

# System and Design Technology Co-optimization of SOT-MRAM for High-Performance AI Accelerator Memory System

Kaniz Mishty and Mehdi Sadi, *Member, IEEE*,

**Abstract**—System on Chips (SoCs) are now designed with their own AI accelerator segment to accommodate the ever-increasing demand of Deep Learning (DL) applications. With powerful Multiply and Accumulate (MAC) engines for matrix multiplications, these accelerators show high computing performance. However, because of limited memory resources (i.e., bandwidth and capacity), they fail to achieve optimum system performance during large batch training and inference. In this work, we propose a memory system with high on-chip capacity and bandwidth to shift the gear of AI accelerators from memory-bound to achieving system-level peak performance. We develop the memory system with Design Technology Co-optimization (DTCO)-enabled customized Spin Orbit Torque (SOT)-MRAM as large on-chip memory through System Technology Co-optimization (STCO) and detailed characterization of the DL workloads. Our workload-aware memory system achieves  $8\times$  energy and  $9\times$  latency improvement on Computer Vision (CV) benchmarks in training and  $8\times$  energy and  $4.5\times$  latency improvement on Natural Language Processing (NLP) benchmarks in training while consuming only around 50% of SRAM area at iso-capacity.

**Index Terms**—DTCO, STCO, AI Accelerator, SOT-MRAM.

## I. INTRODUCTION

THE proliferation of Artificial Intelligence (AI) and Deep Learning (DL) has precipitated the computing hardware community to continually design innovative AI/DL accelerators with large data processing capabilities. Research shows that the AI/DL model accuracy improves as training data set size grows [1]. With increasing data set, model size also grows. Consequently, memory demand in AI/DL accelerators will also grow asymptotically linearly with model and data size [1] [2]. As a result, the bottleneck for state-of-the-art AI/DL models in the accelerator hardware is now memory rather than data and compute availability, and we expect this trend to worsen in the future [2] [3] [4].

The lack of efficient and high-performance data flow between the computing and memory element (i.e., the memory wall or memory bottleneck) masks the improvement coming from the efficient compute system [5]. One promising solution to the memory bottleneck of AI-specific workload is to increase the on-chip memory capacity [6]. For both training and inference, the on-chip memory capacity in the accelerator needs to be

increased to ensure that the intermediate activations, as well as the weights of the current layer, can be loaded. Moreover, significantly more memory is required during training to store the gradients and optimizer states. Inadequate on-chip memory capacity causes frequent DRAM accesses which exacerbates energy costs, as well as stalls the compute cores of AI/DL accelerator until the data is fetched. Because of this large capacity demand, an SRAM-based on-chip memory system can be detrimental due to leakage energy and area inefficiency.

The promising features, such as high density, near-zero leakage power, immunity against radiation-induced soft errors, and CMOS compatibility of emerging Spin-based non-volatile (NVM) magnetic memory (i.e., MRAM) technologies, attracted researchers from academia and industry [7]. Spin Transfer Torque (STT) MRAM, has already shifted its gear from the R&D phase to commercialization as the NAND-based embedded flash replacement [8] [9]. However, MRAM in its regular form cannot be used in AI accelerators due to its slow write speed and high write energy [9] [10].

STT-MRAM, a two-terminal magnetic memory with Magnetic Tunnel Junction (MTJ) as the storing element, flows a bidirectional spin-polarized current through the MTJ for read-write operation [11]. The major challenges of STT-MRAM - poor write performance, Read Disturbance (RD), retention failure, [9] [12] - stem from two main reasons. First, the high write current flowing through the MTJ accounts for almost  $10\times$  energy consumption as SRAM. Large write delay ( $> ns$  range) resulting from spin injection symmetry in switching the magnetic orientation of free layer belittles STT-MRAM's feasibility as an on-chip cache [13]. The stress on the dielectric oxide of the MTJ due to the large write current accelerates the time-dependent wear out of the cell [14]. Second, its shared read-write path makes it vulnerable to RD.

SOT MRAM, considered the next generation of STT-MRAM, offers high performance without compromising reliability issues such as RD. SOT-MRAM is a three-terminal memory cell that uses MTJ as the storing element [15]. By splitting the read-write path and using a different switching scheme, SOT-MRAM resolves all the challenges of STT-MRAM while retaining its every benefit [9] [12] [13] [14] [16]. Isolate read and write path allows the designer to optimize the read and write path independently, decreasing the write current and increasing the read-write operating margin, thus solving the RD-induced reliability issues. Though lacking mass-scale production from foundries due to early-stage manufacturing challenges, [9] [10] [12] [13] [16] [17] have demonstrated the successful fabrication

The authors are with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849 USA

E-mail: kzm0114@auburn.edu; mehdi.sadi@auburn.edu

This work was supported in part by the National Science Foundation (NSF) under Grant Number CRII-2153394.

Manuscript received June 26, 2023; revised October 06, 2023 and November 11, 2023. Accepted November 14, 2023

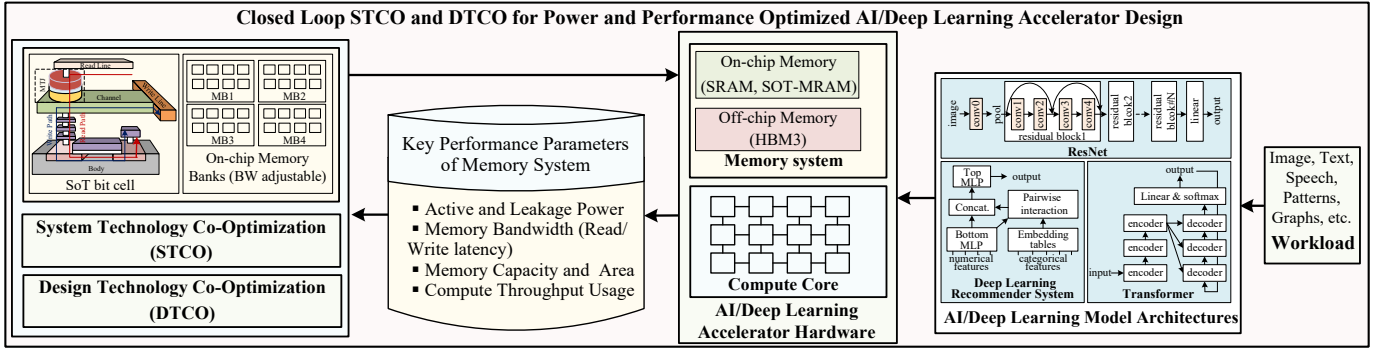


Fig. 1. Workflow of closed-loop analysis for system and device level optimization for AI/Deep Learning Accelerator Design

of SOT-MRAM with attractive specifications. Its attractive features, such as high density, reliability and endurance, zero leakage, read-write latency comparable to SRAM, and research effort to enable mass production make it one of the best candidates for AI accelerator memory system where large on-chip memory is a must for training and inference.

The performance of an AI accelerator depends on both the compute and memory throughput of the device. While most accelerators have enough compute throughput, their performance is limited by memory throughput operating in the *memory bound* region. To address the *memory bound* problem of the AI hardware, in this paper, we perform a closed-loop STCO on AI workloads and DTCO on SOT-MRAM to present a hybrid memory system. To our knowledge, this is the first work that analyzes and evaluates the performance of SOT-MRAM as the on-chip memory of AI accelerators targeting both inference and training. The STCO-DTCO methodology is shown in Fig. 1, and the key contributions of the paper are highlighted as follows.

- We present a power and performance-optimized hybrid memory system for Deep Learning (DL) accelerators through a workload-aware STCO and DTCO. Comprised of off-chip HBM3 DRAM, on-chip SRAMs, and DTCO-enabled SOT-MRAM, the hybrid memory system can support the training and inference of DL workloads. We perform a closed-loop STCO and DTCO by taking into account the (i) System performance attributes (e.g., throughput and energy cost); (ii) Architectural and micro-architectural attributes (e.g., compute resources utilization, memory bandwidth) (iii) Workload attributes at both training and inference (e.g., runtime action counts, dataflow and data reuse) to reach the Pareto optimal solution.
- Using the Deep Learning models' execution profiles, DTCO enables device and circuit level customization of read/write bandwidth, retention time, and capacity of SOT-MRAM memory banks to meet the bandwidth and capacity demands of DL workloads. To achieve dynamic runtime optimization of the power and performance of the accelerator hardware for diverse workloads, memory banks are individually optimized with various bandwidths and capacities.
- Finally, using various DNN benchmarks, we provide a comparative analysis of the existing SRAM-based memory

system and the proposed DTCO-STCO optimized hybrid memory system for AI accelerators.

The rest of the article is organized as follows. Section II discusses the background. In Section III, we present the analytical model for DNN workload profiling, followed by the DTCO of SOT-MRAM in Section IV. Sections V and VI present the results & analysis, and related works, respectively, following the conclusion in Section VII.

## II. BACKGROUND

### A. AI/DL Applications

1) *Computer Vision (CV) and Pattern Recognition*: CV models, also called Convolutional/Deep Neural Networks (CNN/DNN), are the stacks of convolution layers connected straight and/or through residual connection [18] to extract the objects' features, and a few Fully Connected (FC) layers at the end to classify the objects. Image classification, captioning, reconstruction and object/instance segmentation are the scopes of CV models. Deep Residual Networks, having convolutional layers at their core, dominate the CV domain. The input images are convolved with the filter weights to produce the output feature map (*OFMAP*). The *OFMAP* goes through the pooling and normalization layers to act as input (*IFMAP*) to the next layer. The linear and softmax layer at the end finally recognizes the image (Fig. 2). The size of each data entity (*IFMAP*, *OFMAP*, and Weights) depend on the model architecture.

2) *Natural Language Processing (NLP)*: Language modeling deals with processing sequential data. Recurrent Neural Networks (RNN), Long Short Term Memory (LSTM), and Gated Recurrent Unit (GRU) have been used in language modeling until the state-of-the-art Transformer [19] model is introduced. NLP models are used in machine translation, text summarization, speech recognition, syntactic and semantic parsing, question answering, dialog system etc. In Transformer-based models [19], the input sequence propagates through the embedding layer and different sublayers of the encoder stacks to extract different linguistic features and inter-token dependency of the input sequence. The decoder stacks then generate the output sequence by taking the encoded input sequence from the encoder stack and the output sequence generated by itself in the previous timesteps (Fig 3). The input sequence multiplied by different layer weights takes different activation names and shapes throughout the model operation.

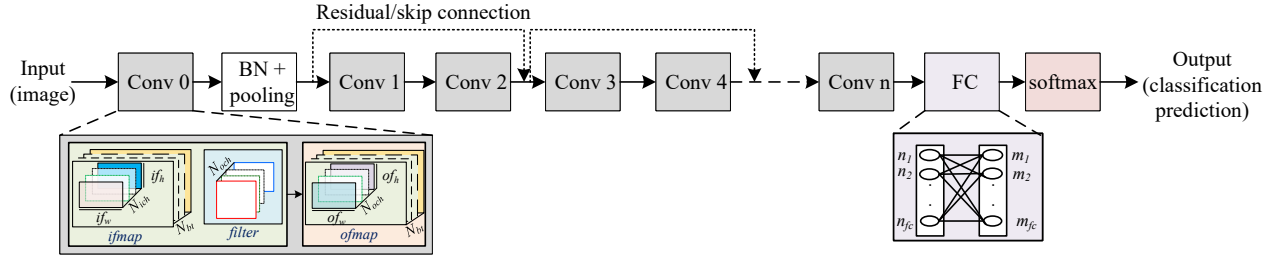


Fig. 2. CV model (CNN/DNN) abstract architecture. Deep convolution (Conv) layers with residual/skip connection followed by fully connected (FC) layer/s. For symbol meaning please see Table I.

