Low-Energy Acceleration of Binarized Convolutional Neural Networks using a Spin Hall Effect based Logic-in-Memory Architecture

Ashkan Samiee, Payal Borulkar, Ronald F. DeMara, *Senior Member, IEEE,* Peiyi Zhao, *Member, IEEE,* Yu Bai, *Member, IEEE,*

Abstract—Deep Learning (DL) offers the advantages of high accuracy performance at tasks such as image recognition, learning of complex intelligent behaviors, and large-scale information retrieval problems such as intelligent web search. To attain the benefits of DL, the high computational and energy-consumption demands imposed by the underlying processing, interconnect, and memory devices on which software-based DL executes can benefit substantially from innovative hardware implementations. Logic-in-Memory (LIM) architectures offer potential approaches to attaining such throughput goals within area and energy constraints starting with the lowest layers of the hardware stack. In this paper, we develop a Spintronic Logic-in-Memory (S-LIM) XNOR neural network (S-LIM XNN) which can perform binary convolution with reconfigurable in-memory logic without supplementing distinct logic circuits for computation within the memory module itself. Results indicate that the proposed S-LIM XNN designs achieve 1.2-fold energy reduction, 1.26-fold throughput increase, and 1.4-fold accuracy improvement compared to the state-of-the-art binarized convolutional neural network hardware. Design considerations, architectural approaches, and the impact of process variation on the proposed hybrid spin-CMOS design are identified and assessed, including comparisons and recommendations for future directions with respect to LIM approaches for neuromorphic computing.

Index Terms—In-memory computing, STT-MRAM, image processing, classifier systems, post-CMOS computing architectures.

1 Introduction

In recent years, the use of Deep Learning (DL) to perform computational tasks has substantially advanced Artificial Intelligence (AI) to more vast and useful tasks beyond image recognition to speech recognition [1], [2], statistical machine translation [3], and human-like reasoning activities like interpretation of abstract art [4], [5]. Among various DL models, the deep Convolutional Neural Network (CNN) was popularized after achieving high accuracy for various recognition challenges [6]. These favorable accuracy and performance characteristics of CNNs are obtained via the deep neural network structures which have become enabled through the recent availability of massively-sized datasets. While progress in the development and application of CNNs continues at a rapid pace, feasibility and scalability challenges remain to widespread the deployment of CNNs. CNNs require a significant amount of floating-point computation and fast memory storage of significant capacity. For example, VGG-19 involves more than 140 million floating-point (FP) parameters and 15 billion FP operations to recognize one image [7]. Therefore, during the training and inference of CNNs, large clusters of Central Processing Units (CPUs) and Graphics Processing Units (GPUs) are demanded [8]. Such CPU and GPU systems provide a compatible deep learning framework for Caffe [9], Theano [10], or TensorFlow [11]. Despite the computational demands, high accuracy performance is achievable and these systems are adaptable by the user for training a custom network with relatively little

This work was supported in part by the Center for Probabilistic Spin Logic for Low-Energy Boolean and Non-Boolean Computing (CAPSL), one of the Nanoelectronic Computing Research (nCORE) Centers as task 2759.006, a Semiconductor Research Corporation (SRC) program sponsored by the National Science Foundation through NSF-CCF-1739635.

engineering effort.

While CPUs and GPUs are currently the predominant devices for CNNs, FPGAs can achieve superior energy efficiency in terms of performance/watt relative to GPUs for many DL tasks [12]. FPGA-based CNN platforms aim at achieving human-like cognitive ability while consuming ultra-low power and achieving high energy efficiency. Such FPGA-based CNN platforms can enable a variety of CNN's applications under low-energy demands such as embedded computing, small mobile devices, and unmanned drones. A recent industry-based study has exploited acceleration of deep learning at the datacenter-scale seeking high cost efficiency through FPGAs [13]. From the academic community, there is also efforts on these FPGA-implemented architectures [14]. However, the gap between GPU and FPGA platforms in both performance and granularity of the computation has remained.

In order to overcome these issues, the Binary Neural Network (BNN) topology has been explored. A BNN is a low precision CNN with binary activations and binary weights, yet having accuracy comparable to full precision nets. Binary Neural Network (BNNs) strive to improve deep neural network efficiency using compact data types. To realize these ideas, the XNOR-Neural Network is proposed herein whereby the weights and the inputs are discretized into binary values. Such BNNs are readily implemented on FPGAs where their dominant computations are bitwise logic operations [12]. Previous experimental results have shown how a BNN can be significantly smaller than an equivalent network with single-precision weight values [15]. However, current FPGA-implemented BNNs require data to be physically moved frequently between flip-flops and logic circuits, which demands significant power consumption and incurs wasteful delays. In contrast, more recent works explore the computational concept of Logic-in-Memory [16], [17] which utilizes an array of cells with embedded logic capabilities and storage by leveraging spintronic devices. The new architecture can realize the computation without data transfer between the memory and logic circuits. Some previous works [18], [19] propose BNNs implemented with Resistive Random-Access Memory (RRAM) and Spin-Orbit Torque Magnetic Random-access Memory (SOT-MRAM) devices. Relative to these related approaches, the major contributions herein include:

- Development of a novel Spintronic-Logic-in-Memory platform using Spin-Orbit Torque Magnetic Random-access Memory (SOT-MRAM) using innovative in-memory approach to bypass data transfer between the memory and logic circuits.
- Novel circuits for binary convolution operations using SOT-MRAM devices. Such design embraces the intrinsic physical switching characteristics of SOT-MRAM devices to achieve lower energy, faster execution and higher accuracy.
- Experimental results show that the proposed Spintronic-Logic-in-Memory (LIM) XNOR neural network (S-LIM XNN) can achieve 1.2-fold lower energy, 1.26-fold faster execution and 1.4-fold higher accuracy when using XNOR-network for AlexNet on ImageNet.

The remainder of this paper is organized as follows: Section 2 introduces the related background and the motivation of the work. Section 3 proposes the LIM computation platform and binary convolution architecture. In section 4, we present the architecture of S-LIM XNN for AlexNet. Section 5 shows experimental results using S-LIM XNN for AlexNet on ImageNet to analyze recognition accuracy, area and energy efficiency. Finally, Section 6 concludes the manuscript.

2 Related Work

2.1 Convolution Neural Networks (CNNs)

Convolution Neural Networks (CNNs) utilize network topologies in which various thresholding layers are flanked together comprised of a convolution layer, batch normalization layer, activation layer, pooling layer, and fully connected layer. CNNs contains a *Convolution Layer*: The convolution layer is the core building

block of CNN. It performs complex convolution between the image set and the filter, shown as

$$f_{out}(x,y,z) = \sum_{i=0}^{h-1}\sum_{j=0}^{w-1}\sum_{k=0}^{c_{in}-1} f_{in}(x+i, y+j, k)\cdot c_z(i,j,k) \quad (1)$$

where i, j and k are the spatial coordinate of input feature map with the size of $h \times w \times c_{in}$. c_Z is the convolution kernel. Activation maps arise due to filters being stacked along the depth to form the full output volume of the convolution layer. *Pooling Layer*: The primary purpose of the pooling layer is to reduce the spatial size of representation and to reduce the number of parameters and computations required in the network. It also plays a crucial role in regulating over-fitting. The pooling layer is located between successive convolution layers and performs non-linear down-sampling. *Fully-connected layer*: The fully-connected layer performs weighted sum operation after several convolution layers and max-pooling layers. Neurons present in this layer have weighted connections to the activations obtained in the previous layer.

2.2 Logic-in-Memory (LIM) architecture

The quest for viable Logic-in-Memory (LIM) models dates back many decades in various forms including cellular arrays and cache structures [20] [21], while recent adaptations have included paradigms supporting Binary Neural Network (BNNs) [16]. The distinguishing feature of a conventional von Neumann architecture is its requirement that data travels to/from memory in order to conduct the computation. Thus, its data processing sequence consists of using distinct hardware elements performing memory access, data transport, computation, and result storage back to memory. Significant power dissipation and delays ensue relative to the computation step. In contrast, a Logic-in-Memory architecture enables local computations with minimal data transfer between memory and logic modules. Therein, the logic function is embedded within the memory device. Once data is loaded, results of the logic function already reside within the memory by skipping some intermediate read and store steps.

