An MRAM-based Deep In-Memory Architecture for Deep Neural Networks

Ameya D. Patil*, Haocheng Hua*, Sujan Gonugondla*, Mingu Kang† and Naresh R. Shanbhag*
*Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801
†IBM T. J. Watson Research Center, Yorktown Heights, NY 10598

***Abstract*—This paper presents an MRAM-based deep in-memory architecture (MRAM-DIMA) to efficiently implement multi-bit matrix vector multiplication for deep neural networks using a standard MRAM bitcell array. The MRAM-DIMA achieves an $4.5\times$ and $70\times$ lower energy and delay, respectively, compared to a conventional digital MRAM architecture. Behavioral models are developed to estimate the impact of circuit non-idealities, including process variations, on the DNN accuracy. An accuracy drop of $\leq 0.5\%$ ($\leq 1\%$) is observed for LeNet-300-100 on the MNIST dataset (a 9-layer CNN on the CIFAR-10 dataset), while tolerating 24% (12%) variation in cell conductance in a commercial 22 nm CMOS-MRAM process.**

I. Introduction

There is a growing interest in implementing deep neural network (DNN) based algorithms in variety of the edge platforms, such as smartphones, IoT sensors, etc [1]. However, conventional digital implementations of DNNs consume large energy and delay per decision primarily due to large data movement [2] requirements, since memory accesses require at least $10\times$ more energy/delay compared to multiply-accumulate (MAC) operations [3]. This inhibits deployment of DNNs in resource-constrained, battery-operated platforms.

Recently, near-memory and in-memory computing approaches [4]–[18] are being widely explored to achieve significant energy-delay benefits over conventional digital implementations [19], [20]. These works use SRAM/MRAM arrays to realize *binary* matrix vector computation within the memory array [4], [8]–[11], and/or employ *modified bitcell circuit* to achieve multi-bit computation at the expense of lower density [5], [7], [12]. As an exception, in [21], [22], an SRAM-based deep in-memory architecture was proposed to efficiently achieve multi-bit *vector dot product* without requiring any modification in standard 6T SRAM bitcell. However, for state-of-the-art DNNs, on-chip SRAM may not be sufficient to store all the parameters, requiring highly energy and latency expensive DRAM accesses. Hence, it is necessary to extend such approach to emerging high density memories, such as MRAM. Works on multi-level RRAM/PCM devices have explored multi-bit in-memory computation [13]–[18]. However, realizing an in-memory MRAM architecture that implements a *multi-bit matrix-vector multiplication (MVM) without modifying bitcell structure* remains a challenge.

In this paper, we propose an MRAM-based deep in-memory architecture (MRAM-DIMA) to achieve multi-bit MVM within the memory array. We employ standard MRAM bitcell without requiring any modifications, thus preserving its density. We propose modified peripheral circuits to achieve such multi-bit computation, even though, unlike RRAM/PCM, each bitcell stores only 1-bit. Proposed MRAM DIMA achieves $4.5\times$ and $70\times$ lower energy and delay, respectively, compared to digital MRAM implementation with the matrix stored in an identical MRAM array. We further quantify accuracy drop due to analog computation and MTJ process variations for LeNet-300-100 on MNIST dataset and a 9-layer CNN on CIFAR-10 dataset and find it to be within 0.5% and 1%, respectively.

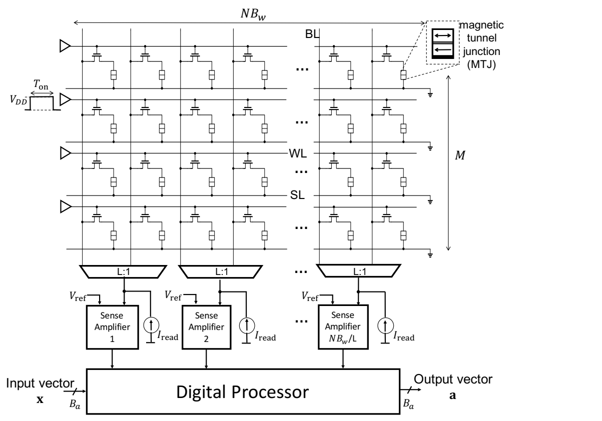


Fig. 1. Digital implementation of matrix-vector multiplication (MVM) with weights stored in 1T-1MTJ MRAM bitcell array (BCA).