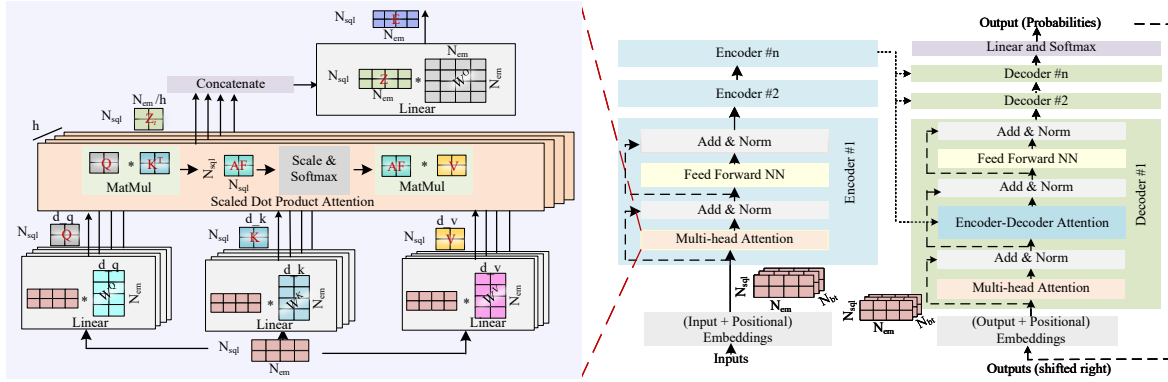


Fig. 3. Transformer model workflow breakdown

## B. AI/DL Accelerators

At the core of AI/DLs is the matrix-matrix/vector multiplication (GEMM) with massive parallelism. Exploiting this parallelism, Systolic Array (SA) based architecture [3] have been used to accelerate the computations. Different dataflows, such as row stationary, output stationary, weight stationary, have been evolved to maximize the reuse and reduce the data movement. Off-chip DRAM access being 100-200 times more energy and latency expensive than any ALU operation or on-chip access [20] plays a crucial role in determining the overall system performance. Another non-conventional type of architecture, In-Memory Computing (IMC) [21] has recently evolved to address the data communication cost for DNN accelerators. However, in this work, we focus on reducing the off-chip memory access for conventional DNN accelerator architectures [3], [20] [22] by increasing the on-chip Global Buffer (GLB) size with SOT-MRAM.

## C. SOT-MRAM

1) *Physical Structure*: With MTJ [11] as storing element, the SOT-MRAM is a three terminal device. Depending on the type of bit cell, there are three to four lines to control the read-write operation. In this work, we consider a two transistor one SOT (2T1SOT) bit cell architecture that requires two access transistors, (i) *Read Wordline (RWL)*, (ii) *Write Wordline (WWL)*, (iii) *Bit Line (BL)*, and (iv) *source Line (SL)* to accommodate separate read-write access path [15] [23] (Fig. 4). The MTJ stack, with its free layer at the interface, is placed on top of a SOT layer (i.e., channel) to ensure SOT-induced

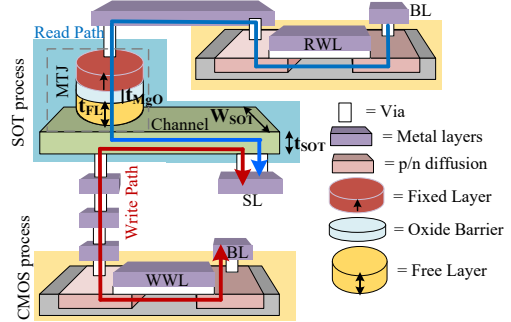


Fig. 4. Physical structure of a SOT-MRAM bit cell highlighting separate read (along blue line) and write (along red line) path

switching. The SOT layer is composed of heavy metals or topological insulators [24].

2) *Read-Write Operation*: Upon the activation of RWL, a small amount of current is passed through BL and grounded SL. The resistive state of the MTJ is captured by sensing the voltage across it and comparing the voltage with a reference value [12]. Low resistive state ( $R_P$ ) and high resistive state ( $R_{AP}$ ) represents bit 0 and 1 respectively. The write operation of MTJ-based MRAM involves switching the resistive status of MTJ. In SOT-MRAM, switching occurs due to Spin Orbit Torque (SOT) effect. Unlike STT-MRAM, a current is passed through the SOT layer to change the MTJ resistive state by switching the magnetic orientation of the free layer. A bidirectional write current flows through BL and SL during write operation. The potential of BL and SL changes depending on the bit value written in the cell. For example, to write '1', current flows

from BL to SL and vice versa to write '0' [12] [14].

### III. DNN WORKLOAD PROFILING

Profiling the target workload is a prerequisite for designing an accelerator for the target workload. Assuming that we have a powerful computing system to handle the exhaustive computations of the DL workload, we focus on providing efficient data movement between the compute and memory system to ensure 100% utilization of computing resources by introducing the workload-aware hybrid memory system. We propose the hybrid memory system by analyzing the Deep Learning model workloads from CV and NLP domain. We analytically model the on-chip bandwidth requirement and memory access patterns of different parts of the workload during inference and training, *Memory and Compute Model*, to develop the memory system for TPU-like [3] DNN accelerators.

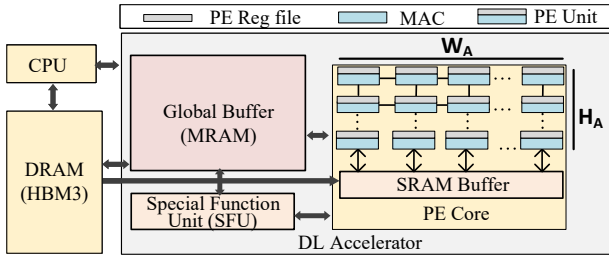


Fig. 5. Block diagram of Accelerator architecture

#### A. Memory Bandwidth Expression

We express the required bandwidth (BW) as a function of compute resources and workload.  $BW$  (bytes/sec) is defined as the rate at which data needs to be transferred to/from memory by a processor to utilize the computation resources of the processor fully. Mathematically,

$$BW = \frac{F_p}{OI} \quad (1)$$

Where  $F_p$  = Theoretical peak performance (ops/sec) = number of operations the accelerator performs per sec. The  $F_p$  of a  $H_A \times W_A$  Processing Element (PE) array (Fig. 5):

$$F_p = H_A * W_A * F_{acc} \quad (2)$$

$F_{acc}$  = Operating frequency of the accelerator.  $OI$  = Operational Intensity of Workload (ops/byte) = number of operations performed per byte accessed. It is a measure of parallelism of the workload. In the subsequent subsections, we will formulate the  $OI$  of Conv. and FC layer to find their BW, respectively. Note that the read and write bandwidth will not be the same for these workloads.

1) *Read Bandwidth ( $BW_{RD}$ ) of Conv. layer:* To formulate an expression for  $OI$  of convolution workload: First, we determine the total number of MAC operations,  $T_{MAC}$ , performed by a  $H_A \times W_A$  PE array per clock cycle

$$T_{MAC} = H_A * W_A \quad (3)$$

TABLE I  
CNN AND SYSTOLIC ARRAY PARAMETERS NOMENCLATURE

$W_A, H_A$	Accelerator array width & height (PEs)
$k_w, k_h$	Kernel width & height
$of_w, of_h$	Output feature map width & height
$if_w, if_h$	Input feature map width & height
$N_{ich}, N_{och}$	No. of input & output channel
$N_{bt}$	Batch size
$n_{fc}, m_{fc}$	No. of neurons in input & output FC layer

Second, we figure out how many bytes should be read from memory to utilize all PEs of the accelerator in one clock cycle. In a row stationary dataflow [20], it takes  $(k_h * k_w + of_h * of_w) * d_w$  bytes of data ( $d_w$  = data type in bytes, i.e., FP32, BF16 etc.) and  $\#(of_h * of_w * k_h * k_w)$  PEs to generate the partial ofmaps corresponding to one input channel. Depending on the size of the PE array, in each iteration (one complete use of accelerator), multiple input channels can be fit. The input channels (i.e., no. of partial ofmaps) computed by the PE array in each iteration:

$$N_{ich\_per\_stp} = \frac{H_A * W_A}{of_h * of_w * k_h * k_w} \quad (4)$$

Total bytes read from memory to utilize all PEs:

$$T_{byte} = \frac{H_A * W_A}{k_h * k_w * of_h * of_w} * (k_h * k_w + if_h * if_w) * d_w \quad (5)$$

We divide the total number of MAC operations,  $T_{MAC}$ , by the total bytes accessed,  $T_{byte}$ , to find  $OI$ :

$$OI = \frac{k_h * k_w * of_h * of_w}{d_w * (k_h * k_w + if_h * if_w)} \quad (6)$$

Substituting the expression of  $OI$  in equation (1) gives  $BW_{RD}$  as a function of array size and workload:

$$BW_{RD} = \frac{(k_h * k_w + if_h * if_w) * d_w}{k_w * k_h * of_h * of_w} * H_A * W_A * F_{acc} \quad (7)$$

For the symbol meanings, please see Fig. 2 and Table I.