2.3 XNOR-Networks

XNOR-Networks have both their activation functions and weights binarized to either +1 or -1. The binarization of active functions and weights can be found in quantized, reduced-precision CNN models. Similarly to CNNs, XNOR-Networks normally consist of several layers: the convolution layer, pooling layer, batch normalization layer, and fully-connected layer [22] as identified below: Convolution Layer: In XNOR-network, computationally-intensive convolution is replaced by an XNOR operation between the input and filter vector, as depicted in Fig. 1 (b). The convolution between input (I) and weight filter (W) using a binary operation is shown below:

$$I * W = (sign(I) \circledast sign(W)) \odot K\alpha \quad (2)$$

where $\circledast$ represents XNOR operation which is followed by a bit-counting operation, α is a scaling factor of weight (average of $|W|$), K is a scaling factor for input I, and $\odot$ is element wise multiplication. Therefore, the convolution with multiplication and addition operations is simplified to an XNOR logic operation followed by a bit-counting operation. In Fig. 1 (c), the example of a binary convolution is shown. The binarized input matrix is XNORed with the binarized weight matrix. The Bitcount results are represented in the matrix entitled by $Bitcount(I \circledast W)$. Then, the scaling factor is applied to the matrix for obtaining the final result which is shown in the matrix of $Bitcount(I \circledast W) \odot K\alpha$.

Pooling Layer: Compared to a CNN pooling layer which features several pooling functions such as maximal pooling, minimal pooling, and average pooling, a XNOR-network pooling layer cannot be applied to binary data due to its significant information loss. Thus, XNOR-Networks insert a pooling layer between the convolution layer and the batch normalization layer. *Batch Normalization layer*: This layer performs normalization of input batches by considering their mean and variance. This reduces the amount of information lost during binarization by linearly shifting and scaling the input distribution to have zero mean and unit

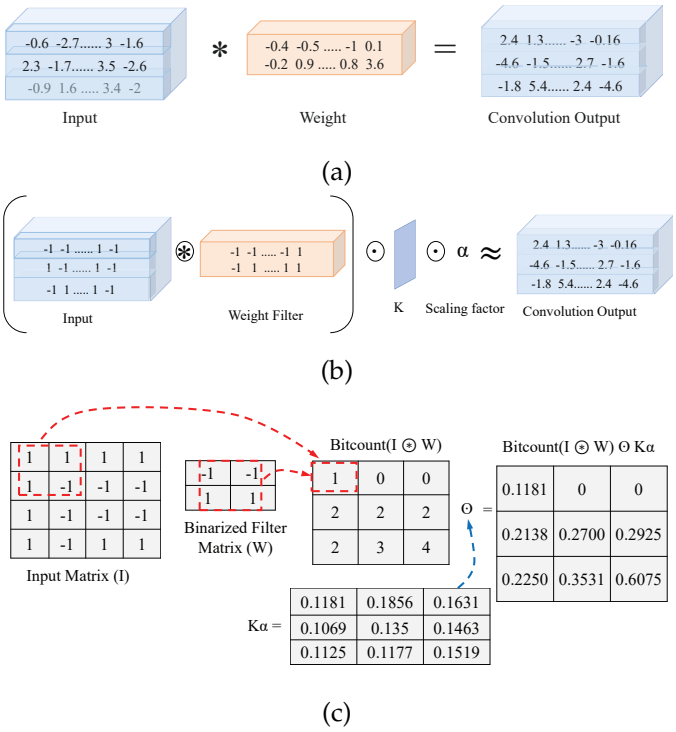


Fig. 1: Convolution Operation present in CNN and XNOR-Networks, (a) Convolution Operation in a CNN. (b) Convolution Operation in a XNOR-Networks. (c) Example of binary convolution in XNOR-network.

variance. Batch normalization accelerates the training process and regularizes the model by using normalization noise. It also restricts the training process from getting stuck in saturated regimes of non-linearities. *Fully-connected layer*: After several convolution layers, max-pooling layers, and batch normalization, the high-level reasoning in XNOR-Networks is done by the fully-connected layer. The final output produced by this layer is in the form of binary fmaps, which is different from the traditional CNN where the final output consists of real-valued fmaps. There are two approaches which can be adopted for binarization of data: a deterministic method and a stochastic method. In the stochastic binarization method, a hard-sigmoid function is used, and the concept of probability is used to assign -1 and +1 to original values of the weights. Compared to the deterministic binarization method, which is relatively straightforward to implement, the stochastic binarization method can be advantageous as it avoids use of the sign function. Thus, a Binarized neural network consists of various layers which are the convolution layer, the pooling layer, the batch normalization and binarization.

3 Logic-in-Memory Platform

3.1 Spin Hall Effect (SOT)-based MTJ device

In order to implement LIM for XNN efficiently, SOT-MRAM devices are employed to construct a spin-based S-LIM XNN. Fig. 2 (a) shows SOT-MRAM device architecture, which consists of two ferromagnetic layers separated by a thin tunneling barrier, which from a magnetization pinned layer and the free layer. The fixed layer has fixed magnetization. In contrast, the magnetization of the free layer depends upon the magnetic anisotropy [23], [24], which can be changed by a charge current and/or by an applied magnetic field passing through the device. Consequently, two ferromagnetic layers (the pinned layer and the free layer) can be used to construct two types of magnetization configurations that are parallel (P) and anti-parallel (AP). Thus, MTJ resistances in two types of magnetization configurations are calculated by using the Tunnel Magnetoresistance (TMR) effect. On the bottom of the device, the heavy-metal (HM) layer is used to pass current and change magnetic directions of the free layer. The free layer magnetic direction is determined by the orientation of applied current which defines the logic state of MTJ. Due to Spin-orbit Torque (SOT) coupling in heavy-metal (HM), electrons are diverted with different spins in opposite directions which give rise to the spin-injection current that is transverse to the applied charge current.

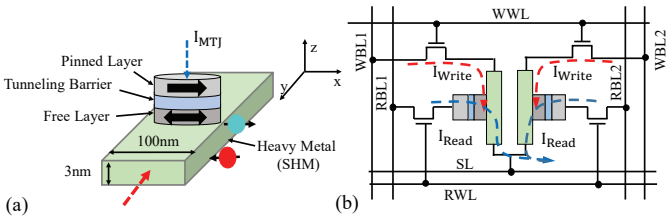


Fig. 2: (a) SOT-MRAM device structure. (b) The proposed Logic-in-Memory (LIM) architecture performing XNOR operations without transferring data outside of memory.

3.2 Computational Binary Convolution Architecture

Section 2 explains the computation-intensive convolution replaced by the XNOR operation

between input and filter vector. In this paper, we propose the use of SHE-MTJ device to build a computational XNOR architecture for performing XNOR logic operations. The architecture of the proposed XNOR operation is shown in Fig. 2 (b). The current flows (red line) through the HM loads to an accumulation of the directed spin. Consequently, due to SHE, the generated spin current produces SOT to the adjacent free layer, loading to a switch of magnetization. In Fig. 2 (b), two SOT-MRAM devices are connected in parallel as a single XNOR cell. Two SOT-MRAM devices are controlled by four access transistors associated with Write Word Line (WWL), Read Word Line (RWL), Write Bit Line (WBL1, WBL2), Read Bit Line (RBL1, RBL2), and Source Line (SL) to perform write and read operations. The WBL1 and WBL2 are connected with voltage drivers to provide the required write voltages. WWL acts to switch on the access transistor during a write operation. For sensing the data stored in two SOT-MRAM devices, two voltage mode Sense Amplifiers (SAs) are connected with the architecture. Two RBLs (RBL1 and RBL2) are connected with the read voltage (V_{read}). The write and read operations by using different biasing conditions are shown in TABLE 1. Besides memory operations, two SOT-MRAM devices are sensed simultaneously to implement an Logic-in-memory XNOR logic function. The binary tensor (-1, +1) are converted to voltages (+V, -V) that are used to switch the SOT-MRAM device between Anti-Parallel (AP) and Parallel (P) states.

TABLE 1: Biasing condition of the proposed Computational XNOR Architecture.

Operations	Write 1(0)	Read
WWL	V_{dd}	0
RWL	0	V_{dd}
RBL1	0	V_{read}
RBL2	0	V_{read}
WBL1	$V_{WP}(V_{WN})$	0
WBL2	$V_{WP}(V_{WN})$	0
SL	0	0

In the proposed architecture, two SOT-MRAM devices can be used to store two bits. The four XNOR combinations (-1, -1), (-1,1), (1,1) (1,-1) can be mapped on two SOT-MRAM devices via conversion to AP and P states. When programming the same two states (-1, -1) and (1, 1) on devices, the reading voltage can induce a compensated voltage through two devices, which can be sensed using a Sense Amplifier (SA). Fig. 3 shows the principle of the XNOR operation using two SOT-MRAM devices. In Fig. 3 (a), two SOT-MRAM devices are presented as two adjustable resistors (R_i and R_j). Two states (AP and P) are programmed on the devices to construct four combinations of two resistances. While a supplied voltage V_{read} is applied, the summed voltages flowing through two SOT-MRAM devices can take on four possible values V_{P-P}, V_{P-AP}, V_{AP-P}, V_{AP-AP}) depending on the SOT-MRAM configuration. The enhanced sensing circuit is proposed to distinguish between these values, which are generated by different MTJ configurations.