II. Preliminaries

A. Notation

In this paper, we assume following matrix vector multiplication (MVM) computation needs to be executed:

$$\mathbf{a} = \mathbf{W}\mathbf{x} \tag{1}$$

where $\mathbf{W}$ denotes a $M \times N$ weight matrix, and $\mathbf{x}$ and $\mathbf{a}$ denote input and output vectors, respectively. Each weight is denoted as $w_{ij}\ \forall\ i \in \{1, \ldots M\}, j \in \{1, \ldots N\}$ and is quantized to B_w bits. Each element of vectors $\mathbf{x}$ and $\mathbf{a}$ is denoted as x_i and a_i, respectively, and quantized to B_a bits.

B. Digital MRAM Architecture

Figure 1 shows a diagram of a conventional digital MRAM architecture with a bitcell array (BCA) of size $N_{\text{row}} \times N_{\text{col}} = M \times NB_w$ which stores the weight matrix $\mathbf{W}$. This BCA has sourceline (SL) and wordline (WL) perpendicular to the bitline (BL). The MRAM bitcell consists of an NMOS access transistor and a magnetic tunnel junction (MTJ) (1T-1MTJ)

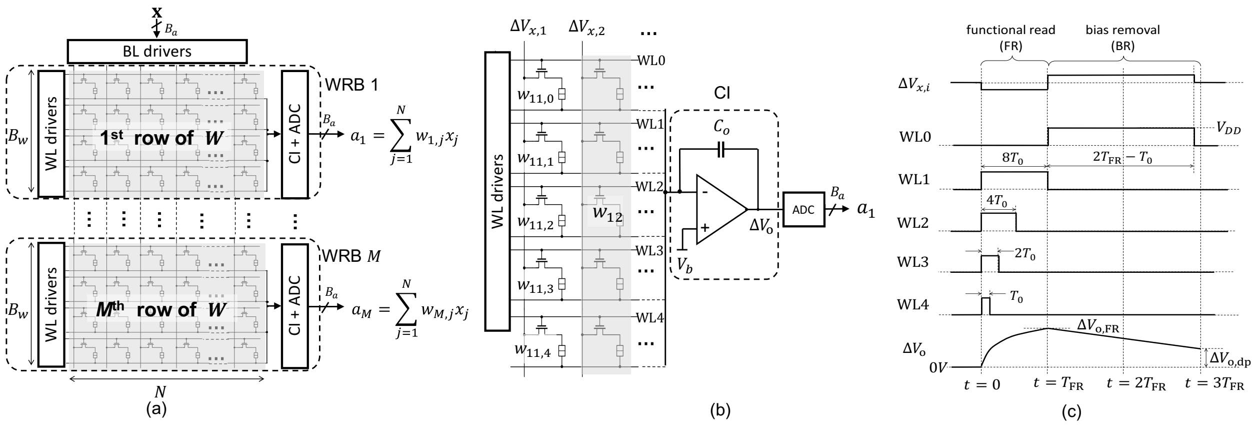


Fig. 2. The proposed MRAM-based deep in-memory architecture (MRAM-DIMA): (a) the overall architecture consisting of M parallel word-row blocks (WRBs) each sharing N BL drivers but possessing separate WL drivers and current integrators (CIs), (b) schematic of a WRB, and (c) a WRB timing diagram.

[23]. The stored bit is read by sensing the resistance across the MTJ (R_{MTJ}), which is low when the two magnetization vectors are parallel to each other (P state, $R_{\text{MTJ}} = R_P$), and is high when they are antiparallel (AP state, $R_{\text{MTJ}} = R_{AP}$). In this paper, bit value 1 (0) corresponds to P (AP) state.

During an MRAM read operation, a constant current of I_{read} is passed through the bitcells and the voltage developed on the BLs is sensed to determine the MTJ state. Access transistors of a single row are activated for T_{on} duration by driving its WL to V_{DD}. A sense amplifier (SA) converts the voltage on the BL to a bit decision. Since the SA is typically wider than a bitcell, it is shared across multiple columns via L : 1 multiplexers (typical value of L is from 8 to 32). Thus, $\frac{N}{L}$ weights are read per cycle, which are input to a digital processor consisting of $\frac{N}{L}$ parallel multipliers followed by an adder tree for computing a dot product.