2) *Write Bandwidth ( $BW_{WR}$ ) of Conv. Layer:* Partial ofmap of a single input channel requires  $\#(of_h * of_w * k_h * k_w)$  PEs. Therefore,  $H_A \times W_A$  PEs generate  $(H_A * W_A) / (of_h * of_w * k_h * k_w)$  ofmaps in each iteration. Each partial ofmap contains  $of_h * of_w$  elements. The total output bytes generated by the PE array in one iteration is, equivalently, the write bandwidth:

$$BW_{WR} = \frac{H_A * W_A * F_{acc} * d_w}{k_h * k_w} \quad (8)$$

3)  *$BW_{RD}$  &  $BW_{WR}$  of FC layer:* The systolic array is a widely used architecture to perform GEMM operation [3]. Depending on the array dimension ( $H_A \times W_A$ ) and operand matrix dimension (input matrix:  $K \times M$ , weight matrix:  $M \times N$ , and output matrix:  $K \times N$ ), we formulate required Read and Write GLB bandwidth for four different cases: (i) Weight matrix dimensions (both) are less than the systolic array dimensions ( $M < H_A, N < W_A$ ), (ii) Height of weight matrix is less than the height of systolic array, but the width of weight matrix is larger than or equal to the width of the systolic

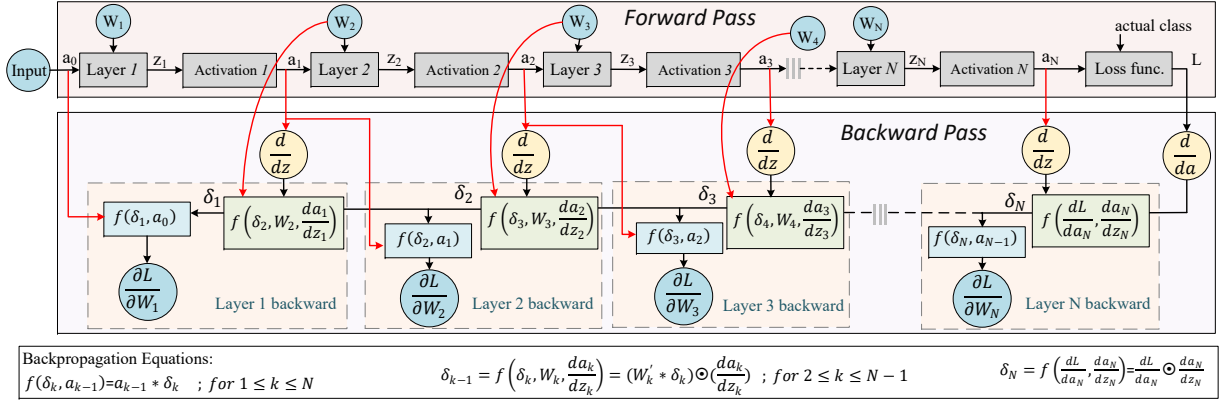


Fig. 6. Computational graph of DNN training

TABLE II  
RD/WR BANDWIDTH EXPRESSION OF FC LAYER FOR DIFFERENT CASES

Cases		$BW_{RD}$	$BW_{WR}$
$M < H_A; N < W_A$	$K < W_A$	$\frac{M*N+K*M}{N+K}$	$\frac{K*N}{2*N+K-1}$
	$K \geq W_A$	$\frac{M*N+W_A*M}{N+W_A}$	$\frac{W_A*N}{2*N+K-1}$
$M < H_A; N \geq W_A$	$K < W_A$	$\frac{M*W_A+K*M}{N+K}$	$\frac{K*W_A}{2*W_A+K-1}$
	$K \geq W_A$	$\frac{M*W_A+W_A*M}{2*W_A}$	$\frac{W_A^2}{2*W_A+K-1}$
$M \geq H_A; N < W_A$	$K < W_A$	$\frac{H_A*N+K*H_A}{N+K}$	$\frac{K*N}{2*N+K-1}$
	$K \geq W_A$	$\frac{H_A*N+W_A*H_A}{W_A+N}$	$\frac{W_A*N}{2*N+K-1}$
$M \geq H_A; N \geq W_A$	$K < W_A$	$\frac{H_A*W_A+W_A*H_A}{W_A+K}$	$\frac{W_A*N}{2*N+K-1}$
	$K \geq W_A$	$\frac{H_A*W_A+W_A*H_A}{2*W_A}$	$\frac{W_A^2}{2*W_A+K-1}$

TABLE III  
PARAMETER NOMENCLATURE FOR ALGORITHM 1 AND 2

$I, O, W$	<i>ifmap, ofmap, weight</i> size in MB
$RD_{DRAM}$	DRAM Read access counts
$WR_{DRAM}$	DRAM Write access counts
$RD_{GLB}$	GLB Read access counts
$WR_{GLB}$	GLB Write access counts
$GI, GO, GW$	<i>ifmap, ofmap, weight</i> Gradient size in MB
$mbpa$	MB of data fetched per memory access
$layer\_f$	Layer size (MB) combining <i>ifmap, ofmap</i> & <i>weights</i>
$layer\_b$	Layer size (MB) in backprop combining upstream, <i>ofmap</i> & <i>weight</i> gradient
$cum\_layer$	Cummulative size of layer
$rd\_f, rd\_b$	DRAM read access during forward & backward pass
$wr\_f, wr\_b$	DRAM write access during forward & backward pass

array ( $M < H_A, N \geq W_A$ ), (iii) Height of weight matrix is larger than or equal to the height of systolic array, but width of the weight matrix is less than the width of the systolic array ( $M \geq H_A, N < W_A$ ), and (iv) Both height and width of weight matrix are larger than or equal to the height and width of systolic array respectively ( $M \geq H_A, N \geq W_A$ ).

In a weight stationary dataflow, it takes  $N$  clock cycles to load the weight matrix into the systolic array. Once the weights are loaded, the input matrix is streamed from left to right and the outputs are collected downward. The input matrix's first

Algorithm 1: DRAM & GLB access count at Inference

```

1 for  $i = 1$  to no. of layers do
2    $RD_{GLB} \leftarrow \frac{I_i}{mbpa_{GLB}}$ 
3   if  $i = 1$  then
4      $WR_{GLB} \leftarrow \frac{I_i + O_i}{mbpa_{GLB}}$ 
5     if  $(I_i + W_i) \leq GLB$  then
6        $RD_{DRAM} \leftarrow \frac{I_i + W_i}{mbpa_{DRAM}}$ 
7     else
8        $RD_{DRAM} \leftarrow \frac{I_i + W_i}{mbpa_{DRAM}} + \frac{I_i + W_i - GLB}{mbpa_{DRAM}}$ 
9     end
10  else
11     $WR_{GLB} \leftarrow \frac{O_i}{mbpa_{GLB}}$ 
12    if  $O_{i-1} \leq UB$  then
13      if  $W_i \leq GLB$  then
14         $RD_{DRAM} \leftarrow \frac{W_i}{mbpa_{DRAM}}$ 
15      else
16         $RD_{DRAM} \leftarrow \frac{W_i}{mbpa_{DRAM}} + \frac{W_i - GLB}{mbpa_{DRAM}}$ 
17      end
18    else
19       $RD_{DRAM} \leftarrow \frac{I_i + W_i}{mbpa_{DRAM}} + \frac{(I_i + W_i) - GLB}{mbpa_{DRAM}}$ 
20    end
21  end
22  if  $i = \text{no. of layers}$  then
23     $WR_{DRAM} \leftarrow \frac{O_i}{mbpa_{DRAM}}$ 
24  else
25    if  $O_i > GLB$  then
26       $WR_{DRAM} \leftarrow \frac{O_i - UB}{mbpa_{DRAM}}$ 
27    else
28       $WR_{DRAM} \leftarrow 0$ 
29    end
30  end
31 end

```

column reaches the weight matrix's last column at  $2N$  clock cycles. The last (or  $K^{th}$ ) column of the input matrix reaches the last column of weight matrix after  $2N + K - 1$  clock cycles and generates the output matrix,  $K \times N$ . Based on the above dataflow and mapping, the peak read-write bandwidth per clock cycle for different cases is summarized in Table II. The expressions are shown for weight stationary dataflow.

From the transformer-based NLP model architecture (Fig. 3), we observe that the dominant operations are the GEMM operations. As a result, we model the read-write bandwidth requirement for different layers of the transformer-based model same as the read-write bandwidth of FC layer. Another



**Algorithm 2:** DRAM & GLB access count at Training

---

```

1  cum_layer ← 0;
2  tmp ← 0;
3  for i = 1 to no. of layers do
4      layer_fi ← Ii + Oi + Wi;
5      layer_bi ← GIi + GOi + GWi;
6      layer(i) ← layer_fi + layer_bi;
7      cum_layer(i) ← tmp + layer(i);
8      tmp ← cum_layer(i);
9      RDGLB ←  $\frac{3 \cdot I_i + O_i + 5 \cdot W_i}{mbpa_{GLB}}$ ;
10     WRGLB ←  $\frac{2 \cdot I_i + 2 \cdot O_i + 3 \cdot W_i}{mbpa_{GLB}}$ ;
11     if cum_layer(i) ≤ GLB then
12         if i = 1 then
13             rd_f(i) ←  $\frac{I_i + W_i}{mbpa_{DRAM}}$ ;
14         end
15         if i = no. of layers then
16             wr_f(i) ←  $\frac{O_i}{mbpa_{DRAM}}$ ;
17         end
18         rd_f(i) ←  $\frac{W_i}{mbpa_{DRAM}}$ ;
19         rd_b(i) ← 0;
20         wr_f(i) ← 0;
21     else
22         if (i ≠ 1) AND (Oi-1 ≤ GLB) then
23             rd_f(i) ←  $\frac{W_i}{mbpa_{DRAM}}$ ;
24         else
25             if Ii + Wi ≤ GLB then
26                 rd_f(i) ←  $\frac{I_i + W_i}{mbpa_{DRAM}}$ ;
27             else
28                 rd_f(i) ←  $\frac{I_i + W_i}{mbpa_{DRAM}} + \frac{I_i + W_i - GLB}{mbpa_{DRAM}}$ ;
29             end
30         end
31         if (GIi + GOi + GWi ≤ GLB) then
32             wr_f(i) ← 0;
33             rd_b(i) ← 0;
34         else
35             wr_f(i) ←  $\frac{GI_i + GO_i + GW_i}{mbpa_{DRAM}}$ ;
36             rd_b(i) ←  $\frac{GI_i + GO_i + GW_i}{mbpa_{DRAM}}$ ;
37         end
38     end
39     wr_b(i) ←  $\frac{W_i}{mb_{per\_acs}}$ ;
40 end

```

---

dominant operation after GEMM is softmax operation. Softmax operation is performed mostly on the scaled attention filter matrix, AF of size  $N_{sql} \times N_{sql}$ ;  $\sigma(AF)_{ij} = \frac{e^{AF_{ij}}}{\sum_{i=1}^{N_{sql}} e^{AF_{ij}}}$ . The softmax operation is generally performed in the Special Function Unit (SFU) [4] (Fig. 5). The bandwidth requirement of the softmax operation depends on the hardware architecture, mapping, and AF matrix dimension. Assuming that the SFU contains  $1 \times H_A$  units, each capable of performing one exponential operation, followed by an accumulator for accumulating the exponentials, and a regular ALU for performing the division the bandwidth of softmax operation on SFU is estimated as  $BW_{softmax} = d_w * H_A$ .

### B. Memory Access Patterns

Our proposed memory system consists of HMB3 (off-chip DRAM memory), a large GLB with multiple SOT-MRAM banks, a smaller double-buffered SRAM, and PE reg file specific to each PE unit (Fig. 5). The banks inside SOT-MRAM are optimized through a DTCO between the SOT-MRAM parameters and the workload requirements. The double-buffered

SRAM holds the weights and partial outputs. In this subsection, we analyze the memory access patterns of CV and NLP models for the proposed memory system.