The possible values for V_{SA} are shown in Fig. 3 (b). In order to realize XNOR logic operation, we propose a sensing mechanism with two sensing amplifiers. The output voltage (V_{SA}) of CIM is connected to the positive input of the first amplifier and the negative input of the second amplifier. As shown via results in Fig. 3 (b), the possible values of V_{SA} for the same SOT-MRAM configurations (P-P or AP-AP) distribute the highest or lowest values. On the other hand, the different SOT-MRAM configurations (P-AP or AP-P) generate similar voltages which are less than the highest value and more than the lowest value. Consequently, only the case where both SOT-MRAMs are in the AP or P configuration leads to an output of logic '1' using two reference voltages (V_{refl} and V_{refr}) on the sensing amplifier. Two SOT-MRAMs with different configurations (AP-P or P-AP) generate logic '0' due to V_{SA} falling in the range between two reference voltages (V_{refl} and V_{refr}). In Fig. 3 (c), the simulation of the proposed design is presented within corresponding states.

Challenges facing SOT-MRAM include read disturbance, read decision failure, and sensitivities to process variation during fabrication. However, the LIM failure model differs from single SOT-MRAM in the writing and reading current flow through each bit cell. In order to have statistical data for analyzing the read

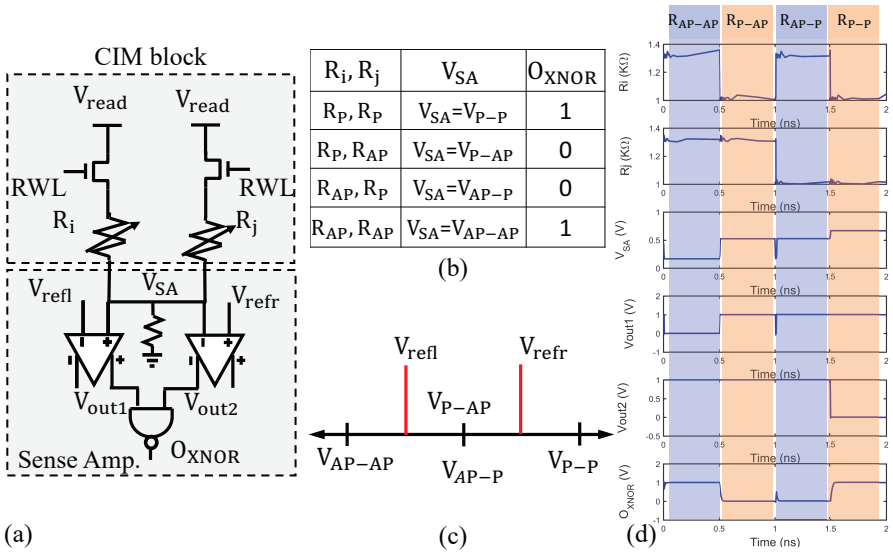


Fig. 3: Architecture and principle of the proposed LIM XNOR operation: (a) LIM operation, (b) current combination for LIM operation, (c) sensing reference voltages dor V_{SA}, and (d) simulation results of the proposed sensing circuits.

disturb and read decision failure, we apply the Monte-Carlo simulation on the proposed SOT-MRAM LIM architecture. Fig. 4 (a) shows the resistance distribution of a single SOT-MRAM device with one thousand time trials. In our simulation, we performed one thousand samples under the variations of MTJ oxide thickness ($\sigma/\mu = 2\%$), transistor V_T ($\sigma/\mu = 5\%$) [25], [26]. In Fig. 4 (b), the current probability density of four possible values (I_{P-P}, I_{P-AP}, I_{AP-P}, I_{AP-AP}) have been plotted. Compared to the single device read current, the proposed SOT-MRAM LIM has lower current value. However, the read margin is larger than normal read and other SOT-MRAM arrays, due to the differential sensing scheme. Furthermore, the left and right read margins are not equal. Theoretically, resistances are equally separated. However, currents are not upon calculation by inverses resistance.

In Fig. 4 (c), the transients simulation is provided to verify the functional operation of the proposed LIM XNOR architecture. Two operations (write and read) within four different cases are presented including input combinations (-1, -1), (-1,1), (1,1) (1,-1). For example, in the beginning, the two negative inputs (-1,-1) are applied to two SOT-MRAM devices. Consequently, both of the SOT-MRAM devices are written into parallel states. During the read process, the lower resistance in two SOT-MRAM devices is sensed. This way, the two SOT-MRAM devices can be programmed within 4 different resistance configurations (R_{P-P}, R_{P-AP}, R_{AP-P}, R_{AP-AP}). The layout of the proposed S-LIM-XNOR gate is shown in Fig. 5 (b). The proposed layout was designed using λ-based design rules [27], [28].

TABLE 2: Device simulation parameters utilized to simulate the proposed architecture.

Symbol	Description	Value
α	Damping Coefficient	0.07
t_{ox}	Qxide Barrier Thickness	1.2nm
t_{HM}	Heavy Metal Thickness	3nm
M_s	Saturation Magnetization	7.8×10^5A/m
Θ_{SHE}	Spin Hall Angle	0.3
A_{ex}	exchange stiffness	1.1×10^{11}J/m
ρ_{HM}	HM Resistivity	$200\mu\Omega \cdot$ cm
MTJ_{Volume}	Dimension of Free Layer	$100 \times 40 \times 3$nm^3
HM_{Volume}	Dimension of Heavy Metal	$100 \times 80 \times 3$nm^3

In order to analyze Process Variation (PV) impact on the proposed S-LIM-XNOR circuits, extensive circuit-level simulation results were performed. NCSU 45nm CMOS process standard cell library [29] was used alongside the SOT-MRAM model via circuit-level simulation through SPICE simulation software. Hence, 10,000 MC simulations were collected on a single S-LIM-XNOR cell in order to quantify the effects of PV on the proposed device structure. In this simulation, various deviations for the CMOS threshold and Tunnel magnetoresistance (TMR) were considered. The TMR (%) is a variable to define relative resistance change which is defined as $TMR := \frac{R_{AP}-R_P}{R_P}$. During

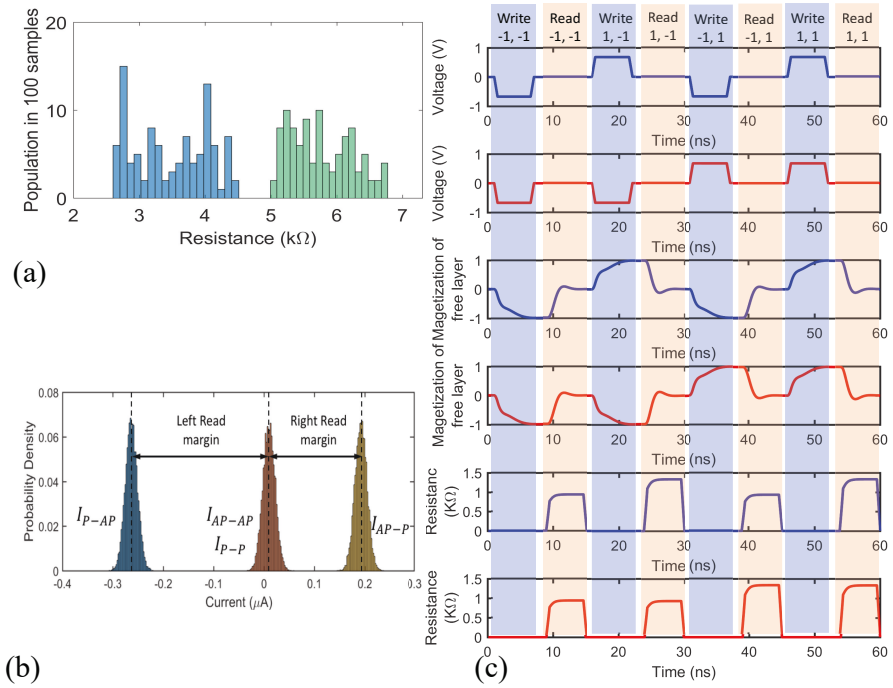


Fig. 4: (a) Resistance distribution of a single SOT-MRAM device, (b) probability density of currents during operation, (c) simulation of the proposed computational XNOR architecture.