The average energy required by the MRAM-digital architecture to implement the MVM computation in (1) is given by:

$$E_{\text{digital}} = MNB_w\left[(I_{\text{read}}V_{DD}T_{\text{on}} + E_{SA}) + LC_{\text{wl-cell}}V_{DD}^2\right] + E_{\text{proc}} \tag{2}$$

where $R_{\text{cell}} = R_{\text{mos}} + \frac{1}{2}(R_P + R_{AP})$, E_{SA} and E_{proc} denote the sense amplifier energy and the total digital processor energy, respectively, T_{on} is the ON time of a single row during BCA read operation, and $C_{\text{wl-cell}}$ denotes the WL capacitance per bitcell. Similarly, the delay to complete the MVM operation is given by:

$$T_{\text{digital}} = MLT_{\text{on}} + T_{\text{proc}}, \tag{3}$$

where T_{proc} denotes the delay of the digital processor in Fig. 1.

III. MRAM-based Deep In-memory Architecture (MRAM-DIMA)

A. Overall Architecture

Figure 2(a) shows the MRAM-DIMA consists of a conventional MRAM BCA and peripheral blocks including WL drivers, BL drivers, and current integrators (CIs) on SLs. The weights are stored in the BCA in a column major format. A set of B_w rows of BCA storing one row of $\mathbf{W}$ constitutes a word-row block (WRB). Each WRB implements a vector dot product to compute one element of the output vector $\mathbf{a}$ in (1). All WRBs operate in parallel, sharing the BLs, but possess separate WL drivers and SL CIs, as shown in Fig. 2(a).

B. MRAM-DIMA Operation

In this subsection, we describe the operation of a single WRB in detail. Figure 2(b) shows a schematic of WRB 1 with $B_w = 5$ and $B_a = 4$. Analog voltages $\Delta V_{x,i} = -x_iV_{\text{lsb}}$ are applied to the BLs via per column DACs (V_{lsb} denotes the DAC output resolution). Next, a functional read (FR) [21] step is initiated, in which the bottom $B_w - 1$ WLs are activated simultaneously by applying pulse-width modulated (PWM) access pulses with the bth WL turned ON for a duration of $2^{B_w-b-1}T_0$. Total FR phase duration is $T_{FR} = 2^{B_w-2}T_0$. All SL currents are summed in the CI, followed by an ADC to generate the digital outputs a_is. At the end of the FR phase, the resulting CI output voltage $\Delta V_{\text{o,FR}}$ is given by:

$$\Delta V_{\text{o,FR}} = \frac{T_{\text{o}}V_{\text{lsb}}}{C_o}\left[\underbrace{\Delta G\sum_{j=1}^{N} w_{ij}x_j}_{=\Delta V_{\text{o,dp}}\text{ (dot product)}} + \underbrace{(2^{B_w-1}-1)\sum_{j=1}^{N} G_jx_j}_{=\Delta V_{\text{o,bias}}\text{ (bias)}}\right] \tag{4}$$

where $\Delta G = G_P - G_{AP}$, $G_P = \frac{1}{R_P + R_{\text{mos}}}$, $G_{AP} = \frac{1}{R_{AP} + R_{\text{mos}}}$, C_o denotes the capacitor in CI, and $G_j = G_{AP}$ ($= G_P$), when $w_{ij} \geq 0$ ($w_{ij} < 0$).

The bias term $\Delta V_{\text{o,bias}}$ in (4) is generated by the non-zero bitcell current when storing a zero-valued bit. This bias term is removed in bias removal (BR) step which discharges ΔV_{o} by enabling only WL0 as shown in Fig. 2(c).