The required number of main memory accesses depends on the GLB size, weight, activation size, and dataflow. Assuming a fixed dataflow, weight stationary in this case, we model the memory access counts during inference and training as a function of the model's workloads and the GLB size in Algorithm 1, and 2 for inference and training, respectively. During inference, inputs (e.g., images, tokens) are read from HBM3, written to GLB, and read from GLB to be operated inside PEs core. The read-only weights are directly loaded from HBM3 to the register file of each PE unit, bypassing the GLB. Using double-buffered SRAM, while the array is computing with loaded weights, the next set of weights is temporarily written to the SRAM buffer to hide the off-chip access latency behind the PE array computation latency. Suppose the GLB size is large enough to hold all samples in the minibatch. In that case, the data entity can be read all at once, resulting in the memory accesses equal to the algorithmic minimum memory accesses. Algorithmic minimum memory access represents the number of elements in the data entity [25]. For weight gradient calculation, during backpropagation of the training, the inputs are read from GLB to PE core, assuming that the GLB is large enough to hold the input images along with the generated ofmap of the current layer, thus avoiding the DRAM accesses during backward pass. In convolution, the inputs can be reused multiple times for convolutional and filter reuse. It can also be reused multiple times during backpropagation to calculate the gradients of different filters. In Transformer-based NLP models, the embedded input can be reused thrice as input to Key, Query, and Value linear layer. It can also be reused thrice during backpropagation to calculate the weight gradient of the Key, Query, and Value linear layer. The training workflow is complicated and requires many more memory accesses (both off-chip and on-chip) compared to inference. For example, to calculate the weight gradients of Layer 1, it requires the current layer's activation gradient  $\frac{da_1}{dz_1}$ , input ( $a_0$ ), next layer's weight ( $W_2$ ) and the upstream gradient from Layer 2 ( $\delta_1$ ) (Fig. 6).

The pseudo code of Alg. 1 models the inference memory access patterns. The inputs and weights must be loaded from DRAM for the first layer. Depending on the combined size of input & weight matrix size, and GLB size, it requires either algorithmic minimum read accesses or more than that (lines 3-9 of Alg. 1). For the rest of the layers, if the *ofmap* of the previous layer can fit in GLB, then no read accesses are required for input activation, as the *ofmap* of the previous layer will act as the *ifmap* to the next layer. Only the weights are read from DRAM for such layers (lines 12-20 of Alg. 1). The opposite case applies to write accesses: the *ofmap* of the last layer must be written to the DRAM. For other layers, it needs to be written to DRAM depending on its size and GLB size (lines 22-30 of Alg. 1). No write accesses are required for weight matrices during inference. As the weights bypass the GLB during inference, the GLB read accesses are calculated from the *ifmap* size for each layer (line 2). The write accesses are calculated from the *ofmap* except for the 1st layer (line 11, 4). See Table III for symbol meanings.

The pseudo code of Algorithm 2 models the training behavior. We initialize several temporary variables:  $layer\_f_i$  (comprising of  $ifmap$ ,  $ofmap$ , and  $weight$  matrices of  $i^{th}$  layer),  $layer\_b_i$  (comprising of upstream gradient,  $ofmap$ , and  $weight$  matrix gradients of  $i^{th}$  layer), and  $layer_i$  (combining  $layer\_f_i$  and  $layer\_b_i$ ) as shown in lines 4-6 in Alg. 2.  $cum\_layer(i)$  contains all layers' all entities up to  $i^{th}$  layer (line 7-8, Alg. 2). If the GLB is large enough to hold  $cum\_layer(i)$ , we just need to read the  $ifmap$  of the first layer &  $weight$  of all layers from DRAM during the forward pass and write all layers' updated  $weight$  during the backward pass and last layer's  $ofmap$  to DRAM during the forward pass (lines 11-20 & 39, Alg. 2). Otherwise, the forward pass is the same as the inference (lines 22-30, Alg. 2). During the backward pass, depending on the size of upstream gradients,  $ofmap$ , and  $weight$  gradients, it accesses the gradients from DRAM (lines 31-37, Alg. 2). The GLB read-write accesses are shown in lines 9-10 in Alg. 2. The  $ifmap$  of each layer needs to be read twice, once during the forward pass and once during the backward pass. The upstream gradient, equal in size as  $ifmap$ , must be read once during the backward pass. The  $ofmap$  is read once during the backward pass to calculate the upstream gradient. The weight is read 5 times (once during the forward pass, 4 times during the backward pass). The  $ifmap$  and  $ofmap$  are written twice, once during the forward pass and once during the backward pass. The  $weight$  is written thrice, twice during forward pass and once during backward pass.

#### IV. DTCO OF SOT-MRAM

To ensure overall system performance for AI workloads, the memory system should have large on-chip memory to avoid frequent DRAM accesses, and the on-chip memory should have high bandwidth to prevent the system from being memory-bound while being energy efficient. In this section, we perform a DTCO of SOT-MRAM in bit-cell level based on the workload profiling done in section III.

##### A. Optimizing critical switching current $I_c$

In SOT-MRAM, the magnetic orientation of the free layer is switched by Spin-Orbit Torque induced by spin Hall and interfacial effects between the channel (i.e., SOT layer) and free layer (FL) of MTJ. An in-plane charge current is flown through the channel to generate a spin current that exerts a spin torque on the free layer, which rotates the free layer's magnetic orientation. The critical current density required to switch the magnetic orientation of FL is expressed as [26]

$$j_c = \frac{2e\mu_0 M_{s,FL} t_{FL}}{\hbar \theta_{SH}} \left( \frac{H_{k,eff}}{2} - \frac{H_x}{\sqrt{2}} \right) \quad (9)$$

Where  $H_{k,eff}$  is the effective anisotropy field,  $H_x$  is the applied field,  $M_{s,FL}$  is the saturation magnetization of free layer, and  $t_{FL}$  is its thickness. Our interest is in lowering the switching current to achieve low write energy. Here, the free layer thickness  $t_{FL}$  and spin Hall efficiency  $\theta_{SH}$  act as a control knob for critical switching current.  $\theta_{SH}$  is a material-specific parameter and its higher value is expected to reduce the switching current. The typical value of  $\theta_{SH}$  in

heavy metal alloys ranges between 0.1 to 0.5 [24]. However, recent topological insulators as SOT layer can have a very large  $\theta_{SH}$ . [27] demonstrated  $\theta_{SH} = 152$  with *BiSb* thin films.

##### B. Optimizing read-write pulse width

1) *Read pulse width*: The reading of SOT-MRAM involves sensing the resistance of the MTJ. A small amount of current is passed through the MTJ stack and the voltage across the stack  $V_{data+}$  or  $V_{data-}$  is compared against a reference voltage  $V_{ref} = \frac{1}{2}(V_{data+} + V_{data-})$  to read out the stored bit. The read Sensing Margin  $SM = |V_{ref} - V_{data}|$  is typically very small. Sensing and amplifying this small difference requires a strong and complex Sense Amplifier that contributes to most of the read latency and energy. The SM is determined by the Tunnel Magneto Resistance ratio ( $TMR\ ratio = \frac{R_{AP}-R_P}{R_P}$ ) of MTJ. A higher TMR ratio produces a larger SM by making  $V_{data+}$  higher and  $V_{data-}$  lower. Thus the TMR ratio is inversely proportional to the read latency [28]. The higher the TMR window, the higher the read speed and the less effort required on the periphery. The typical range of the TMR ratio is between 100 to 300%. The TMR is tunable by oxide thickness [29] as shown in Fig. 15 (a). In SOT-MRAM, we can increase the oxide thickness, thanks to the decoupled read-write path of SOT-MRAM, to achieve a high TMR and increase the read speed without worrying about the large incubation time [30].

TABLE IV  
DTCO CONTROL PARAMETERS & THEIR IMPACT ON POWER, PERFORMANCE AND AREA (PPA)

DTCO Parameters	Impact on PPA
Spin Hall angle $\theta_{SH}$	$\theta_{SH} \uparrow$ , $j_c \downarrow$ , Switching energy $\downarrow$
Free layer thickness $t_{FL}$	$t_{FL} \downarrow$ , $j_c \downarrow$ , Switching energy $\downarrow$ , Area $\downarrow$
SOT layer dimension $A_{SOT}$	$A_{SOT} \downarrow$ , $\tau_p \downarrow$ , Area $\downarrow$ , Write Bandwidth $\uparrow$
Oxide thickness $t_{MgO}$	$t_{MgO} \uparrow$ , TMR $\uparrow$ , Read Bandwidth $\uparrow$

2) *Write pulse width  $\tau_p$* : The width of the write current pulse for switching is inversely proportional to the magnitude of the applied current density in the SOT layer  $j_{sw}$  [24]

$$\tau_p \propto \frac{1}{j_{sw}} \quad (10)$$

As the area of the SOT layer ( $A_{SOT}$ ) is scaled down, the effective current density increases,  $j_{sw} \propto 1/(A_{SOT})$ . Successful switching should take place when  $j_{sw} > j_c$ . We can increase  $j_{sw}$  by reducing the SOT layer dimension and decrease  $j_c$  by increasing  $\theta_{SH}$  or by decreasing  $t_{FL}$ . Thus we can achieve successful switching in much shorter pulse width (equation 10). [31] demonstrated the switching at 180ps, [32] at 400ps, and [33] at 210ps. Switching in shorter pulse width ensures larger write bandwidth which is essential for memory systems used in AI/Deep Learning hardware. The key DTCO parameters of SOT-MRAM and their impact on Power, Performance and Area (PPA) are listed in Table IV.

#### V. RESULTS AND ANALYSIS

In this section, we provide the results and analysis of the STCO on the CV and NLP workloads during inference

and training and present the optimum Power, Performance, and Area results by performing the DTCO of SOT-MRAM. We developed a MATLAB-based framework to implement our analytical *Memory and Compute Model* to capture the relationship between the memory access counts and the memory hierarchy sizes in typical systolic array based AI accelerators. Unlike ScaleSim [34] and Timeloop [25] simulator, which only support profiling DNN workloads in inference mode to date, our model captures both training and inference behavior of CV and NLP models. We also verified our model's results with Timeloop in inference mode.

### A. Bandwidth Demand

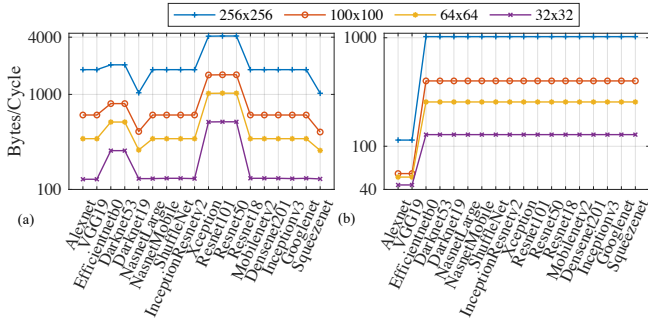


Fig. 7. Bandwidth requirement of CV models for different PE array size. (a) Read Bandwidth, (b) Write Bandwidth.

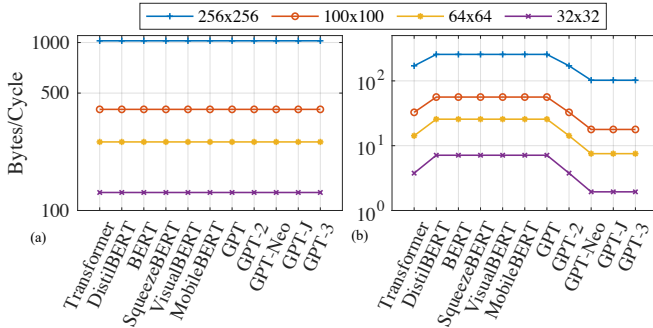


Fig. 8. Bandwidth requirement of NLP models for different PE array size. (a) Read Bandwidth (for GEMM and softmax operation), (b) Write Bandwidth.