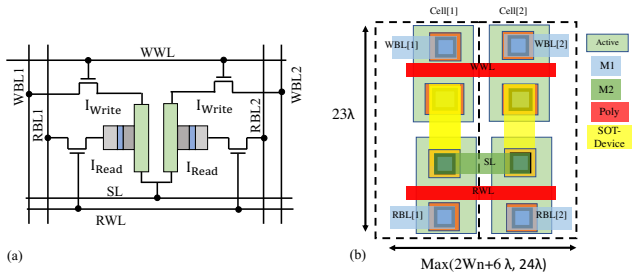


Fig. 5: (a) The schematic of the S-LIM XNOR cell, (b)Layout of the proposed S-LIM XNOR.

the simulation, values of V_{th}, W, L in CMOS devices vary in the SPICE netlist following the Gaussian distribution with the mean equal to the nominal model card, which is provided in [30]. In this paper, we employ the Bit Error Rate (BER) to quantify the error which is defined as the number of wrong output bits divided by all the input bits applied in both P and AP states. In Fig 6 (a), the BER average is presented based on 10,000 MC simulations. Considering 10% variation on TMR, all architectures exhibit the highest BER for $TMR = 100\%$. While TMR increases, the results exhibit further improvement in reliability. Among three recent BNN designs, the proposed S-LIM-XNOR conveys improved BER results due to its differential design characteristics. Fig 6 (b) (c) show the simulation results by considering the (W/L) ratio of PMOS and NMOS, respectively. Fig 6 (b) presents the simulation results with PMOS W/L=2 and NMOS W/L=1. The simulation results show improvement of BER with larger TMR. In Fig 6 (c), the PMOS W/L=4 and NMOS W/L=2 enhance reliability of the proposed architecture.

3.3 Data Mapping to Spintronic-LIM

In order to map the data on the LIM array, a conventional data transfer algorithm is not sufficient due to certain constraints created by SOT-MRAM device physics and LIM operation. Herein, the memory system consisting of several banks is defined. Each bank is constructed by an LIM array within rows and columns. Thus, the proposed LIM XNOR operation is performed on two data vectors if they satisfy three criteria: 1) two data vectors must be stored in the same bank, 2) two data vectors are mapped to different rows, 3) two data vectors are stored in the same set of columns. By satisfying these constraints, we propose a data allocation algorithm for the proposed LIM array. Firstly, we define a Memory access Data Flow Graph (MDFG) that is used to model the

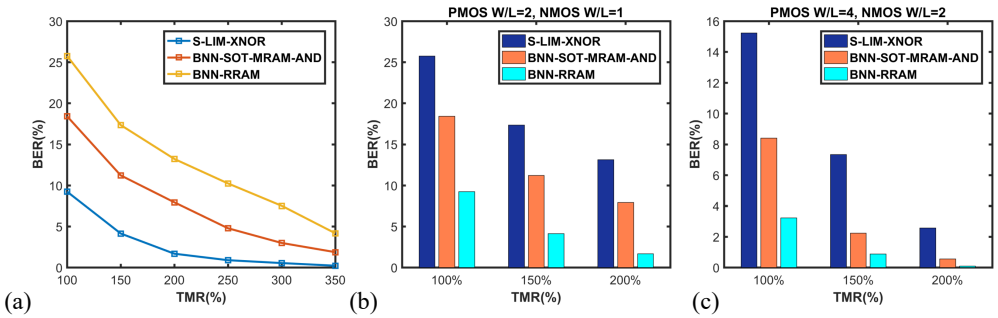


Fig. 6: (a) Average BER for $\delta TMR = 10\%, \delta Vth = 10\%$, (b) Average BER for $\delta TMR = 10\%, \delta Vth = 10\%$ with (W/L)P=2, (W/L)N=1, (c) Average BER for $\delta TMR = 10\%, \delta Vth = 10\%$ with (W/L)P=4, (W/L)N=2.

processing unit to visit call LIM XNOR logic operation. Herein, we use an MDFG which is a node-weighted directed graph by G_{MDFG}. In the data allocation algorithm, the MDFG graph is defined with several variables where V is input with an amount of memory nodes, E is the set of edges, D is a set of data, v_i is the ith task, N_r is the read operations, and N_w is the write operations. Thus, the total energy consumption is $EN_h = E_r * N_r + E_w * N_w$. The above equation is referred to as the $compute\_energy$ function herein. In Algorithm 1, a data remapping allocation is developed utilizing a node-weighted directed graph by G_{MDFG} with parameters input to the algorithm. In line 1, $compute\_energy$ functions calculates the energy of the operation. If the energy requirement is less than the maximum energy max_e, and the total write operation is less than the write threshold T_{r_w}, the algorithm evaluation commences, as shown in line 2. In Algorithm 1, the proposed algorithm can select two types of writing operations in line 7 and 14. *Type 1*: In this type, element to element operations are performed between two arrays that are X and Y. The LIM bank takes a vector from Bank X and maps it to the first row. Then, the vector from bank Y is mapped to the second row of the LIM bank, as shown in Fig. 7. To effectively utilize LIM for this type, we utilize the array alignment technique. Here alignment of elements of X[k] and Y[k] for any value of k is done in which X XNOR Y can be converted to the LIM operation. An extension of this technique is row interleaved placement. This technique is mostly used for larger data structure that is not present in the same memory bank. It ensures that the two elements on which the operation is to be performed are mapped to the same bank. *Type 2*: In this type, operations are performed between a small array X and a large array Y. A column replication technique is used for this type of operation, in which a single element of the small array X is replicated across the column to fill an entire row. These elements of X and Y become properly aligned, and LIM XNOR operations are performed on them. It has little initial overhead during the replication stage, and has significantly fewer access relative to the larger array Y. Fig. 7 shows the two types' modes of data mapping.

Algorithm 1: Data allocation algorithm for the proposed LIM.

Input : Given targets $G_{MDFG} = (V, E, D, v, N_r, N_w)$; a write threshold T_{r_w}; writing operation max_w
Output: A data allocation and task scheduling with minimized energy

1 $EN(h) = compute\_energy(h)$
2 **if** $cw \leqslant T_{r_w}$ *and* $EN(h) < max_e$ *do* **then**
3 $L \leftarrow h$
4 $L_{eff} = parallel\_alg(L)$
5 $(A[M][N], B[M][N]) = data\_comp(L_{eff})$
6 **for** *i=1 to M* **do**
7 **if** *Type1=1* **then**
8 **for** *j=1 to N* **do**
9 $A[i][j] = Data_X(j)$
10 $A[i+1][j] = Data_Y(j)$
11 **end**
12 **end**
13 **else**
14 **if** *Type2=1* **then**
15 **for** *j=1 to N* **do**
16 $A[i][j] = Data_X(i)$
17 $A[i+n][j] = Data_Y(j)$
18 **end**
19 **end**
20 **end**
21 **end**
22 **end**

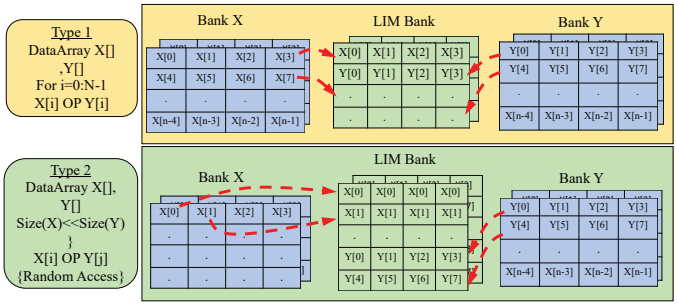


Fig. 7: Data mapping for two computation types.

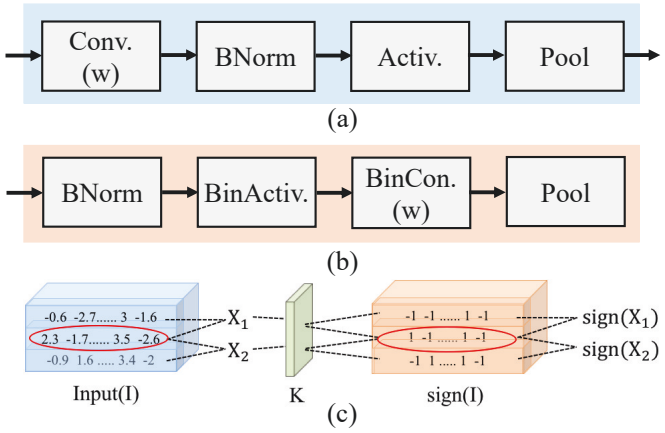


Fig. 8: (a) Conventional CNN blocks, (b) XNOR-network blocks, (c) Example of overhead problem in binary activation layer.