C. Scaled-up MRAM-DIMA

The current integrator in Fig. 2(b) has a finite output swing, i.e., $\Delta V_{\text{o}} \leq \Delta V_{\text{o,max}}$. This output swing limitation is exploited to naturally implement clipped ReLU activation function in DNNs. However, in order to achieve correct computation, it is also necessary that $\Delta V_{\text{o,FR}} < \Delta V_{\text{o,max}}$. This condition is

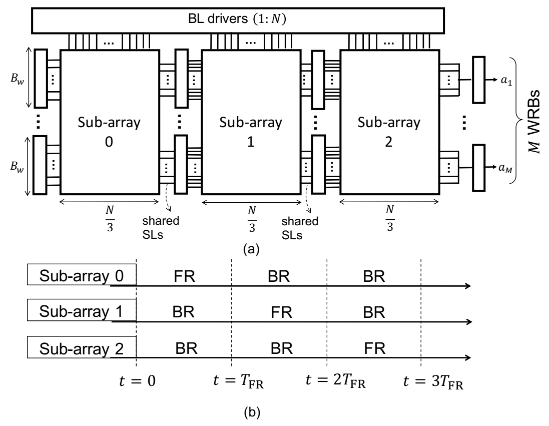


Fig. 3. Realizing scaled-up MRAM-DIMA via phase multiplexed computations: (a) architecture, and (b) timing diagram.

challenging to meet as vector dimension N increases (see (4)), since $\Delta V_{\text{o,bias}}$ gets canceled only during the BR step.

To design MRAM-DIMA for large N, we propose swing budgeting via *phase multiplexed computation* within each WRB. We partition the $MB_w \times N$ BCA vertically into three sub-arrays, each consisting of $\frac{N}{3}$ columns as shown in Fig. 3(a). All three sub-arrays share their SLs and CIs. However, we introduce distinct WL drivers for individual sub-arrays to operate two in the BR phase, while the third in FR step as shown in Fig. 3(b). This approach drastically reduces the resulting $\Delta V_{\text{o,bias}}$ at $t = T_{\text{FR}}$ and $t = 2T_{\text{FR}}$. For LeNet-300-100 DNN on MNIST dataset, it is estimated that $\Pr\{\Delta V_{\text{o,bias}} > \Delta V_{\text{o,max}}\} < 0.01\%$ at any given time instant using the behavioral models developed in sec III-D, resulting in negligible system-level accuracy drop.

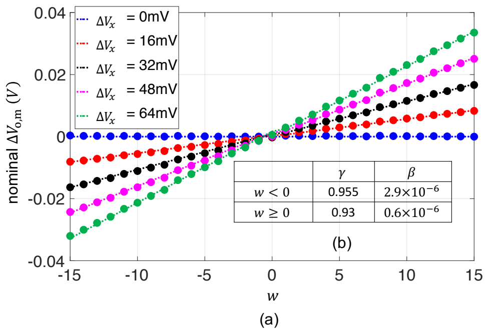
D. Modeling Energy, Delay, and MTJ Process Variations

The average energy consumption of MRAM-DIMA for implementing $M \times N$ MVM is:

$$E_{\text{dima}} = MNB_w\left[\left(\frac{2^{B_w}-2}{B_w}\right)\bar{x}V_{\text{lsb}}G_{\text{cell}}V_{DD}T_0 + C_{\text{wl-cell}}V_{DD}^2\right] + ME_{\text{adc}} + ME_{\text{CI}} + NE_{\text{dac}} \tag{5}$$

where $G_{\text{cell}} = \frac{G_P+G_{AP}}{2}$, $\bar{x}$ denotes average value of $\mathbf{x}$ across its N components, E_{adc}, E_{CI}, and E_{dac} denote the energy consumed in the ADC, CI, and DAC, respectively. The DAC is assumed to be equipped with an opamp-based voltage follower to drive the required current in BL [24]. Hence, E_{dac} and E_{CI} are dominated by the respective opamp bias current energies. The value of $\bar{x}$ is estimated from the distribution of activations in DNNs. Compared to the digital MRAM energy consumption (in (2)), the BCA energy is reduced due to two factors, one, the small value of $\bar{x}$, thanks to high activation sparsity in DNNs, caused by ReLU, and second, reduced read current per bitcell, since multiple bitcell currents are aggregated before ADC operation. The MRAM-DIMA delay is:

$$T_{\text{dima}} = 3 \times 2^{B_w-2}T_0 + T_{\text{adc}} + T_{\text{dac}} \tag{6}$$