In Fig. 7 (a), (b), we plot the read-write on-chip bandwidth demand in *bytes/cycle* of 18 widely used CV models. Resnet101 and Resnet50 running on a  $256 \times 256$  PE array will demand the highest read bandwidth, 4017 bytes/cycle, from GLB, whereas Squeezenet will demand the lowest bandwidth, 1028 bytes/cycle. Naturally, as the PE array size increases, the computation capacity per cycle  $T_{MAC}$  increases which demands more data from memory to keep all PEs active. From the workload perspective, we observe that the most contributing factor to the read bandwidth demand is its inverse relationship with the filter and ofmap size. We explain the inverse relationship of filter and ofmap size with the read bandwidth using the convolutional reuse concept. As the filter size decreases, the scope of convolutional reuse decreases. The ofmap again depends on the filter and ifmap size. With the

decrease of filter size and ofmap size, the convolutional reuse decreases, giving rise to more bandwidth demand. The layer of Resnet101 that requires the most bandwidth (4017 bytes/cycle) has the ofmap dimension ( $7 \times 7$ ) and filter dimension ( $1 \times 1$ ). On the other hand, the most demanding (1028 bytes/cycle) layer of Squeezenet has the ofmap dimension ( $18 \times 18$ ) and filter dimension ( $1 \times 1$ ). Another observation is that though  $1 \times 1$  convolution reduces the computation complexity, it requires more bandwidth from memory, i.e., becomes memory intensive. The write bandwidth is also inversely proportional to the filter size. However, in  $1 \times 1$  convolutions, it depends on the number of outputs generated by the PE array. The write bandwidth is always smaller than the read bandwidth (Fig. 7 (b)) as it takes more than one operands to generate one output. For example, in a  $3 \times 3$  convolution, it takes 18 operands to generate a single output; in a  $1 \times 1$  convolution, it takes two operands.

As mentioned in section III-A3, the bandwidth requirement for transformer-based model are calculated using the expressions of Table II. The dimension of the operand matrices is larger than the PE array dimension, hence following Case IV (Table II, Section III-A3), the read bandwidth of all models depends on the PE array size (Fig. 8 (a)). The write bandwidth depends on the PE array dimension and the input sequence length. The softmax read bandwidth depends on the SFU width, and matches with the GEMM read bandwidth. As different models are trained with different input sequence lengths [35], their write bandwidth demand is not the same across all models. The parameter sizes and settings of the models used in this work are shown in Table V. The models having the highest sequence length (2048) have the lower write bandwidth demand 102 bytes/cycle running on a  $256 \times 256$  PE array (Fig. 8 (b)).

### B. Impact of on-chip memory

Compared to a GLB size of 2MB, the DRAM access counts for all CV models decrease significantly if we increase the GLB size. In inference, reaching the 100% reduction in access means it only needs to read the initial inputs, weights for each layer, and write the final layer output, no DRAM access is needed for the intra and inter-layer operations. Further increase in GLB size will not improve the performance in these cases. For 16 samples, DRAM access is reduced by 100% for 14 models at 128MB, and most models experience a reduction of  $>80\%$  at 64MB (Fig. 9 (a)). Fig. 9 (b), (c) show the performance speed up and energy saving coming from these DRAM access reductions.

We observe a slower improvement in the DRAM access reduction during training unless the GLB size is large enough, at least 256MB for most models (Fig. 9 (d)). However, even the smaller percent reduction in DRAM access results in significant performance and energy improvement (Fig. 9 (e), (f)). This is because training requires at least  $2 \times$  DRAM accesses as inference. The smaller percent reduction of a large number of DRAM accesses translates to a significant energy and latency improvement. A similar trend is observed for NLP models. Transformer-based NLP models are usually larger than the CV models. This is the reason we achieve more performance speedup and energy reduction even at smaller DRAM access



TABLE V  
PARAMETERS OF NLP MODELS

Model	Enc. layer	Dec. layer	Attn. head	Word Embedding ( $N_{em}$ )	Intermediate dimension ( $d_{ff}$ )	Seq. length ( $N_{sql}$ )	Vocab. size ( $N_{vocab}$ )
Transformer	12	6	8	512	2048	1024	37000
BERT	12	-	12	768	3072	512	30522
Distil BERT	6	-	12	768	3072	512	30522
Mobile BERT	24	-	4	128	512	512	30522
Squeeze BERT	12	-	12	768	3072	512	30522
Visual BERT	12	-	12	512	3072	512	30522
GPT	-	12	12	768	2048	512	40478
GPT-2	-	12	12	768	2048	1024	50257
GPT-3	-	96	96	12288	49152	2048	50257
GPT-Neo	-	24	16	2048	8192	2048	50257
GPT-J	-	28	16	4096	16384	2048	50400

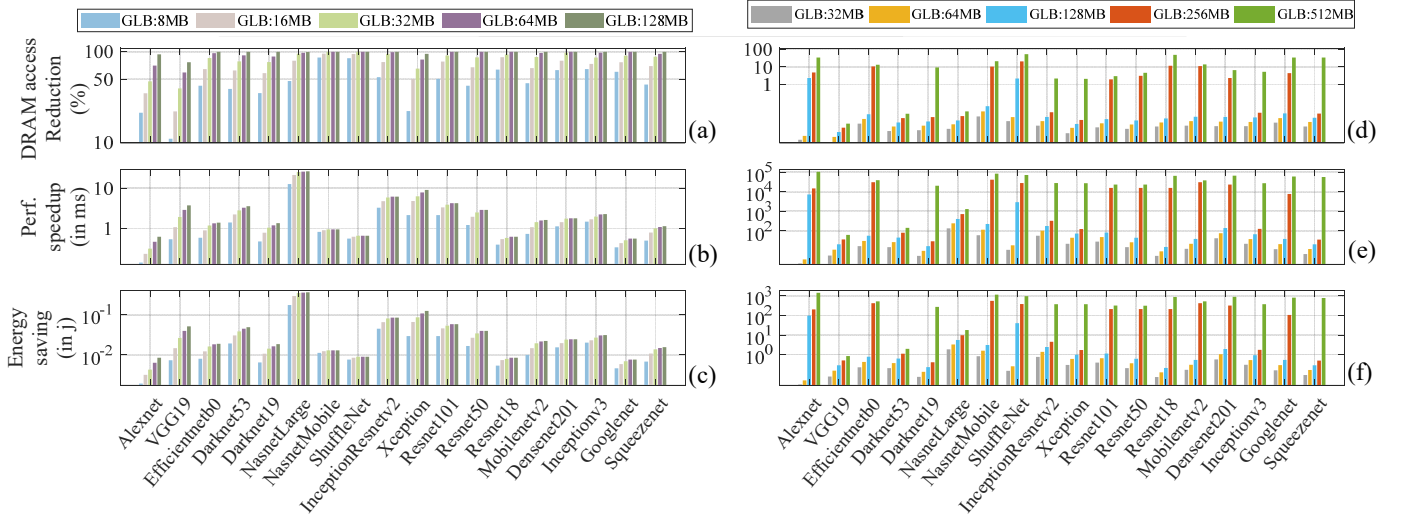


Fig. 9. Impact of larger GLB memories on performance and energy efficiency for CV models at inference and training. Percentage reduction in DRAM accesses at inference (a) and training (d). Performance Speedup from DRAM access reductions at inference (b) and training (e). Energy savings from reduced DRAM accesses at inference (c) and training (f). Both cases compare results to a baseline of 2MB GLB running 16 samples.

reduction rate (Fig. 11). We also observe that DNN models learn faster if we increase the batch size. However, for a fixed GLB size, the DRAM access count increases significantly at the larger batch size, causing performance slowdown and more energy consumption. Fig. 10 and Fig. 12 (a, b, c for inference and d, e, f for training) show the increase in DRAM access count and its associated impact on performance and energy for CV and NLP models respectively at different batch sizes. The key takeaway from this analysis is that we can reduce the energy and latency associated with DRAM accesses if we increase the GLB size. For larger batch sizes, the energy and latency improvement is even more. Because at large batch sizes, throughput increases at the cost of DRAM accesses. As we increase the GLB size, DRAM accesses reduce, and we achieve latency and energy reduction.

### C. DTCO of SOT for PPA Optimization

From section V-B we see that the GLB size of 64MB (for inference) and 256MB (for training) offer significant energy and performance improvement. However, it is not feasible and efficient to use such large SRAMs because of its area and leakage power, even if the low-power techniques are employed. Section V-A implies that we need approximately

4000bytes/cycle bandwidth between GLB and PE array for larger array size (256x256). In this subsection, we provide the SOT-MRAM DTCO results and observation meeting the requirements stated in the above two subsections. We perform the DTCO in Cadence Virtuoso tool using the compact SOT-MRAM model from [15], and use Synopsys 14nm library [37] for the CMOS transistors and peripheral circuits.

**1c optimization:** To realize the impact of SOT efficiency,  $\theta_{SH}$  on  $I_c$ , we sweep  $\theta_{SH}$  from 0.1 to 100 (Fig. 13 (a)). With  $\theta_{SH}$  100,  $I_c$  goes as low as 0.5uA. Even though the widely used SOT layers are made of heavy metal alloys having smaller  $\theta_{SH}$  (e.g., 0.1 to 0.5), recent advancement in material engineering demonstrates that in topological insulator  $\theta_{SH}$  can go as high as 152 [27]. We recommend using topological insulators as the SOT layer to achieve a lower switching current.

Next, we analyze the impact of SOT layer geometry on the switching current (Fig. 13 (b), (c)).  $I_c$  scales down linearly with the decrease of SOT layer width, and  $w_{SOT}$  can be set to desired value based on the performance and reliability requirement (Fig. 13 (b)). While  $I_c$  scales linearly with the width of the SOT layer, the thickness of the SOT layer has an interesting effect on the switching current. The SOT layer should be relatively thin but bulk enough for heavy metal layers

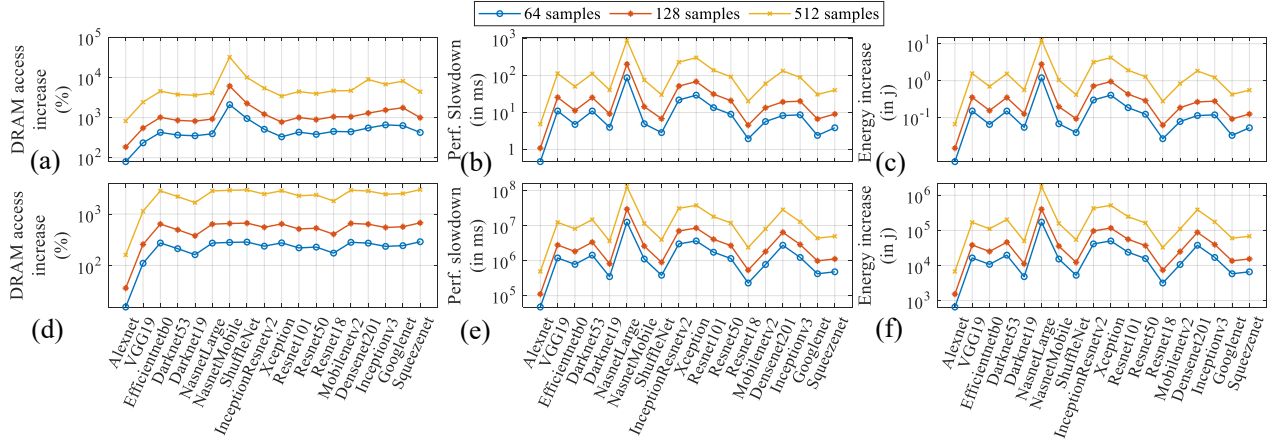


Fig. 10. Impact of batch size on performance and energy efficiency for CV models at inference and training. Percentage increase in DRAM accesses at inference (a), at training (d). Performance slowdown (latency increase) from extra DRAM accesses at inference (b), at training (e). Energy increase from extra DRAM accesses at inference (c), at training (f). In both cases, results are compared to a baseline of 16 samples running with 4MB GLB.

