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× and 70× 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 ≤0.5% (≤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× 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× and 70× 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.

II. PRELIMINARIES

A. Notation

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

\[ \mathbf{a} = \mathbf{Wx} \]  

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 \( w_{ij} \) for \( i \in \{1, \ldots, M\}, j \in \{1, \ldots, N\} \) 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 source line (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)
[23]. The stored bit is read by sensing the resistance across the MTJ \( R_{\text{MTJ}} \), which is low when the two magnetization vectors are parallel to each other (P state, \( R_{\text{MTJ}} = R_{\text{AP}} \)), and is high when they are antiparallel (AP state, \( R_{\text{MTJ}} = R_{\text{AP}} \)). In this paper, bit value \( R \) is read.

During an MRAM read operation, a constant current of \( I_{\text{read}} \) 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 \( T_{\text{on}} \) 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:

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

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

\[
T_{\text{digital}} = M L T_{\text{on}} + T_{\text{proc}}, \tag{3}
\]

where \( T_{\text{proc}} \) 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 \( B_{\text{w}} \) rows of BCA storing one row of \( W \) constitutes a word-row block (WRB). Each WRB implements a vector dot product to compute one element of the output vector \( a \) 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 \( B_{\text{w}} = 5 \) and \( B_{\text{a}} = 4 \). Analog voltages \( \Delta V_{\text{zij}} = -x_i V_{\text{lsb}} \) are applied to the BLs via per column DACs (\( V_{\text{lsb}} \) denotes the DAC output resolution). Next, a functional read (FR) [21] step is initiated, in which the bottom \( B_{\text{w}} - 1 \) WLs are activated simultaneously by applying pulse-width modulated (PWM) access pulses with the \( b \)th WL turned ON for a duration of \( 2 B_{\text{w}} - 1 - T_{\text{on}} \). Total FR phase duration is \( T_{\text{FR}} = 2 B_{\text{w}} - 2 T_{\text{on}} \). All SL currents are summed in the CI, followed by an ADC to generate the digital outputs \( a_i \). At the end of the FR phase, the resulting CI output voltage \( V_{o,FR} \) is given by:

\[
V_{o,FR} = \frac{T_{\text{on}} V_{\text{lsb}}}{C_o} \left[ \Delta G \sum_{j=1}^{N} w_{ij} x_j + (2 B_{\text{w}}^2 - 1) \sum_{j=1}^{N} G_j x_j \right]
= \Delta V_{o,\text{dot}} \text{(dot product)} + \Delta V_{o,\text{bias}} \text{(bias)} \tag{4}
\]

where \( \Delta G = G_P - G_{\text{AP}}, G_P = \frac{1}{R_{\text{P}} + R_{\text{max}}}, G_{\text{AP}} = \frac{1}{R_{\text{AP}} + R_{\text{max}}}, C_o \) denotes the capacitor in CI, and \( G_j = G_{\text{AP}} = G_P \), when \( w_{ij} \geq 0 \). The bias term \( \Delta V_{o,\text{bias}} \) 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 \( V_o \) 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., \( \Delta V_o \leq \Delta V_{o,\text{max}} \). 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 \( \Delta V_{o,\text{FR}} < \Delta V_{o,\text{max}} \). This condition is
implementing sub-arrays to operate two in the BR phase, while the third
However, we introduce distinct WL drivers for individual
Pr
LeNet-300-100 DNN on MNIST dataset, it is estimated that
within each
Sub-array 0FRBR
Sub-array 1BRFR
Sub-array 2BRFR
\begin{align}
E_{\text{dima}} &= MNB_w \left[ \left( \frac{9B_w - 2}{B_w} \right) \bar{x} \text{V}_{\text{th}} G_{\text{cell}} V_{\text{DD}} T_0 \right. \\
& + C_{\text{wl-cell}} V_{\text{DD}}^2 \left] + ME_{\text{adc}} + ME_{\text{CI}} + NE_{\text{dac}} \right.
\end{align}
where \(G_{\text{cell}} = \frac{G_{\mu} + G_{\Delta \mu}}{2}\), \(\bar{x}\) denotes average value of \(x\) across
its
weighted
sharing
SLs
shared
SLs
M WRBs
\begin{align}
\Delta V_{o,\text{max}} &= \frac{\gamma T_0 V_{\text{th}}}{C_o} \left[ \Delta G w + (2B_w^{-1} - 1) \beta \right] x + \eta
\end{align}
where \(\eta\) denotes the spatial noise arising due to process variations in the
MTJ. The fitting parameters \(\beta\) and \(\gamma\) are obtained
via circuit simulations in a commercial 22 nm CMOS-MRAM process
process variations in the BCA vertically into
\begin{align}
\Delta V_{o,\text{bias}} &= \Delta V_{o,\text{bias}}[t = T_{\text{FR}} & \text{ and } t = 2T_{\text{FR}}] \text{ for }
\text{LeNet-300-100 DNN on MNIST dataset, it is estimated that }
\end{align}
\begin{align}
T_{\text{dima}} &= 3 \times 2^{B_w^{-2} - 2} T_0 + T_{\text{adc}} + T_{\text{dac}}
\end{align}
where \(T_{\text{adc}}, T_{\text{dac}}\) denotes the delay of ADC and the DAC, respectively. The delay of MRAM-DIMA is nearly constant, resulting in the speed-up increasing with \(M\).

To design MRAM-DIMA for large \(N\), we propose swing
budgeting via *phase multiplexed computation* within each

\[ \sum_{i=0}^{n-1} A_i \times B_i \]

where \(T_{\text{adc}}, T_{\text{dac}}\) 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 \(\Delta V_{o,\text{dp}}\) 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:

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