	γ	β
$w < 0$	0.955	2.9×10^{-6}
$w \geq 0$	0.93	0.6×10^{-6}

Fig. 4. (a) Simulated (circles) and modeled (lines) transfer function of a scalar multiplier in WRB operating on two operands w and $x \propto \Delta V_x$ for $B_w = 5$ in a commercial 22 nm CMOS-MRAM process, and (b) estimated fitting parameters.

where T_{adc}, T_{dac} denotes the delay of ADC and the DAC, respectively. The delay of MRAM-DIMA is nearly constant, resulting in the speed-up increasing with M.

The expression in (4) for an analog vector dot product output $\Delta V_{\text{o,dp}}$ ignores the impact of circuit non-idealities, such as body-effect of access transistors, virtual ground voltage bounce in the current integrator, and process variations in the MTJ. To account for these non-idealities, we model the analog output of a multiplier with a B_w-bit operand w and an analog input x in the WRB as follows:

$$\Delta V_{\text{o,m}} = \frac{\gamma T_0 V_{\text{lsb}}}{C_o}\left[\Delta G w + (2^{B_w-1}-1)\beta\right]x + \eta \tag{7}$$

where η denotes the spatial noise arising due to process variations in the MTJ. The fitting parameters β and γ are obtained via circuit simulations in a commercial 22 nm CMOS-MRAM process (see in Fig. 4).

Process variations at the multiplier output arises from R_{MTJ} variations across different bitcells. Hence, η in (7) follows a zero-mean Gaussian distribution with the variance σ_η^2 given by:

$$\sigma_\eta^2 = \left(\frac{T_0V_{\text{lsb}}}{C_0}x\right)^2\left((2^{B_w-1}-1)^2G_0^2 + \sum_{b=1}^{B_w-1}4^{b-1}{G_b}^2\right)\left(\frac{\sigma}{\mu}\right)^2_{\text{G-bc}} \tag{8}$$

where $\left(\frac{\sigma}{\mu}\right)_{\text{G-bc}}$ denotes σ-to-μ ratio of G_{cell}, and $G_b \in \{G_P, G_{AP}\}$ is the conductance of the b-th cell in a column, where $b \in \{0, \ldots, B_w-1\}$. It is to be noted that both nominal $\Delta V_{\text{o,m}}$ and σ_η scale linearly with T_0 and V_{lsb} keeping the signal-to-noise ratio unchanged. We estimate $\left(\frac{\sigma}{\mu}\right)_{\text{G-bc}} \leq 6\%$ via Monte Carlo circuit simulations for both P and AP states.

IV. Simulation Results

A. Design Choices

In this paper, we choose $B_w = 5$ and $B_a = 4$, since these are typical DNN precision requirements for inference [25], [26]. For MRAM-DIMA, we assume a single-slope ADC architecture with shared ramp generation circuit across the WRBs, typically employed in column-parallel ADCs in CMOS image sensors [27]. From the measured 8 bit, 7.1 MS/s 65 nm single-slope ADC results in [21], we conservatively

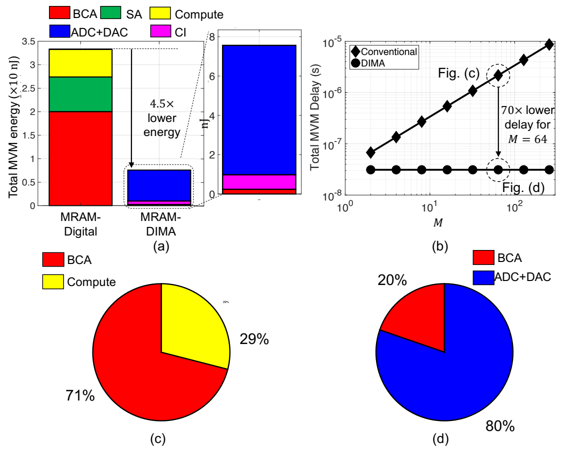


Fig. 5. Simulation results for MVM operation: (a) energy breakdown, (b) delay as a function of M, (c) component-wise delay contributions in digital MRAM, (d) component-wise delay contributions in MRAM-DIMA for $M \times N = 64 \times 576$, $B_w = 5$, and $B_a = 4$.

