-in of the next computation cycle. This computation ends after $2 \times m$ cycles (m is # bits in elements).

b) Bit-wise Convolver: The BWNNs consist of few other layers, in addition to binarized layers, i.e. first layer that takes image as inputs, not implemented by `add/sub` and Fully-Connected (FC) layers. It is worth mentioning that FC layers are equivalently implemented by bit-wise convolution operations with 1×1 kernels [1]. Such layers are accelerated by exploiting *AND* and *count* operations as thoroughly explained in [1], [5], [22]. The 2-input in-memory AND mechanism of the accelerator could be leveraged to process bulk bit-wise

AND logic. Moreover, count operation is converted to addition of bits. In the interest of space, we omit the hardware design of this part and allude the readership to IMCE platform [5].

IV. Experimental Results

Here, we compare our MRAM platform with two SA configurations (SA-I and SA-II) with various BWNNs acceleration methods based on DRAM, GPU, and ASIC. It is clear that the larger chip area is, then the higher performance for our platform and other accelerators are achieved due to having additional sub-arrays or computational units, albeit the memory die size impacts the area cost. To have a fair comparison in this work, we report the area-normalized results (performance/energy per area), henceforth.

a) Configuration: ***MRAM:*** We use 512Mb total capacity with 256x512 memory sub-array organized in a H-tree routing manner. Accordingly, we developed an extensive bottom-up evaluation framework with two new simulators. For the device simulations, NEGF and LLG with spin Hall effect equations were taken to model SOT-MRAM bit-cell [5], [14]. At circuit level, we develop a Verilog-A model for 2T1R bit-cell, which can be used along with interface CMOS circuits in Cadence Spectre. We specifically used 45nm NCSU PDK library [23] to assess the presented circuit designs and obtain the performance metrics. At architectural level, based on the device-circuit results, we first extensively modified NVSim [24]. It can change the configuration files (.cfg) according to the various memory array organization and assess the the accelerator performance for PIM operations. Then, we develop a behavioral simulator in Matlab that assess the energy and latency parameters on BWNNs. Besides, we integrated our mapping optimization algorithm to increase the throughput w.r.t. the available resources. ***DRAM:*** We designed a processing-in-DRAM platform based on DRISA platform [25] for BWNNs. Here, we specifically use the $1T1C\text{-}adder$ mechanism to compare the platforms. This mechanism mainly adds a very large n-bit adder circuitry after SAs to do n-bit BLs addition. For simulations, CACTI [26] was modified. We also implemented the adders and controllers in Design Compiler [27]. ***GPU:*** We exploited the NVIDIA GTX 1080Ti Pascal GPU with 3584 CUDA cores with 11TFLOPs peak performance. NVIDIA's system management interface was used to evaluate the energy consumption. Here, we shrunk the results by 50% to eliminate the energy used for cooling. ***ASIC:*** We implemented an ASIC accelerator based on YodaNN [19] and synthesized the design with Design Compiler [27] with 45 nm technology. CACTI tool was used to estimate the SRAM and eDRAM performance [28].

We use a DNN with following configuration: six convolutional layers (binary-weight), two pooling layers (average) and two FCs. We avoid quantizing the first and last layers [1], [5] to prevent any degradation in prediction accuracy. An 8-bit configuration of the inputs is also considered for evaluation. For the data-set, we select the SVHN with 32×32 cropped colored images centered around each digit. We then re-sized the images to 40×40 before feeding to the model.

b) Energy Consumption & Performance: Fig. 6a depicts the MRAM platform's energy consumption results (considering the number of frames per joule) versus other accelerators to run the same model and network having two batch sizes i.e. 8/32. It can be observed that the lower energy-efficiency is obtained when batch is larger. Besides, we can see that MRAM solution with SA-I configuration is selected as the most energy-efficient design compared with others due to its fully-parallel and energy-saving and operations. It indicates 1.2× and 3.9× on average better energy-efficiency than that of MRAM-SA-II and DRAM 1T1C-adder platforms. DRAM-based PIM accelerators [25] suffer a high refresh power. Besides due to charge sharing characteristic of capacitors, they deal with *data-overwritten* bottleneck. As result a result of this issue, the computation ultimately overwrites the initial operands. For alleviating this issue, the researchers have come up with *multi-cycle operations* leading to even degrading the PIM platform's performance [11], [18]. It is worth pointing out that in DRAM-1T1C-adder platform, an n-bit adder is designed after SAs, however it cannot provide better performance because of its intrinsically non-parallel operations that limits its energy-saving. In addition, we can see that MRAM platform shows ∼33.8× more energy-efficiency compared with ASIC-256 solution. This reduction is basically from: (1) the energy-efficient `add/sub` operations in the presented accelerator that replace the computationally-intensive standard convolution, (2) the high resource-utilization ratio of the MRAM platform as compared with limited and multiplexed computation in ASIC design. Fig. 6a depicts that MRAM platform achieves ∼72× better energy-efficiency compared to the GPU.