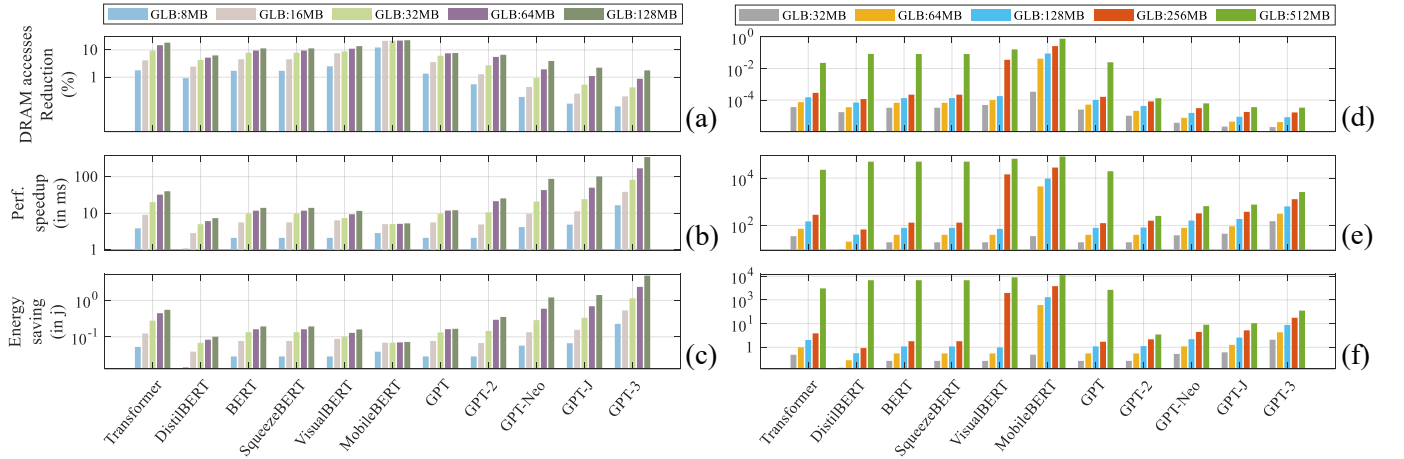


Fig. 11. Impact of larger GLB memories on performance and energy efficiency for NLP models at inference and training. Percentage reduction in DRAM accesses at inference (a), at training (d). Performance Speedup from DRAM access reductions at inference (b), at training (e). Energy savings from reduced DRAM accesses at inference (c), at training (f). In both cases, results are compared to a baseline of 2MB GLB running 16 samples

to experience the bulk effect to achieve high SOT efficiency. Once it crosses optimum thickness, which is 3nm (Fig. 13 (c)), many of the charges that are injected into the metal do not contribute to the switching, and  $I_c$  increases.

The smaller the free layer thickness,  $t_{FL}$ , the smaller the switching current (Fig. 13 (d)). We also scale the diameter of MTJ,  $d_{MTJ}$ , to reduce the MTJ area. However, with the scaling down of  $d_{MTJ}$  together with  $t_{FL}$ , the thermal stability factor  $\Delta$  also scales down, reducing the memory's data retention time  $t_{ret}$ . Non-volatility is a great feature of MRAM, but it can be compromised to achieve higher density, higher bandwidth, and lower energy when the target application is a cache. Because, in the cache even for AI workloads, the data lifetime is much shorter, typically in the seconds range [38]. Fig. 14(b) shows  $\Delta$  and  $t_{ret}$  as functions of free layer volume. While scaling down  $t_{FL}$  to optimize  $I_c$ , and  $d_{MTJ}$  to optimize area, we keep an eye on the reliability of the stored data. We consider a retention failure rate of  $10^{-9}$  (i.e., 1 bit flip per billion).

2) *Bandwidth optimization:* As shown in Fig. 15 (a), TMR ratio of the MTJ device can be increased by increasing the

oxide thickness [29]. We increase the oxide thickness to decrease the read latency (Fig. 15 (b)). The write pulse width is inversely proportional to the applied switching current. While we want to lower the applied current to achieve low energy, the higher amplitude of the applied current is required for faster magnetization reversal. However, switching occurs at smaller pulse width at the iso-current if we scale down the SOT layer width. This is because of the smaller critical current at smaller geometry (Fig. 13 (b,d)). Fig. 14(a) shows that switching pulse width can be reduced significantly by scaling down the SOT layer width. Thus, we can achieve higher write bandwidth by scaling down the SOT layer width to meet the high BW demand from AI workloads.

#### D. Process & Temperature Variation and Bitcell Simulation

In this subsection, we perform Process and Temperature variation on the DTCO-optimized parameters, design the peripheral circuits, and test the read-write operation on the bit cell at scaled parameters.

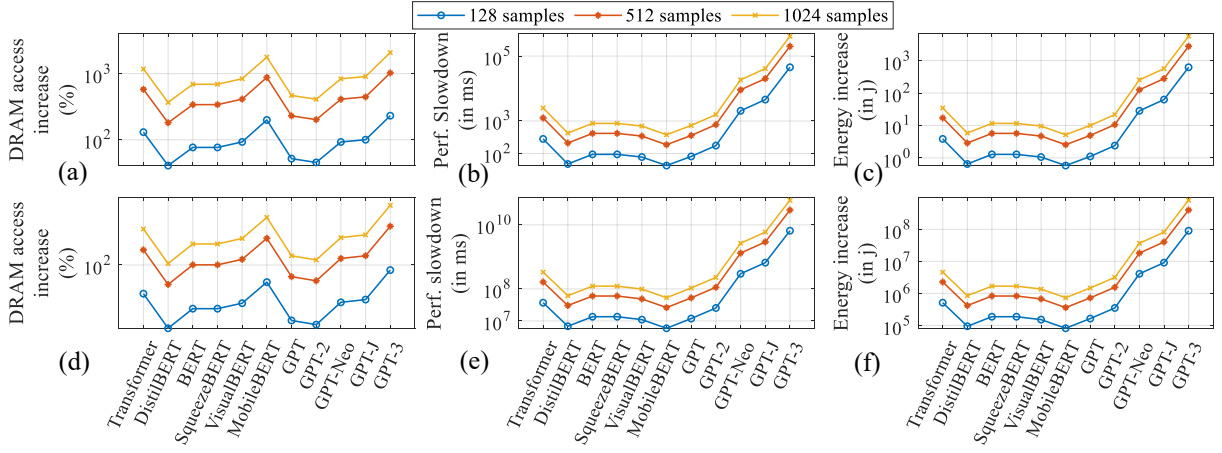


Fig. 12. Impact of batch size on performance and energy efficiency for NLP models at inference and training. Percentage increase in DRAM accesses, inference (a), and training (d). Performance slowdown (latency increase) from extra DRAM accesses at inference (b), at training (e). Energy increase from extra DRAM accesses at inference (c), at training (f). Results are compared to a baseline of 16 samples running with 4MB GLB.

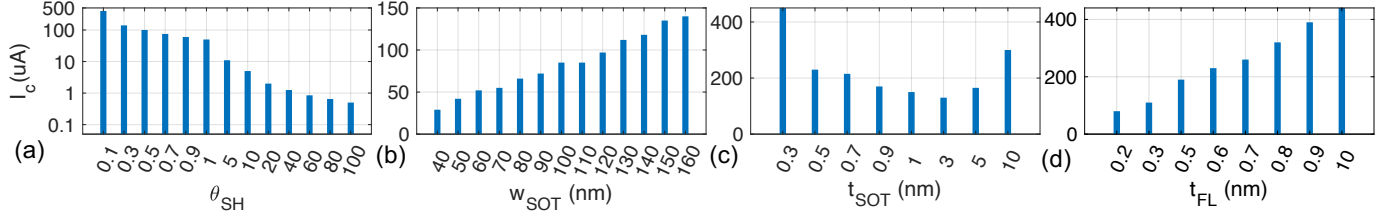


Fig. 13. Critical current vs  $\theta_{SH}$ (a),  $w_{SOT}$ (b),  $t_{SOT}$ (c), and  $t_{FL}$ (d).

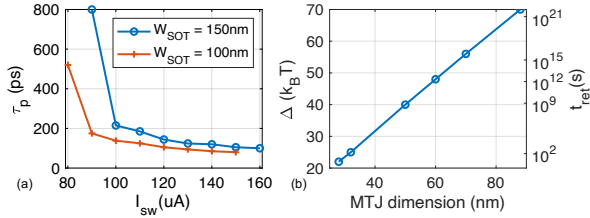


Fig. 14. (a) Switching pulse width  $\tau_p$  vs applied switching current  $I_{sw}$ . (b) Thermal stability factor  $\Delta$  (left Y-axis) and retention time  $t_{ret}$  (right Y-axis) vs MTJ dimension for a fixed retention failure rate,  $P_{RF} = 10^{-9}$ . At  $\Delta = 70$ , MTJ dimension = 88nm, retention time is  $> 10$  years [36].

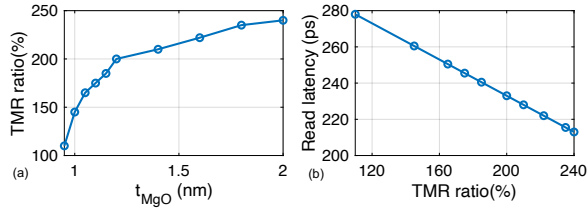


Fig. 15. Impact of, (a) oxide thickness on TMR. (b) TMR on read latency.

1) *Process and Temperature variation:* To incorporate process variations, we model MTJ diameter, free layer thickness, and SOT layer width as Gaussian variables in the Verilog A model of SOT-MTJ [15]. We assume standard deviations ( $\sigma$ ) as 5% of their respective means ( $\mu$ ) and perform Monte Carlo simulations with 5000 samples within  $4\sigma$  variation. We also consider the temperature variations. The extreme point at the

TABLE VI  
SOT-MRAM DTCO OPTIMIZED PARAMETERS. 30% GUARD-BAND ARE ADDED WITH THICKNESS AND WIDTH FOR PROCESS VARIATIONS.