estimate $E_{\text{adc}} = 0.84\,\text{pJ/sample}$ and $T_{\text{adc}} + T_{\text{dac}} = 25\,\text{ns}$ for 4 bit precision, with T_{adc} dominating the delay. Furthermore, assuming $\Delta V_{\text{o,max}} = \frac{V_{DD}}{3} = 300\,\text{mV}$, $V_{\text{lsb}} = 4\,\text{mV}$, and using $C_o = 200\,\text{fF}$, the system-level output swing requirements for mapping DNN computations described in sec IV-C dictate $T_0 = 256\,\text{ps}$, which is previously shown to be achievable [22], [23], [28] without significant energy overhead.

In the digital MRAM architecture, we use a BCA of identical capacity as in DIMA to store the weight matrix. Using the measured offset voltage variation in SA [29] and observed $\left(\frac{\sigma}{\mu}\right)_{\text{G-bc}}$ in the PDK, we derive $I_{\text{read}} = 40\,\mu\text{A}$. Similarly, we assume $T_{\text{on}} = 3\,\text{ns}$, $L = 8$, and $E_{SA} = 40\,\text{fJ}$ from the results reported in [29]–[32]. We obtain energy and delay of a full-adder via SPICE simulations and use it to estimate E_{proc} and T_{proc} in (2) and (3), respectively.

B. Energy and Throughput benefits

In Fig. 5, we plot energy and delay models in Sec III-D for $M \times N = 64 \times 576$ MVM computation corresponding to a 3×3 convolutional layer with 64 input and output channels. The MRAM-DIMA achieves a $4.5\times$ lower energy as shown in Fig. 5(a), primarily due to elimination of SA, and BCA energy reduction due to activation sparsity and reduced read current per bitcell. In MRAM-DIMA, E_{dac} and E_{CI} dominate due to their large opmap bias currents. The aggregated maximum current on SLs in a WRB is smaller than maximum current on single BL due to activation sparsity and phase multiplexed bias removal. This leads to smaller bias current energy in E_{CI} than that of the voltage follower in E_{dac}. The delay benefits of MRAM-DIMA over digital MRAM scale linearly with M due to the inherent parallelism in MRAM-DIMA (see Fig. 5(b)). A delay reduction of up to $70\times$ is achieved for $M = 64$. While memory access delay dominates in digital MRAM, (see Fig. 5(c)), T_{adc} constitutes 80% of T_{dima} (Fig. 5(d)).

C. Impact on System-level Accuracy

We estimate the impact of MRAM-DIMA's analog computation and process variations on the system-level accuracy

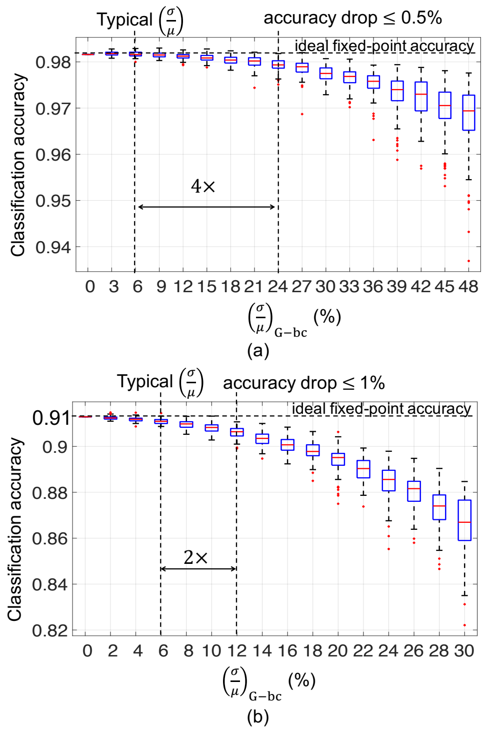


Fig. 6. Classification accuracy vs $\left(\frac{\sigma}{\mu}\right)_{\text{G-bc}}$ obtained using the models in Sec III-D for: (a) LeNet-300-100 on MNIST dataset, and (b) 9-layer CNN on CIFAR-10 dataset. For each value of $\left(\frac{\sigma}{\mu}\right)_{\text{G-bc}}$, 100 instances of the BCA were generated to obtain the statistics shown in the error bars.