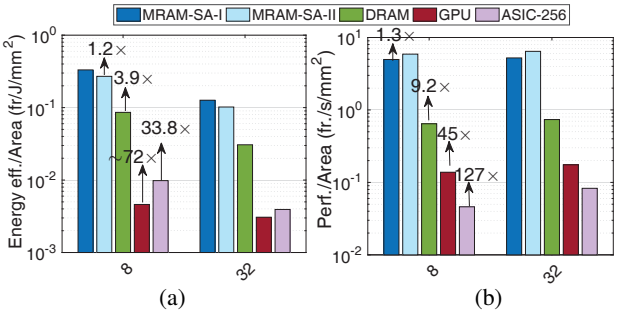
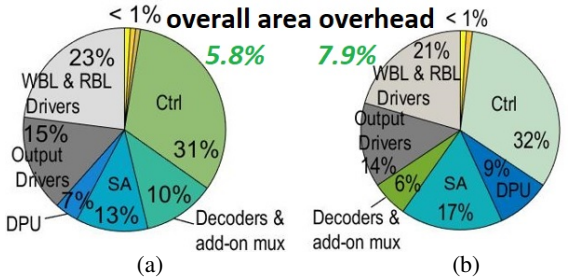


Fig. 6. (a) Energy-efficiency and (b) Performance evaluation of different accelerators for running BWNNs normalized to area (Y-axis=Log scale).

Fig. 6b illustrates the MRAM platform's performance results (frames per second). Based on this figure, we can see that MRAM with SA-II configuration is ∼1.3× faster than that of SA-I solution mainly owning to its one-cycle addition scheme. Note that, as the results are normalized to area and SA-II configuration imposes larger area overhead than that that of SA-I design, we cannot observe a remarkable performance improvement in this case. MRAM with SA-II configuration platform outperforms DRAM and ASIC solutions by 9.2× and 127×, respectively. This is coming from: first, parallel and high-speed in-memory logic operations of the MRAM platform versus multi-cycle processing-in-DRAM operations and second, the widening mismatch between data transfer

and computation in ASIC platform and even 1T1C-adder solution. We can also observe that the higher performance is achieved for MRAM platform solution compared DRAM as the batch size is getting larger. This is because its more paralleled computations. Compared to GPU-based solution, MRAM platform can obtain roughly 45× higher performance normalized to area.

c) Area Overhead: To estimate the area overhead of two configurations of MRAM platform, three main hardware cost sources must be taken into consideration. First, add-on transistors to SAs connected to each BL. Second, the modified MRD overhead; we modify each WL driver by adding two more transistors in the typical buffer chain. Third, the Ctrl's overhead to control enable bits; ctrl generates the activation bits with MUX units with 6 transistors. Overall, the presented processing-in-MRAM platform based in SA-I imposes 5.8% overhead to main memory die, where SA-II imposes 7.9%. Fig. 7 reports such area overhead breakdown for two configurations.



(a) (b)

Fig. 7. Area overhead of processing-in-MRAM platform with (a) SA-I and (b) SA-II.

V. Conclusion

In this paper, we reviewed some of our recent Processing-in-Memory accelerators based on MRAM computational sub-arrays to efficiently accelerate the inference process of BWNNs within digital non-volatile memory rather than using analog crossbar operation. We especially investigated the performance of two distinct in-memory addition schemes compared to other digital methods based on processing-in-DRAM/GPU/ASIC design to tackle DNN power and memory wall bottleneck.