Parameter	Value	Parameter	Value
Spin Hall angle $\theta_{SH}$	1	TMR	240%
Free layer thickness $t_{FL}$	0.5nm	MTJ diameter $d_{MTJ}$	55nm
SOT width $w_{SOT}$	130nm	SOT thickness $t_{SOT}$	3nm
Oxide thickness $t_{MgO}$	3nm	Thermal stability factor $\Delta$	45

right side of the scaled target parameter is  $\mu + 4\sigma, T_{cold}$  (Fig. 16). From equations 9 and 10,  $I_{sw}$  and  $\tau_p$  are independent of Temperature. As a result, the worst case for write operation (highest  $I_{sw}$  and longest  $\tau_p$ ) is at  $\mu + 4\sigma$ . This point is, however, benign to the read operation and retention failure. As we scale down  $d_{MTJ}$  and  $t_{FL}$ ,  $\Delta$  also reduces, reducing  $t_{ret}$ , and  $I_{data}$ .  $\Delta$  reduces further as temperature increases [11]. Thus, the worst case for read operation (smallest  $I_{data}$ ) and retention failure (smallest  $t_{ret}$ ) is at  $\mu - 4\sigma, T_{hot}$  (see Fig. 16). As  $I_{data}$  reduces, the difference between  $I_{data1}$  and  $I_{data0}$  becomes even smaller and difficult to sense.

To ensure the reliability of the SOT-MRAM bit cell, we add a 30% guard band on the scaled SOT device parameters: 20% for process variation and 10% for temperature variation. The optimized DTCO parameters after adding the PT induced 30% guard-band are shown in Table VI.

2) *Write operation:* To write SOT-MRAM bitcell, we bias BL with the data-to-be-written and SL with the complement of data-to-be-written. Assuming that the magnetization state of the Reference layer is -1, to write 1 into the MTJ bitcell, we

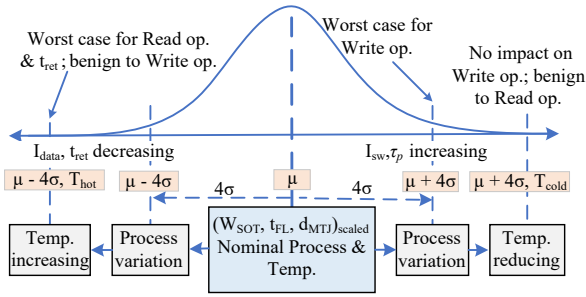


Fig. 16. Impact and distribution of Process and Temperature variation on scaled parameters.

switch the magnetic orientation of the Free layer to +1 state resulting in a high resistive state. To achieve this state, we turn on the WWL, connect BL to VDD and SL to the ground. The resultant current switches the free layer's magnetic orientation from -1 to +1. The opposite bias is applied to write 0. We do not need any additional peripheral circuits for the write operation of SOT-MTJ.

3) *Read operation:* Read operation involves sensing the current passing through MTJ at P and AP states. For our SOT-MRAM bitcell, with the parameters shown in Table VI,  $I_{data0} = 20\mu A$  and  $I_{data1} = 33\mu A$ . We design and optimize the read circuitry to sense this small differential current, as shown in Fig. 17. Our proposed read sensing circuit only contains an additional current mirror block (to amplify current), and it does not require the precharge circuits compared to SRAM. Hence, there is no additional area overhead in the periphery compared to SRAM. The dynamic power consumption are shown in Table VII

To capture the stochastic nature of MTJ switching, we simulate the bit cell for 1000 bitstream. We achieve a read and write yield of 100%, and at 250ps and 520ps, respectively. This results in read bandwidth of 4 Gbps and a write bandwidth of 1.9 Gbps. We then dynamically allocate the memory bus width on-demand to satisfy the bandwidth requirement for different workloads and PE array size stated in section V-A.

TABLE VII  
DYNAMIC POWER CONSUMPTION (IN uW) OF SRAM AND SOT-MRAM.  
(1/0) MEANS THE CORRESPONDING POWER TO ACCESS BIT 1 AND 0.

	Read(1/0)	Write(1/0)
SRAM	426	373
SOT-MRAM	150/368	325/300

#### E. System level performance evaluation of SOT-MRAM based Memory

In this subsection, we analyze the PPA (Power, Performance, and Area) metrics at the system level on the DNN/CNN benchmarks with SRAM, SOT-MRAM, and DTCO-optimized-SOT-MRAM. We use the Destiny [39] memory simulator to find the array-level data for both SRAM and SOT-MRAM. We modify Destiny source code to reflect: (i) SOT switching mechanism, (ii) special read sensing circuit for SOT-MRAM, and (iii) 14nm CMOS technode. Then, we feed the extracted

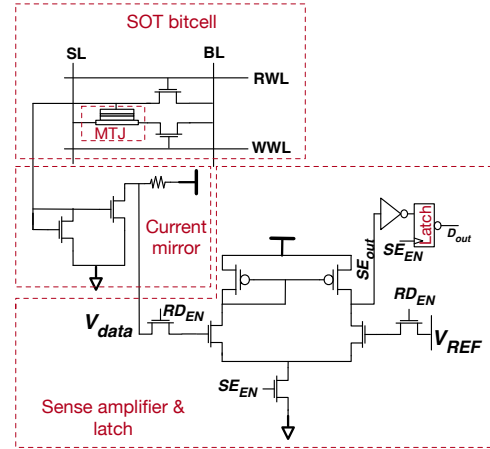


Fig. 17. SOT-MTJ bitcell with read sensing circuitry.

bitcell-level data of SOT-MRAM in the *.cell* file to find the PPA at the desired memory capacity.

Based on the array-level results from Destiny, and DRAM & GLB access counts from Algorithms 1, and 2, we estimate the system-level power and performance. Finally, we analyze the area of the memory modules of different technologies (14nm SRAM, SOT-MRAM, and DTCO-opt-SOT-MRAM) at iso-capacity. This analysis only incorporates the PPA metrics from the memory system (DRAM and GLB), assuming that the PPA of the compute unit is constant. With SOT-MRAM as GLB, we see significant energy and latency improvement over SRAM at 64MB (for inference) and 256MB (for training) (see Fig. 18 (a-d) for DNN benchmarks and (e-h) for NLP benchmarks). On average, the 64MB SOT-MRAM offers  $5\times$  energy reduction and  $2\times$  latency reduction over 64MB SRAM across all CNN models at inference. Our DTCO-optimized-SOT-MRAM offers further improvement,  $7\times$  energy, and  $8\times$  latency reduction over SRAM at iso-capacity. For latency improvement, the most contributing factor is the DRAM access reduction with large GLB and the smaller read/write latency of SOT-MRAM at larger capacity compared to SRAM. At smaller capacity, SRAM is way faster than SOT-MRAM [10], [14]. We observe that the most contributing factor in energy reduction ( $>50\%$ ) is the near-zero leakage power of SOT-MRAM compared to high leakage power of SRAM. The improvement is even more in training mode;  $6\times$  ( $8\times$  with SOT-opt.) energy reduction and  $2\times$  ( $9\times$  with SOT-opt.) latency reduction. With 64MB SOT-MRAM, NLP models in inference mode experience  $2\times$  ( $3\times$  with SOT-opt.) energy reduction and  $2\times$  ( $4\times$  with SOT-opt.) latency reduction than 64MB SRAM. Like CV benchmarks, with 256MB SOT-MRAM, NLP benchmarks also experience more energy improvement,  $6\times$  ( $8\times$  with SOT-opt.), and latency improvement,  $2.5\times$  ( $4.5\times$  with SOT-opt.), in training mode. The more improvement in training mode is because of two reasons: (i) GLB size increases from 64MB to 256MB, and (ii) GLB access counts are significantly large (at least  $5\times$ ) in training. Our DTCO-opt-SOT-MRAM further adds value to PPA by its smaller silicon area,  $0.54\times$  at 64MB and  $0.52\times$  at 256MB of 14nm SRAM at iso-capacity (Fig. 19).



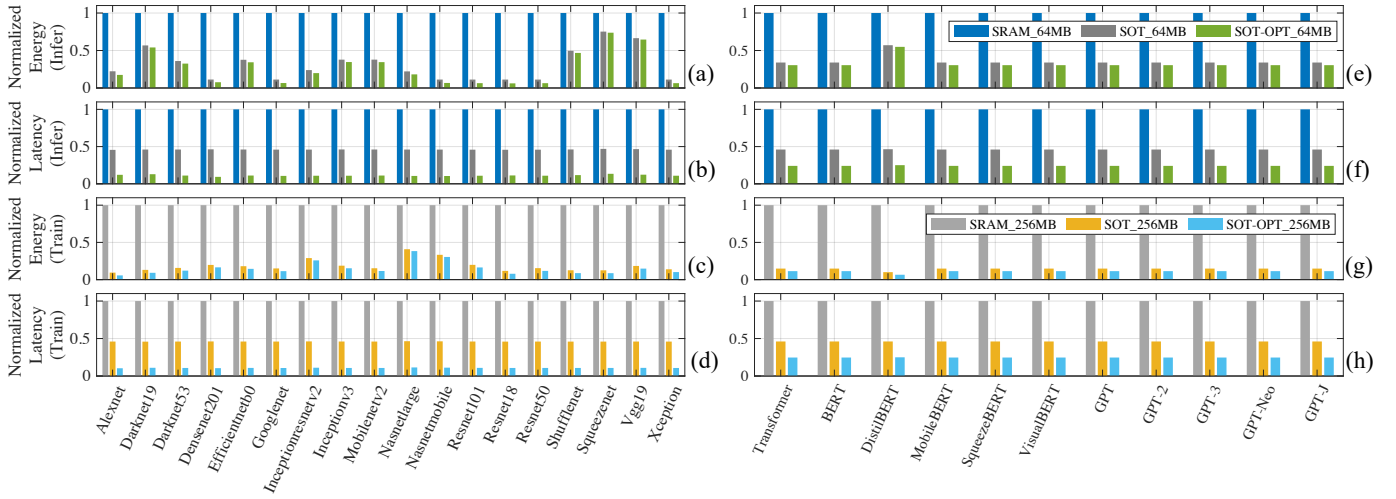


Fig. 18. System level energy improvement with SOT-MRAM and DTCO-optimized-SOT-MRAM over SRAM at the same size for CV (a-d) and NLP (e-h) models. The top plots show energy (a, e) and latency (b, f) for inference, and the bottom plots show energy (c, g) and latency (d, h) for training.