4 Spintronic-Logic-in-Memory (LIM) XNOR Neural Network (XNN)

In BNNs, there are two efficient approximations to the conventional CNN: Binary-weight network and XNOR-network. In Binary-Weight-Networks, the conventional filters are approximated with low precision binary values. Inputs to convolution layers are tensors in real-value. In XNOR-networks, weights and inputs are binarized. Thus, the conventional convolution is implemented using XNOR logic and bitcounting operations.

4.1 Training XNOR-networks

Compared to conventional CNN blocks as shown in Fig. 8 (a), a XNOR-network has different functional blocks. A conventional CNN consists of several functional layers such as the convolution layer, the batch normalization layer, the activation layer, and the pooling layer, as shown in Fig. 8 (a) respectively. Besides the conventional layers which are introduced in the previous section, the batch normalization layer is used to normalize the input batch using its mean and variance. The activation layer is modeled by an element-wise non-linear function, e.g., Sigmoid or ReLU.

The pooling layer takes any types of pooling (e.g. max, min, and average) on the input batch. In Fig. 8 (b), the order of the functional layer of a XNOR-network is presented. Compared to CNN, the pooling layer of the receiving binary input batch induces a significant information loss. For example, the min-pooling layer takes the binary input batch and returns a tensor that most of its elements are equal to -1. Thus, in order to avoid such a significant loss of information, the pooling layer is directly connected after the convolution layer (BinConv). To overcome the information loss issue caused by binarization, in the XNOR-network, we normalize the input before it becomes binarized. The normalized process enforces the input to hold a mean of zero, whose thresholding at zero leads to less binarization error. The binary activation layer, labeled BinActiv., is used to compute K and $sign(I)$ as delineated in Eq. 2. The convolution process requires computing the scaling factors β for all possible sub-tensors in input (I), which is the same size as weight (W), where weight $W \in \mathbb{R}^{c\times w\times h}$, input $I \in \mathbb{R}^{c\times w_{in}\times h_{in}}$, and $w_{in} \gg w, h_{in} \gg h$. The sub-sensors are illustrated in Fig. 8 (c) by X_1 and X_2. However, due to overlaps between two sub-tensors (X_1 and X_2), the binary activation layer computes a large number of redundant operations, depicted in Fig. 8 (c). To overcome the redundant computation issues, a matrix $A = \frac{\sum |I_{:,:,i}|}{c}$ is computed as the average of absolute values of the elements in I across the channel. Thus, A convolves with a 2D-filter $k \in \mathbb{R}^{w\times h}$. Then, the output of such convolution $K = A * k$ consists of scaling factors for all sub-tensors in the input I, , where $\forall ij, k_{i,j} = \frac{1}{w\times h}$. K consists of scaling factors β for all sub-tensors in the input I. K_{ij} represents the corresponded to β for a sub-tensor centered at the location (ij) across width and height. Once the scaling factor α for the weight and scaling factors β for all sub-tensors

in the input I (K), the convolution between input I and weight W can be approximated using binary operations. The detailed information is described in reference [22]. In the convolution layer, labeled BinConv, the inputs K and $sign(I)$ are computed using binary convolution which is explained in the previous section. In the last layer, the pooling operation is applied. It is noted that non-binary operations is a small portion compared to binary operations. Once the proposed architecture has been designed, the training algorithms for binary gradient descent and k-bit quantization are applied to the training datasets, which can occur off-line.

4.2 Hardware Architecture of Spintronic-Logic-in-Memory (LIM) XNOR Neural Network (XNN)

The BNN approach reduces computational complexity due to the use of binarized parameters and operations compared to well-trained CNNs that employ floating-point operations. In terms of hardware considerations, BNNs reduce memory size and bandwidth requirements. In addition, the binary operation used to replace floating-point operations can be performed efficiently on the hardware. The proposed S-LIM XNN with LIM XNOR operation is well-suited to BNN. Thus, in this section, we evaluate AlexNet on the proposed hardware architecture. AlexNet is a deep CNN which performs ImageNet classification with high accuracy. For comparison, Rastegari [22] converted AlexNet with five convolution layers and two fully connected layers to binary AlexNet. In order to enable a classification of AlexNet to output a heatmap, we convert fully-connected layers into convolution layers. Similar methods have been employed in recent research work [22], [31] as shown in Fig. 9 (a).

In Fig. 9, the architecture of the proposed S-LIM XNN capable of AlexNet processing is presented. In our experimental results, the approximation of convolution by using XNOR-net can accelerate each convolution operation by 56.13-fold based on the same CPU configuration. We also observe that the convolution layer with a small channel and filter size by using XNOR-net convolution approximation does not incur considerable speedup. Therefore, the first layer Conv1 and the last layer Conv8 remain as real number operations because in the first layer, the channel size is three and in the last layer the filter size is 1×1, as shown in Fig. 9 (a). Such observation is also discussed in [22]. The computational block (Conv2-Conv7) in the proposed S-LIM XNN is made by four consecutive blocks, which are 1) Batch Normalization, 2) Binary Activation, 3) Binary Convolution, and 4) Pooling as depicted in Fig. 9 (a). To implement such computational blocks, we split four blocks into two hardware units: 1) Computational Unit (CU) 2) Spintronic-Logic-In-Memory (S-LIM) unit. The architecture is illustrated in Fig. 9 (b). In Fig. 9 (c), input (I) and weight (w) are mapped into Image Bank and Kernel Bank. Then, the Image Bank and Kernel Bank are written into the proposed S-LIM by switching on write mode of SOT-MRAM device. Before writing to the S-LIM, the real number input is binarized and signed by using the Sign-Func module. According to the proposed LIM-based computational XNOR architecture, the input tensors in the form of -1 and +1 are converted to a positive and negative voltage. Thus, according to the XNOR operation, we obtain results of 0 or 1 using sense amplifiers. XNOR and Bi-counting ($BitCount(XNOR(I(B), W(B))$) approximates the binary convolution of two vectors I and W. Such XNOR operation implements on the proposed S-LIM architecture without requiring logic circuits, which enables the Logic-in-Memory computations. Therefore, the data used for binary convolution stores within memory and avoids the read and write operation of the memory, which leads to limited throughput of bitwise operations. In this paper, we add the CU so that the input batch is binarized and normalized to distributed as zero mean and unit variance in order to ease the rate of information loss, which is realized as follows:

$$I_o(R) = \frac{I_i(R) - \mu}{\sqrt{\sigma^2 + \varepsilon}}\gamma + \beta \tag{3}$$

where μ and σ are mean and variance, respectively. The parameters ε, γ and β are to be learned during the training process. In Sign-Func computation, the input and weight with

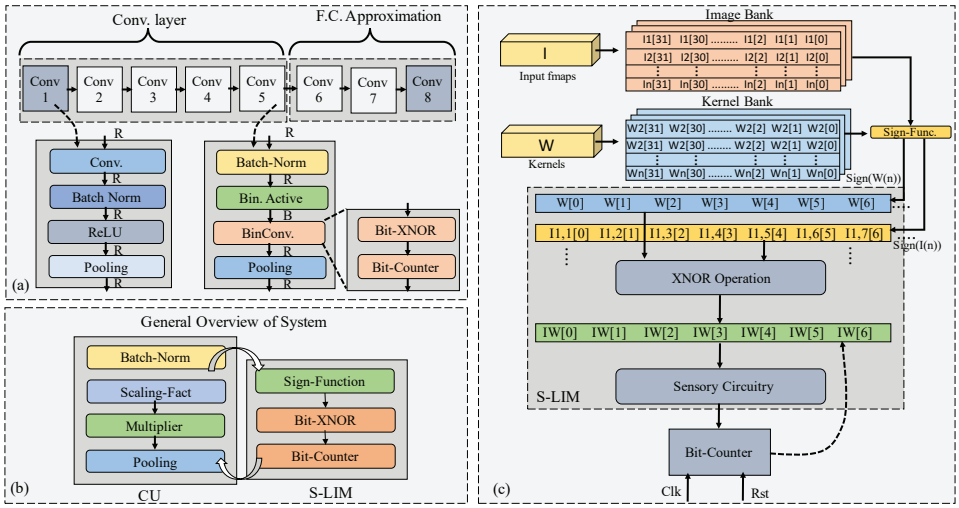


Fig. 9: (a) Architecture of AlexNet implemented by BNN where the pipeline indicates Real number values (R) and Binary number values (B), (b) overview of the system, and (c) dataflow within the architecture.

binary presented real numbers are converted to sign binary number. Then, they input to CIM for convolution computation. The computation equation can be delineated as:

$$I(B) = sign(I(R)), W(B) = sign(W(R)) \quad (4)$$

Scaling-Function is used to compute scaling factors for binary convolution and sign-function. The factor α is given by the following equation:

$$\alpha = \frac{||W(R)||_{l1}}{n} \quad (5)$$

Multiplication is applied at the output of bit-counting. In this process, the CU multiplies scaling factors α and K with bit-counting output to approximate convolution. The equation realized during this process is given as:

$$I * W = BitCount(XNOR(I(B), W(B)) \odot K\alpha \quad (6)$$

5 Experimental Results

In order to analyze the proposed S-LIM-XNN, we have utilized a hierarchical simulation framework including circuit-level and application-level simulations. In the circuit-level evaluation, the SOT-MRAM device is leveraged in SPICE simulation to verify the designated functional circuits [32], [33]. At the application level, we used Ubuntu 16.04-based framework provided compatibility with various deep learning software packages and libraries. The architecture of AlexNet, ResNet-18, and VGG-19 are evaluated on datasets e.g., CIFAR-10, MNIST, and SVHN. In this paper, we also evaluate the S-LIM-XNN on the ImageNet (ILSVRC2012) dataset. ImageNet provides $\backsim$1.2M training images within 1K categories and 50K validation images.