Acknowledgements

This work is supported in part by the National Science Foundation under Grant No. 1740126, Semiconductor Research Corporation nCORE, Cyber Florida Collaborative Seed Award Program and UCF NSTC Research Seed Award.

References

[1] 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," *arXiv preprint arXiv:1606.06160*, 2016.

[2] M. Courbariaux *et al.*, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, 2015, pp. 3123–3131.

[3] J. Song, "Binary generative adversarial networks for image retrieval," *arXiv preprint arXiv:1708.04150*, 2017.

[4] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3. IEEE Press, 2016, pp. 27–39.

[5] 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.

[6] S. Angizi, Z. He *et al.*, "Cmp-pim: an energy-efficient comparator-based processing-in-memory neural network accelerator," in *Proceedings of the 55th Annual Design Automation Conference*. ACM, 2018, p. 105.

[7] C. Eckert, X. Wang, J. Wang, A. Subramaniyan, R. Iyer, D. Sylvester, D. Blaauw, and R. Das, "Neural cache: Bit-serial in-cache acceleration of deep neural networks," in *Proceedings of the 45th Annual International Symposium on Computer Architecture*. IEEE Press, 2018, pp. 383–396.

[8] S. Angizi, Z. He, and D. Fan, "Parapim: a parallel processing-in-memory accelerator for binary-weight deep neural networks," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. ACM, 2019, pp. 127–132.

[9] S. Angizi *et al.*, "Dima: a depthwise cnn in-memory accelerator," in *ICCAD*. IEEE, 2018, pp. 1–8.

[10] S. Angizi, J. Sun, W. Zhang, and D. Fan, "Aligns: A processing-in-memory accelerator for dna short read alignment leveraging sot-mram," in *Proceedings of the 56th Annual Design Automation Conference 2019*. ACM, 2019, p. 144.

[11] S. Angizi and D. Fan, "Graphide: A graph processing accelerator leveraging in-dram-computing," in *GLSVLSI*. ACM, 2019, pp. 45–50.

[12] S. Angizi, Z. He, F. Parveen, and D. Fan, "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.

[13] S. Angizi, Z. He, A. Awad, and D. Fan, "Mrima: An mram-based in-memory accelerator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019.

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

[15] Z. He *et al.*, "High performance and energy-efficient in-memory computing architecture based on sot-mram," in *NANOARCH*. IEEE, 2017, pp. 97–102.

[16] S. Angizi, Z. He, N. Bagherzadeh, and D. Fan, "Design and evaluation of a spintronic in-memory processing platform for nonvolatile data encryption," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 9, pp. 1788–1801, 2017.

[17] S. Jain *et al.*, "Computing in memory with spin-transfer torque magnetic ram," *IEEE TVLSI*, pp. 470–483, 2018.

[18] V. Seshadri *et al.*, "Ambit: In-memory accelerator for bulk bitwise operations using commodity dram technology," in *Micro*. ACM, 2017, pp. 273–287.

[19] R. Andri, L. Cavigelli, D. Rossi, and L. Benini, "Yodann: An architecture for ultralow power binary-weight cnn acceleration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 48–60, 2017.

[20] S. Li *et al.*, "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in *2016 53nd DAC*. IEEE, 2016.

[21] S. Angizi, J. Sun, W. Zhang, and D. Fan, "Graphs: A graph processing accelerator leveraging sot-mram," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 378–383.

[22] S. Angizi and D. Fan, "Imc: energy-efficient in-memory convolver for accelerating binarized deep neural network," in *Proceedings of the Neuromorphic Computing Symposium*. ACM, 2017, p. 3.

[23] (2011) Ncsu eda freepdk45. [Online]. Available: http://www.eda.ncsu.edu/wiki/FreePDK45:Contents

[24] 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.

[25] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, "Drisa: A dram-based reconfigurable in-situ accelerator," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2017, pp. 288–301.

[26] K. Chen, S. Li, N. Muralimanohar, J. H. Ahn, J. B. Brockman, and N. P. Jouppi, "Cacti-3dd: Architecture-level modeling for 3d die-stacked dram main memory," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2012, pp. 33–38.

[27] S. D. C. P. V. . Synopsys, Inc.

[28] N. Muralimanohar *et al.*, "Cacti 6.0: A tool to model large caches," *HP Laboratories*, pp. 22–31, 2009.