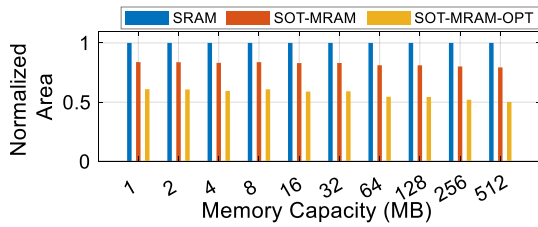


Fig. 19. Area improvement of SOT-MRAM and SOT-MRAM-OPT

## VI. RELATED WORK

SOT-MRAMs have been widely studied as the next generation of STT-MRAM to leverage all benefits of MRAMs as embedded memory [9] [10] [12] [13] [16] [17]. However, very few studies have evaluated the performance of SOT-MRAM as on-chip memory in system-level for AI accelerators. [14] and [40] demonstrated the performance improvement of SOT-MRAM as L2 data cache compared to SRAM L2 cache on MiBench, SPEC2000 and SPEC2006 benchmarks. SOT-MRAMs have also been explored in the context of DL accelerators as a promising technology for In-Memory Computing (IMC) or Computing-In Memory (CIM) [41] [42] [43] [44] [45]. IMC/CIM over conventional AI accelerator has pros and cons, and the detailed comparison between these two domains is outside the scope of this work. Our work, where we use SOT-MRAM as the cache storage element, differs from crossbar-based in-memory computing. While the scope of SOT-MRAM has been explored both as regular CPU cache and IMC for DL accelerator to some extent, to the best of our knowledge, unlike IMC, this is the first work that presents a comprehensive analysis of SOT-MRAM as on-chip memory for application in AI/DL accelerators.

## VII. CONCLUSION

In this research, we presented a System and Design Technology Co-optimization methodology for efficient and high-performance memory system design with SOT-MRAM for

modern AI accelerators. Guided by detailed target workload characterization, our memory system comprises of HBM3 DRAM, a DTCO-enabled SOT-MRAM GLB and a small SRAM buffer. Our large SOT-MRAM GLB significantly reduces the energy and latency by reducing expensive DRAM accesses while still having acceptable on-chip access energy and latency, achieving overall system-level high performance. We finally demonstrate that our memory system performs  $8\times$  and  $9\times$  better in terms of energy and latency respectively on CV benchmarks in training (7 and 8 times better in inference) and  $8\times$  and  $4.5\times$  better in terms of energy and latency respectively on NLP benchmarks in training (3 and 4 times better in inference) while consuming only around 50% of SRAM area at iso-capacity.

## REFERENCES

- [1] J. Hestness, N. Ardalani, and G. Diamos, "Beyond human-level accuracy: Computational challenges in deep learning," in *24th Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '19, 2019.
- [2] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.
- [3] N. Jouppi et al., "A Domain-Specific Architecture for Deep Neural Networks," ACM Communications, 2018.
- [4] "NVIDIA Ampere100 GPU." [Online]. Available: <https://www.nvidia.com/en-us/data-center/ampere-architecture/>
- [5] Q. Cao et al., "Are mobile dnn accelerators accelerating dnn?" in *International Workshop on Embedded and Mobile Deep Learning*, 2021.
- [6] J. Park et al., "Deep learning inference in facebook data centers: Characterization, performance optimizations and hardware implications," *arXiv preprint arXiv:1811.09886*, 2018.
- [7] H.-S. P. Wong et al., "Stanford Memory Trends," 2020. [Online]. Available: <https://nano.stanford.edu/stanford-memory-trends>
- [8] H. Li, M. Bhargava, P. N. Whatmough, and H. Philip Wong, "On-Chip Memory Technology Design Space Explorations for Mobile Deep Neural Network Accelerators," in *Design Automation Conference (DAC)*, 2019.
- [9] T. Endoh, H. Honjo, K. Nishioka, and S. Ikeda, "Recent progresses in stt-mram and sot-mram for next generation mram," in *2020 IEEE Symposium on VLSI Technology*, 2020, pp. 1–2.
- [10] M. Gupta et al., "High-density sot-mram technology and design specifications for the embedded domain at 5nm node," in *2020 IEEE International Electron Devices Meeting*, 2020, pp. 24.5.1–24.5.4.
- [11] A.V. Khvalkovskiy et al., "Basic principles of STT-MRAM cell operation in memory arrays," in *Journal of Physics D: Applied Physics*, 2013.



- [12] M. Natsui et al., "Dual-port field-free sot-mram achieving 90-mhz read and 60-mhz write operations under 55-nm cmos technology and 1.2-v supply voltage," in *IEEE Symposium on VLSI Circuits*, 2020.
- [13] K. Garelo, F. Yasin, and G. S. Kar, "Spin-orbit torque mram for ultrafast embedded memories: from fundamentals to large scale technology integration," in *IEEE International Memory Workshop (IMW)*, 2019.
- [14] F. Oboril, R. Bishnoi, M. Ebrahimi, and M. B. Tahoori, "Evaluation of hybrid memory technologies using sot-mram for on-chip cache hierarchy," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 3, pp. 367–380, 2015.
- [15] M. Kazemi et al., "Compact model for spin-orbit magnetic tunnel junctions," *IEEE Transactions on Electron Devices*, 2016.
- [16] S.Z. Rahaman et al., "Size-dependent switching properties of spin-orbit torque mram with manufacturing-friendly 8-inch wafer-level uniformity," *IEEE Journal of the Electron Devices Society*, vol. 8, pp. 163–169, 2020.
- [17] H. Honjo et al., "First demonstration of field-free sot-mram with 0.35 ns write speed and 70 thermal stability under 400°C thermal tolerance by canted sot structure and its advanced patterning/sot channel technology," in *2019 IEEE International Electron Devices Meeting (IEDM)*, 2019.
- [18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [19] A. Vaswani et al., "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
- [20] Y. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," in *IEEE JSSC*, 2017.
- [21] N. Verma, H. Jia, H. Valavi, Y. Tang, M. Ozatay, L.-Y. Chen, B. Zhang, and P. Deaville, "In-memory computing: Advances and prospects," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 3, pp. 43–55, 2019.
- [22] "NVIDIA." [Online]. Available: <http://nvidia.org>
- [23] Y. Seo, K.-W. Kwon, X. Fong, and K. Roy, "High performance and energy-efficient on-chip cache using dual port (1r/1w) spin-orbit torque mram," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 3, pp. 293–304, 2016.
- [24] A. Manchonet et al., "Current-induced spin-orbit torques in ferromagnetic and antiferromagnetic systems," *Reviews of Modern Physics*, vol. 91, no. 3, p. 035004, 2019.
- [25] A. Parashar et al., "Timeloop: A systematic approach to dnn accelerator evaluation," in *2019 IEEE international symposium on performance analysis of systems and software*. IEEE, 2019, pp. 304–315.
- [26] K.-S. Lee, S.-W. Lee, B.-C. Min, and K.-J. Lee, "Threshold current for switching of a perpendicular magnetic layer induced by spin hall effect," *Applied Physics Letters*, vol. 102, no. 11, p. 112410, 2013.
- [27] N. H. D. Khang, Y. Ueda, and P. N. Hai, "A conductive topological insulator with large spin hall effect for ultralow power spin-orbit torque switching," *Nature materials*, vol. 17, no. 9, pp. 808–813, 2018.
- [28] B. Wu et al., "Field-free 3t2sot mram for non-volatile cache memories," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, pp. 4660–4669, 2020.
- [29] K. Tsunekawa et al., "Giant tunneling magnetoresistance effect in low-resistance cofeb/mgo (001)/cofeb magnetic tunnel junctions for read-head applications," *Applied Physics Letters*, vol. 87, no. 7, 2005.
- [30] K. Wang, J. Alzate, and P. K. Amiri, "Low-power non-volatile spintronic memory: Stt-ram and beyond," *Journal of Physics D: Applied Physics*, vol. 46, no. 7, p. 074003, 2013.
- [31] Garelo, Kevin et al., "Ultrafast magnetization switching by spin-orbit torques," *Applied Physics Letters*, vol. 105, no. 21, 2014.
- [32] Wu, YC et al., "Voltage-gate-assisted spin-orbit-torque magnetic random-access memory for high-density and low-power embedded applications," *Physical Review Applied*, vol. 15, no. 6, p. 064015, 2021.
- [33] Garelo, Kevin et al., "Sot-mram 300nm integration for low power and ultrafast embedded memories," in *2018 IEEE Symposium on VLSI Circuits*. IEEE, 2018, pp. 81–82.
- [34] A. Samajdar et al., "A systematic methodology for characterizing scalability of dnn accelerators using scale-sim," in *IEEE International Symposium on Performance Analysis of Systems and Software*, 2020.
- [35] "Hugging Face." [Online]. Available: <https://huggingface.co/>
- [36] Garelo, K. et al., "Manufacturable 300nm platform solution for field-free switching sot-mram," in *Symposium on VLSI Technology*, 2019.
- [37] "Synopsys." [Online]. Available: <https://www.synopsys.com/>
- [38] K. Mishty and M. Sadi, "Designing efficient and high-performance ai accelerators with customized stt-mram," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 29, pp. 1730–1742, 2021.
- [39] M. Poremba et al., "Destiny: A tool for modeling emerging 3d nvm and edram caches," in *Design, Automation Test in Europe*, 2015.
- [40] Y. Seo, K.-W. Kwon, X. Fong, and K. Roy, "High performance and energy-efficient on-chip cache using dual port (1r/1w) spin-orbit torque mram," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 3, pp. 293–304, 2016.
- [41] L. Chang, X. Ma, Z. Wang, Y. Zhang, Y. Xie, and W. Zhao, "Pxnor-bnn: In/with spin-orbit torque mram preset-xnor operation-based binary neural networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 11, pp. 2668–2679, 2019.
- [42] S. Angizi, Z. He, A. S. Rakin, and D. Fan, "Cmp-pim: An energy-efficient comparator-based processing-in-memory neural network accelerator," in *55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2018.
- [43] H. Wang et al., "A new mram-based process in-memory accelerator for efficient neural network training with floating point precision," in *IEEE International Symposium on Circuits and Systems*, 2020.
- [44] G. Yuan, X. Ma, S. Lin, Z. Li, and C. Ding, "A sot-mram-based processing-in-memory engine for highly compressed dnn implementation," *arXiv preprint arXiv:1912.05416*, 2019.
- [45] Y. Luo et al., "Performance benchmarking of spin-orbit torque magnetic ram for deep neural network (dnn) accelerators," in *2022 IEEE International Memory Workshop (IMW)*, 2022, pp. 1–4.



**Kaniz Mishty** received the B.S. degree in Electronics and Communication Engineering from Khulna University of Engineering and Technology, Bangladesh, in 2018. She is currently working towards her Ph.D. degree in ECE at Auburn University, AL, USA. Her research interests are energy efficient AI hardware design and AI/ML in CAD. She interned with Apple Inc. in Summer '22, 23 and Qualcomm in '21 on AI application in SoC and custom circuit design.



**Mehdi Sadi** (S'12-M'17) is an Assistant Professor at the Department of Electrical and Computer Engineering at Auburn University, Auburn, AL. Dr. Sadi earned his PhD in ECE from University of Florida, Gainesville, in 2017, MS from University of California at Riverside, USA in 2011 and BS from Bangladesh University of Engineering and Technology in 2010. Prior to joining Auburn University, he was a Senior R&D SoC Design Engineer at Intel Corporation in Oregon. Dr. Sadi's research focus is on developing algorithms/CAD techniques for implementation, design, reliability, and security of AI hardware. He was the recipient of SRC best in session award, Intel Xeon Design Group recognition awards, and National Science Foundation CRII award.