of: (a) LeNet-300-100 MLP on MNIST dataset, and (b) a 9-layer CNN on CIFAR-10 dataset. These networks were trained using DoReFa-net [33] training methodology. The activations pass through a clipped ReLU activation function and are quantized between $[0, 1]$. The weights lie in the range between $[-\alpha_{ij}, +\alpha_{ij}]$, where α_{ij} is the scaling parameter for the i-th filter in the j-th layer, and they all lie in the range $[0.03, 0.15]$. Limitation on the voltage swing at the CI output ($\Delta V_{\text{o,max}}$) naturally implements clipped ReLU, while the α_{ij}s are implemented by appropriately scaling T_0. We emulate MRAM-DIMA's computation in PyTorch by using nominal behavioral models described in Sec III-D for deterministic non-idealities and add bitwise random noise in bitcell conductance values to model MTJ variations. The MRAM-DIMA is able to tolerate a $4\times$ and $2\times$ higher $\left(\frac{\sigma}{\mu}\right)_{\text{G-bc}}$ than its typical value while maintaining $< 1\%$ drop in accuracy compared to ideal fixed-point computation for LeNet-300-100 and 9-layer CNN as shown in Fig. 6(a) and (b), respectively.

V. Discussion

Future work will investigate design optimization, layout and pitch-matching constraints, and architectural dataflow mappings to realize efficient DNN system implementations composed of the proposed MRAM-DIMA MVM tiles.

Acknowledgment

This work was supported by DARPA FRANC Program.

References

[1] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, and F. Kawsar, "An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices," in *Proceedings of the 2015 international workshop on internet of things towards applications*. ACM, 2015, pp. 7–12.

[2] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*. IEEE, 2016, pp. 367–379.

[3] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *IEEE International Solid-State Circuits Conference-(ISSCC)*, 2014, pp. 10–14.

[4] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A mixed-signal binarized convolutional-neural-network accelerator integrating dense weight storage and multiplication for reduced data movement," in *VLSI Symp. on Circuits (VLSIC)*. IEEE, 2018.

[5] 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 *IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 2018, pp. 488–490.

[6] R. Liu, X. Peng, X. Sun, W.-S. Khwa, X. Si, J.-J. Chen, J.-F. Li, M.-F. Chang, and S. Yu, "Parallelizing SRAM arrays with customized bit-cell for binary neural networks," in *Proceedings of the 55th Annual Design Automation Conference*. ACM, 2018, p. 21.

[7] A. Jaiswal, I. Chakraborty, A. Agrawal, and K. Roy, "8T SRAM cell as a multi-bit dot product engine for beyond von-neumann computing," *arXiv preprint arXiv:1802.08601*, 2018.

[8] S. Jain, A. Ranjan, K. Roy, and A. Raghunathan, "Computing in memory with spin-transfer torque magnetic RAM," *arXiv preprint arXiv:1703.02118*, 2017.

[9] F. Parveen, S. Angizi, Z. He, and D. Fan, "Low power in-memory computing based on dual-mode sot-mram," in *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2017, pp. 1–6.

[10] F. Parveen, Z. He, S. Angizi, and D. Fan, "HielM: Highly flexible in-memory computing using STT MRAM," in *Design Automation Conference (ASP-DAC)*. IEEE, 2018, pp. 361–366.

[11] Z. He, S. Angizi, and D. Fan, "Exploring STT-MRAM based in-memory computing paradigm with application of image edge extraction," in *IEEE International Conference on Computer Design (ICCD)*. IEEE, 2017, pp. 439–446.

[12] Y. Pan, P. Ouyang, Y. Zhao, W. Kang, S. Yin, Y. Zhang, W. Zhao, and S. Wei, "A multilevel cell STT-MRAM-based computing in-memory accelerator for binary convolutional neural network," *IEEE Transactions on Magnetics*, no. 99, pp. 1–5, 2018.