Algorithm 2 delineates the procedure to train the proposed S-LIM XNN. First, we compute $\mathcal{A}$ and $\mathcal{B}$ for filters at each layer. Then, the forward propagation algorithm is performed using the weights and scaling factors. After results of the BinaryForward function have been produced, the backward propagation is applied based on the gradients. In the last step, we update the parameters and the learning rate by updating algorithms, such as SGD update with momentum or ADAM [34].

5.1 Memory Storage Comparison

Using the proposed training algorithm on the ImageNet large-scale dataset, the required storage memory for three different architectures (AlexNet, ResNet-18, and VGG-19) [18], [22], [35] are simulated. In order to train the network easily and maintain acceptable accuracy, the first and last layer data within these three XNN architectures remain as full-precision values. It is noted that compared to the theoretical analysis, the real implementation has degraded due

Algorithm 2: Training an L-layers S-LIM XNN [22]

Input : Given targets (I,Y), cost function $C(Y, \hat{Y})$, initial weight W^t, and current learning rate η^t
Output: update weight W^{t+1} and learning rate η^{t+1}

Binarizing weight filters:
for *l = 1 to L do* **do**
  for *k^{th} filter in l^{th} layer do* **do**
    $\mathcal{A}_{lk} = \frac{1}{n}||W^t_{lk}||_{l1}$
    $\mathcal{B}_{lk} = sign(W^t_{lk})$
    $\widehat{\mathcal{W}}_{lk} = \mathcal{A}_{lk}\mathcal{B}_{lk}$
  end
  $\hat{Y} = Binaryforward(I, \mathcal{B}_{lk}, \mathcal{A}_{lk})$
  $\frac{\partial C}{\partial W} = BinaryBackward(\frac{\partial C}{\partial \hat{Y}}, \widehat{W})$
  $W^t = UpdateParameters(W^t, \frac{\partial C}{\partial W}, \eta^t)$
  $\eta^{t+1} = UpdateLearningrate(\eta^t, t)$
end

to the last layer which often consumes a large number of parameters. In general, the XNN can reduce by 10-fold the storage memory compared to a full-precision float-CNN as shown in Fig. 11 (a).

5.2 Classification Accuracy on Top-1 and Top-5 Across Training Epochs

In this subsection, we train the proposed S-LIM XNN with a well-known ImageNet classification task. In Fig 10, the accuracy of different methods are plotted. In every iteration of training, the input images with 256 pixels are randomly selected for training. Fig 10 shows simulation results for running 20 epochs. The batch size limit was set to 512. To implement a loss function, we employ negative-log likelihood with the soft-max function. In the proposed AlexNet case, the Local-Response-Normalization (LRN) layer is not employed. In order to update parameters, we utilize an adaptive moment estimation (ADAM) optimization method due to its faster convergence and better accuracy for binary inputs [36], [37]. We initiate the learning rate at 0.1 and decrease the learning rate by 0.01 every four epochs. In the inference process, the image input is resized to 224 for forward propagation. To evaluate the performance, the Top-1 error rate and Top-5 error rate are employed. It refers to the benchmarking method of machine learning systems which basically means that for a given image, if the target label is the model's top prediction or the correct label is in the models top 5 predictions then the image is correct. Fig. 10 (a) delineates classification accuracy on Top-1 using the ImageNet dataset upon the number of training epochs. In Fig. 10 (a), we compare the proposed S-LIM XNN, BinaryNet (BNN) [37], and BNN based on AND-bitcount operation [18] with classification accuracy. It is noted that the BNN based on the AND-bitcount operation performs less accurately than S-LIM XNN, which shares similar architecture and learning algorithms. In BNNs architecture based on AND-bit-counting, the binary convolution is approximated by the AND logic operation and the bitcounting operation. The AND logic operation incurs information loss due to this approximation. For example, if $A = 00010$ and $B = 01111$, the $bitcount(XNOR) = 2$ and $bitcount(AND) = 1$. Moreover, S-LIM-XNN has unique binarization and block structure compared to the BinaryNet (BNN). For the binarization, S-LIM-XNN finds optimal scaling factors at each iteration of training. The blocks structure shown in Fig. 8 has a unique order that decreases the quantization loss for the training process. In all epochs, the BinaryNet (BNN) results in reduced accuracy compared to the proposed S-LIM XNN and BNN based on AND-bitcount, due to different binarization methods and network structures. Furthermore, the BNN-based training algorithm is based on deterministic binarization. Fig. 10 (b) presents classification accuracy on Top-5. The simulation results substantiate these claims and observations.

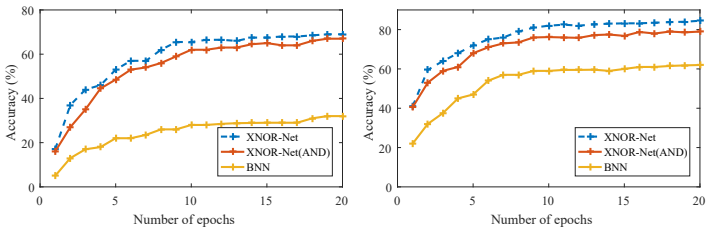


Fig. 10: (a) Classification accuracy on Top-1 using ImageNet dataset upon number of training epochs, and (b) Classification accuracy on Top-5 using ImageNet dataset upon number of training epochs.

5.3 Area and Energy Estimation in Different Layers

In this subsection, AlexNet for ImageNet is evaluated in terms of area and energy consumption. In order to estimate area and energy, we utilize a comprehensive device-to-architecture evaluation method. The proposed circuit-level simulation is performed on SPICE simulation software with the NCSU 45nm CMOS process standard cell library [29]. The SOT-MRAM is modeled and initiated with parameters shown in Table 2. The NEGF approach is employed to realize MTJ resistance (R_{MTJ}) upon the calculated heavy metal resistance (R_{SHM}). The area and power consumption of each circuit element is listed in Table 3. In circuit-level simulation, we use the system level memory evaluation tool NVsim to estimate the area and energy consumption. We implement the proposed S-LIM with the parameter obtained by SPICE simulation on the configuration files to construct a new bank, mat, and subarray [38].

TABLE 3: Area and Power Cost of Circuit Elements.

	Area	Power
8bit DAC	3096T	$30mW$
Sense Amplifier	244T	$0.25mW$
8bit ADC	450T	$35mW$
1bit ADC	244T	$1.73mW$
4bit ADC	72T	$12mW$

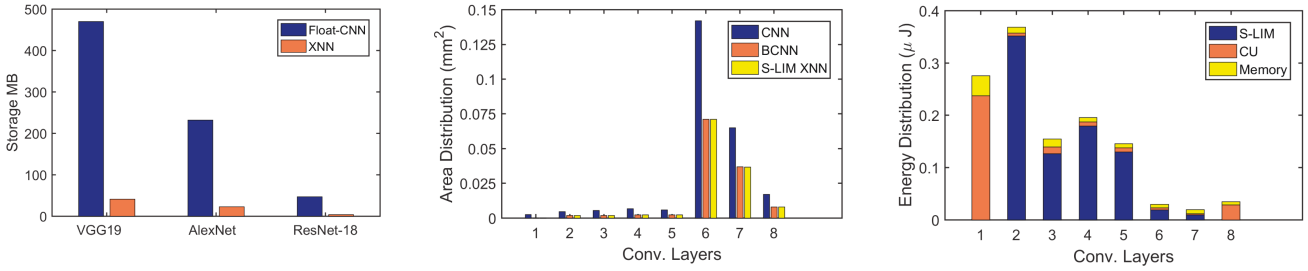
$T = \frac{W}{L} \cdot 3F^2$, $W/L = 3$ and technology node $F = 45nm$
The power consumption of S-LIM-XNOR Cell is estimated by $V_{avg}^2 g_{avg}$, where $g_{avg} = \sqrt{g_{on} g_{off}}$ [38].
The ADC system clock is assumed as 100MHz. The energy consumption is assumed by considering digital arithmetic logic and memory access within 45nm CMOS technology [39].

According to the device's parameters listed above, a plot of the area distribution for each of the convolution layers is shown in Fig. 11 (b). Although we convert fully connected layers to convolution layers, the last three layers consume excessive area due to numerous weights which require a large number of subarrays. In Fig. 11 (b), we delineate the area distribution for AlexNet in three different implementations (CNN, BCNN, S-LIM XNN). Compared to CNN and BCNN, S-LIM XNN results in less area in all convolution layers. Among BNN AlexNet architecture, the BCNN and S-LIM XNN remain similar in area distribution.

The energy distribution is shown in Fig. 11. In this figure, we categorize the energy consumption into several items: memory, S-LIM, and CU. It is noted that, in most of the layers, S-LIM consumes significant energy compared to the other components, due to intensive binary convolution steps. In the first and last layers, CU dominates the majority of energy consumption since those were not binarized.

5.4 Performance Comparison of Different Accelerators for AlexNet

Comparing the proposed S-LIM XNN to different accelerators of AlexNet can result in some noteworthy conclusions. For instance, Table 4 lists computation area, energy, and execution time of different accelerators for AlexNet on the ImageNet dataset. In general, although BNN consists of various computational layers such as scaling process which requires extra computations using multipliers, the BNN-based implementations can excel over CNNs for the metrics of area, energy and execution time. To evaluate the cost of the scaling operation, we perform an efficiency analysis. Namely, we assume the conventional convolution has total number of operations ($cN_W N_1$), where c is the number of channels, $N_W = wh$ and $N_1 = w_{in}h_{in}$. On the other side, the binary convolution method has $cN_W N_1$ and N_1 non-binary operations. On computing units able to perform n binary operations in one clock cycle, where n is a positive integer, then, the proposed method can achieve speedup by $S = \frac{cN_W N_1}{\frac{1}{n}cN_W N_1 + N_1}$. Thus, our efficiency analysis concludes results similar to reference [22]. Among various hardware realizations of BNN, BNN-ASIC consumes the least area and energy. Table 4 also lists comparisons to present various platforms including NVIDIA Jetson TK1 GPU (GPU1), server-class NVIDIA Tesla K40 GPU (GPU2), and Xilinx Zynq-7000 SOC FPGA [12]. Although FPGA- and GPU-implemented BNNs achieve accelerated execution, their designs can consume excessive energy and area, compared to the potential implementations with emerging devices such as RRAM and SOT-MRAM. For instance, the BNN-RRAM employs an RRAM-based crossbar



(a) (b) (c)

Fig. 11: (a) The required memory for binary and floating-weights in three different architectures(AlexNet, ResNet-18 and VGG-19), (b) The area distribution of different convolution layers in different implementations of the AlexNet, (c) the energy distribution of different convolution layers for AlexNet.

to store the weights and multiply with matrix-vectors. However, the BNN-RRAM design faces some challenges: 1) the RRAM device is limited by resistance precision; 2) RRAM is impacted by writing variation and endurance reliability concerns; 3) RRAM requires matrix splitting due to limited array size leading to additional writing and sensing circuity; and 4) RRAM requires larger peripheral circuits such as buffers and DAC/ADC, which consumes more than 85% of the area and energy. Therefore, XNOR-network employing SOT-MRAM can offer a favorable alternative to overcome these issues. Compared to BNN-RRAM, the designs using SOT-MRAM devices can attain improved performance in terms of area and energy consumption. The proposed S-LIM XNN approach acts to significantly reduce power consumption, latency, and error rate stemming from several perspectives. At the architectural level, the proposed accelerator employs a Logic-in-Memory concept to reduce some data transfer time and energy. Namely, the proposed XNOR accelerator has two functionalities: memory storage and XNOR logic operation. Thus, the XNOR accelerator can process XNOR logic operation within data which is stored in the accelerator without requesting CU to fetch data from memory, perform the logic operation, and store data back to memory. Consequently, during instruction execution, the associated offloading of the memory removes these unnecessary data movements between CU and memory. Similar to our conclusion, the latest papers with Logic-in-Memory tool reach similar observations using various approaches [40], [41], [42], [43]. At the algorithm level, the XNOR-net employs binary convolution to approximate the computationally- intensive conventional transformation to sufficient accuracy to achieve suitable operation at a fraction of the complexity. According to the algorithm, non-binary operations which require the communication between CU and XNOR accelerator are much less frequently executed, than the binary operations which are mainly performed in XNOR accelerator. In section 5.3, we considered the power consumption and performance based on the number of blocks and number of operations of each block which are involved in the computation to substantiate the results. Among the two designs based on the SOT-MRAM device, the proposed S-LIM XNN can achieve 1.2x lower energy, 1.26x faster execution, and 1.4x higher accuracy. These significant contributions arise from the S-LIM XNN which exploits the physics of the SOT-MRAM device to implement an XNOR logic operation, which is more accurate than the AND bit-counting operation. Furthermore, such architectures naturally construct a differential logic operation which leads to less energy consumption, shorter sensing time, and larger sensing margin. Finally, the write endurance with MRAM devices is typically noted as very high, especially in the case of 3-terminal SOT-MRAM devices utilized herein which isolate the oxide barrier from write current.

6 Conclusion

In summary, an S-LIM XNN design based on XNOR topology using SOT-MRAM devices can offer a direct computational method for energy-efficient binary convolution within memory. The approach leverages the inherent physical

TABLE 4: Comparison of different accelerators for AlexNet on ImageNet.

	Area ($mm2$)	Energy ($\mu J/Img$)	Exe.Time (ms)
CNN-RRAM[19]	21.25	544.85	-
BNN-RRAM[19]	9.19	2275.34	-
BNN-FPGA[12]	-	27918	5.95
BNN-GPU1[12]	-	324020	90
BNN-GPU2[12]	-	23725	0.73
BNN-ASIC[12]	19	352	-
BNN-AND[18]	5.28	310.42	10.7
S-LIM XNN	4.74	309.78	8.75

behavior of the storage device to reduce energy consumption during computation and cut down the execution time, while sustaining accuracy. The net impact of these methods are captured by the simulation results indicating 1.2-fold lower energy, 1.26-fold faster execution and 1.4-fold higher accuracy than other recent BNN-based AND bit-counting designs using competing device approaches.

REFERENCES

[1] X. He, K. Wang, H. Huang, T. Miyazaki, Y. Wang, and S. Guo, "Green resource allocation based on deep reinforcement learning in content-centric iot," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2018.

[2] T. N. Sainath, A. r. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 8614–8618.

[3] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014. [Online]. Available: http://arxiv.org/abs/1409.0473

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[5] M. Alawad, Y. Bai, R. DeMara, and M. Lin, "Robust large-scale convolution through stochastic-based processing without multipliers," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2017.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Computer Science*, 2014.

[8] A. Coates, B. Huval, T. Wang, D. J. Wu, A. Y. Ng, and B. Catanzaro, "Deep learning with COTS HPC systems," in *International Conference on International Conference on Machine Learning*, 2013, pp. III–1337.

[9] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," pp. 675–678, 2014.

[10] T. D. Team, R. Alrfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, and A. Belikov, "Theano: A python framework for fast computation of mathematical expressions," 2016.

[11] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, and M. Devin, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2016.

[12] R. Zhao, W. Song, W. Zhang, T. Xing, J.-H. Lin, M. Srivastava, R. Gupta, and Z. Zhang, "Accelerating Binarized Convolutional Neural Networks with Software-Programmable FPGAs," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '17. New York, NY, USA: ACM, 2017, pp. 15–24. [Online]. Available: http://doi.acm.org/10.1145/3020078.3021741

[13] G. Venkatesh, E. Nurvitadhi, and D. Marr, "Accelerating deep convolutional networks using low-precision and sparsity," *CoRR*, vol. abs/1610.00324, 2016. [Online]. Available: http://arxiv.org/abs/1610.00324

[14] N. Suda, V. Chandra, G. Dasika, A. Mohanty, Y. Ma, S. Vrudhula, J.-s. Seo, and Y. Cao, "Throughput-Optimized OpenCL-based FPGA Accelerator for Large-Scale Convolutional Neural Networks," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '16. New York, NY, USA: ACM, 2016, pp. 16–25. [Online]. Available: http://doi.acm.org/10.1145/2847263.2847276

[15] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," *CoRR*, vol. abs/1603.05279, 2016. [Online]. Available: http://arxiv.org/abs/1603.05279

[16] D. Pala, G. Causapruno, M. Vacca, F. Riente, G. Turvani, M. Graziano, and M. Zamboni, "Logic-in-Memory architecture made real," pp. 1542–1545, 05 2015.