[13] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2016, pp. 14–26.

[14] 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/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2016, pp. 27–39.

[15] L. Song, X. Qian, H. Li, and Y. Chen, "Pipelayer: A pipelined ReRAM-based accelerator for deep learning," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2017, pp. 541–552.

[16] M. Le Gallo, A. Sebastian, R. Mathis, M. Manica, H. Giefers, T. Tuma, C. Bekas, A. Curioni, and E. Eleftheriou, "Mixed-precision in-memory computing," *Nature Electronics*, vol. 1, no. 4, p. 246, 2018.

[17] I. Boybat, M. Le Gallo, S. Nandakumar, T. Moraitis, T. Parnell, T. Tuma, B. Rajendran, Y. Leblebici, A. Sebastian, and E. Eleftheriou, "Neuromorphic computing with multi-memristive synapses," *Nature communications*, vol. 9, no. 1, p. 2514, 2018.

[18] A. Sebastian, T. Tuma, N. Papandreou, M. Le Gallo, L. Kull, T. Parnell, and E. Eleftheriou, "Temporal correlation detection using computational phase-change memory," *Nature Communications*, vol. 8, no. 1, p. 1115, 2017.

[19] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.

[20] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun *et al.*, "DaDianNao: A machine-learning supercomputer," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2014, pp. 609–622.

[21] M. Kang, S. K. Gonugondla, A. Patil, and N. R. Shanbhag, "A multi-functional in-memory inference processor using a standard 6T SRAM array," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, 2018.

[22] S. K. Gonugondla, M. Kang, and N. R. Shanbhag, "A variation-tolerant in-memory machine learning classifier via on-chip training," *IEEE Journal of Solid-State Circuits*, no. 99, pp. 1–11, 2018.

[23] S. H. Kang and S.-O. Jung, "Embedded STT-MRAM: Device and design," in *More than Moore Technologies for Next Generation Computer Design*. Springer, 2015, pp. 73–99.

[24] Y. Long, T. Na, and S. Mukhopadhyay, "Reram-based processing-in-memory architecture for recurrent neural network acceleration," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, no. 99, pp. 1–14, 2018.

[25] C. Sakr, Y. Kim, and N. Shanbhag, "Analytical guarantees on numerical precision of deep neural networks," in *International Conference on Machine Learning*, 2017, pp. 3007–3016.

[26] B. Moons, K. Goetschalckx, N. Van Berckelae, and M. Verhelst, "Minimum energy quantized neural networks," in *Asilomar conference on Signals, Systems and Computer*, 2017.

[27] M. F. Snoeij, A. J. Theuwissen, K. A. Makinwa, and J. H. Huijsing, "Multiple-ramp column-parallel ADC architectures for CMOS image sensors," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 12, pp. 2968–2977, 2007.

[28] J. Zhang, Z. Wang, and N. Verma, "In-memory computation of a machine-learning classifier in a standard 6t sram array." *J. Solid-State Circuits*, vol. 52, no. 4, pp. 915–924, 2017.

[29] Q. Dong, Z. Wang, J. Lim, Y. Zhang, Y.-C. Shih, Y.-D. Chih, J. Chang, D. Blaauw, and D. Sylvester, "A 1Mb 28nm STT-MRAM with 2.8 ns read access time at 1.2 V VDD using single-cap offset-cancelled sense amplifier and in-situ self-write-termination," in *IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2018, pp. 480–482.

[30] J. Kim, K. Ryu, S. H. Kang, and S.-O. Jung, "A novel sensing circuit for deep submicron spin transfer torque mram (stt-mram)," *IEEE Transactions on very large scale integration (VLSI) systems*, vol. 20, no. 1, pp. 181–186, 2012.

[31] Q.-K. Trinh, S. Ruocco, and M. Alioto, "Dynamic reference voltage sensing scheme for read margin improvement in STT-MRAMs," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 4, pp. 1269–1278, 2018.

[32] L. Bagheriye, S. Toofan, R. Saeidi, and F. Moradi, "A novel sensing circuit with large sensing margin for embedded spin-transfer torque MRAMs," in *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*. IEEE, 2018, pp. 1–5.

[33] 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.