[17] A. Grossi, C. Zambelli, P. Olivo, P. Pellati, M. Ramponi, C. Wenger, J. Alvarez-Hrault, and K. Mackay, "An automated test equipment for characterization of emerging mram and rram arrays," *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 2, pp. 269–277, April 2018.

[18] D. Fan and S. Angizi, "Energy efficient in-memory binary deep neural network accelerator with dual-mode sot-mram," in *2017 IEEE 35th International Conference on Computer Design (ICCD)*. IEEE, 2017, pp. 609–612.

[19] T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang, "Binary convolutional neural network on rram," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2017, pp. 782–787.

[20] H. S. Stone, "A logic-in-memory computer," *IEEE Trans. Comput.*, vol. 19, no. 1, pp. 73–78, Jan. 1970. [Online]. Available: https://doi.org/10.1109/TC.1970.5008902

[21] W. H. Kautz, "Cellular logic-in-memory arrays," *IEEE Transactions on Computers*, vol. C-18, no. 8, pp. 719–727, Aug 1969.

[22] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," *CoRR*, vol. abs/1603.05279, 2016. [Online]. Available: http://arxiv.org/abs/1603.05279

[23] F. A. Shah, V. K. Sankar, P. Li, G. Csaba, E. Chen, and G. H. Bernstein, "Compensation of orange-peel coupling effect in magnetic tunnel junction free layer via shape engineering for nanomagnet logic applications," *Journal of Applied Physics*, vol. 115, no. 17, p. 17B902, 2014. [Online]. Available: http://dx.doi.org/10.1063/1.4863935

[24] Z. He and D. Fan, "Energy efficient reconfigurable threshold logic circuit with spintronic devices," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 2, pp. 223–237, April 2017.

[25] K.-W. Kwon, X. Fong, P. Wijesinghe, P. Panda, and K. Roy, "High-density and robust stt-mram array through device/circuit/architecture interactions," *IEEE Transactions on Nanotechnology*, vol. 14, no. 6, pp. 1024–1034, 2015.

[26] Y. Bai, D. Fan, and M. Lin, "Stochastic-based synapse and soft-limiting neuron with spintronic devices for low power and robust artificial neural networks," *IEEE Transactions on Multi-Scale Computing Systems*, vol. PP, no. 99, pp. 1–1, 2017.

[27] S. K. Gupta, S. P. Park, N. N. Mojumder, and K. Roy, "Layout-aware optimization of stt mrams," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '12. San Jose, CA, USA: EDA Consortium, 2012, pp. 1455–1458. [Online]. Available: http://dl.acm.org/citation.cfm?id=2492708.2493064

[28] Y. Seo, K. 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, Sept 2016.

[29] C. H. Oliveira, M. T. Moreira, R. A. Guazzelli, and N. L. Calazans, "Ascend-freepdk45: An open source standard cell library for asynchronous design," in *Electronics, Circuits and Systems (ICECS), 2016 IEEE International Conference on*. IEEE, 2016, pp. 652–655.

[30] Y. Ye, F. Liu, S. Nassif, and Y. Cao, "Statistical modeling and simulation of threshold variation under dopant fluctuations and line-edge roughness," in *2008 45th ACM/IEEE Design Automation Conference*, June 2008, pp. 900–905.

[31] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[32] K. Jabeur, G. D. Pendina, G. Prenat, L. D. Buda-Prejbeanu, and B. Dieny, "Compact modeling of a magnetic tunnel junction based on spin orbit torque," *IEEE Transactions on Magnetics*, vol. 50, no. 7, pp. 1–8, July 2014.

[33] G. Prenat, K. Jabeur, P. Vanhauwaert, G. D. Pendina, F. Oboril, R. Bishnoi, M. Ebrahimi, N. Lamard, O. Boulle, K. Garello, J. Langer, B. Ocker, M. Cyrille, P. Gambardella, M. Tahoori, and G. Gaudin, "Ultra-fast and high-reliability sot-mram: From cache replacement to normally-off computing," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 1, pp. 49–60, Jan 2016.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[35] W. Tang, G. Hua, and L. Wang, "How to train a compact binary neural network with high accuracy?" in *AAAI*, 2017, pp. 2625–2631.

[36] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[37] M. Courbariaux and Y. Bengio, "BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1," *CoRR*, vol. abs/1602.02830, 2016. [Online]. Available: http://arxiv.org/abs/1602.02830

[38] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 994–1007, July 2012.

[39] S. Sheu, M. Chang, K. Lin, C. Wu, Y. Chen, P. Chiu, C. Kuo, Y. Yang, P. Chiang, W. Lin, C. Lin, H. Lee, P. Gu, S. Wang, F. T. Chen, K. Su, C. Lien, K. Cheng, H. Wu, T. Ku, M. Kao, and M. Tsai, "A 4mb embedded slc resistive-ram macro with 7.2ns read-write random-access time and 160ns mlc-access capability," in *2011 IEEE International Solid-State Circuits Conference*, Feb 2011, pp. 200–202.

[40] A. Boroumand, S. Ghose, M. Patel, H. Hassan, B. Lucia, K. Hsieh, K. T. Malladi, H. Zheng, and O. Mutlu, "Lazypim: An efficient cache coherence mechanism for processing-in-memory," *IEEE Computer Architecture Letters*, vol. 16, no. 1, pp. 46–50, 2017.

[41] S. Ghose, K. Hsieh, A. Boroumand, R. Ausavarungnirun, and O. Mutlu, "Enabling the adoption of processing-in-memory: Challenges, mechanisms, future research directions," *arXiv preprint arXiv:1802.00320*, 2018.

[42] L. Han, Z. Shen, Z. Shao, H. H. Huang, and T. Li, "A novel reram-based processing-in-memory architecture for graph computing," in *2017 IEEE 6th Non-Volatile Memory Systems and Applications Symposium (NVMSA)*. IEEE, 2017, pp. 1–6.

[43] S. Xu, X. Chen, Y. Wang, Y. Han, X. Qian, and X. Li, "Pimsim: A flexible and detailed processing-in-memory simulator," *IEEE Computer Architecture Letters*, vol. 18, no. 1, pp. 6–9, 2019.

Ashkan Samiee received the B.S. degree in electronic engineering from Polytechnic University, Tehran, Iran, in 2014, and the M.S. degree in electrical engineering from Texas A&M University, Texas, USA, in 2016. He is currently working on his second master in Computer engineering in California State University, Fullerton.

Payal Borulkar received the M.Sc degree in Computer Engineering from the California State University, Fullerton in 2018.

Ronald F. DeMara (S87-M93-SM04) received the Ph.D. degree in Computer Engineering from the University of Southern California in 1992. Since 1993, he has been a full-time faculty member at the University of Central Florida where he is a Professor of Electrical and Computer Engineering, and joint faculty of Computer Science, and has served as Associate Chair, ECE Graduate Coordinator, and Computer Engineering Program Coordinator. His research interests are in computer architecture with emphasis on reconfigurable logic devices, evolvable hardware, and emerging devices, on which he has published over 275 articles and holds one patent. He is a Senior Member of IEEE and has served on the Editorial Boards of IEEE Transactions on Computers, IEEE Transactions on VLSI Systems, and IEEE Transactions on Emerging Topics in Computing. He received IEEEs Joseph M. Bidenbach Outstanding Engineering Educator Award in 2008.

Peiyi Zhao received a B.S. in Information and Electronics from Zhejiang University in 1987 and he obtained Ph.D in Computer Engineering from the Center for Advanced Computer Studies of University of Louisiana, Lafayette, 2005. He was an assistant professor in Chapman University, Orange, CA, USA, from 2005-2012. He is currently an associate professor in Integrated Circuit and Embedded Systems Lab, Fowler School of Engineering in Chapman University.

Yu Bai is an Assistant Professor in the Computer Engineering Program at the California State University, Fullerton. He earned his Ph.D. degree in Electrical Engineering from the University of Central Florida in 2016. Yu received his BS degree in 2008 in Electrical Engineering from the Ukraine National Aviation University and his M.S. degree in 2011 in Electrical and Computer Engineering from the University of Texas-Pan American. His research interests include stochastic computing, neuromorphic computing, FPGA design, a nano-scale computing system with novel silicon and post-silicon devices, and low power digital and mixed-signal CMOS circuit design.