Received October 15, 2021, accepted October 28, 2021, date of publication November 8, 2021, date of current version November 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3125685

STT-BSNN: An In-Memory Deep Binary Spiking Neural Network Based on STT-MRAM

VAN-TINH NGUYEN[1], **(Graduate Student Member, IEEE),**
QUANG-KIEN TRINH[2], **(Member, IEEE), RENYUAN ZHANG**[1], **(Senior Member, IEEE),**
AND YASUHIKO NAKASHIMA[1], **(Senior Member, IEEE)**
[1]Graduate School of Information Science, Nara Institute of Science and Technology (NAIST), Ikoma 630-0192, Japan
[2]Radio Electronics Faculty, Le Quy Don Technical University, Hanoi 143330, Vietnam

Corresponding author: Quang-Kien Trinh (kien.trinh@lqdtu.edu.vn)

This work was supported by the Vietnam National Foundation for Science and Technology Development (NAFOSTED) under Grant 102.01-2018.310.

ABSTRACT This paper proposes an in-memory binary spiking neural network (BSNN) based on spin-transfer-torque magnetoresistive RAM (STT-MRAM). We propose residual BSNN learning using a surrogate gradient that shortens the time steps in the BSNN while maintaining sufficient accuracy. At the circuit level, presynaptic spikes are fed to memory units through differential bit lines (BLs), while binarized weights are stored in a subarray of nonvolatile STT-MRAM. When the common inputs are fed through BLs, vector-to-matrix multiplication can be performed in a single memory sensing phase, hence achieving massive parallelism with low power and low latency. We further introduce the concept of a dynamic threshold to reduce the implementation complexity of synapses and neuron circuitry. This adjustable threshold also permits a nonlinear batch normalization (BN) function to be incorporated into the integrate-and-fire (IF) neuron circuit. The circuitry greatly improves the overall performance and enables high regularity in circuit layouts. Our proposed netlist circuits are built on a 65-nm CMOS with a fitted magnetic tunnel junction (MTJ) model for performance evaluation. The hardware/software co-simulation results indicate that the proposed design can deliver a performance of 176.6 TOPS/W for an in-memory computing (IMC) subarray size of 1×288. The classification accuracy reaches 97.92% (83.85%) on the MNIST (CIFAR-10) dataset. The impacts of the device non-idealities and process variations are also thoroughly covered in the analysis.

INDEX TERMS Binary spiking neural network, emerging memory technology, in-memory computing, neuromorphic computing, process variation, STT-MRAM.

I. INTRODUCTION

Deep neural networks (DNNs) in on-edge AI devices face the challenge of high energy consumption due to the requirement of a large number of tensor operations, which incurs not only a high computational workload but also large memory accesses [1]–[5]. The conventional computer architecture with limited memory bandwidth and a sequential computing framework is not ideal for such operations, especially for DNNs in on-edge AI devices such as the Internet of Things (IoT) or mobile systems, which have strict resource and power budget constraints. IMC has been recently introduced as a revolutionary approach to solving the memory bottleneck challenge [6]. This approach partially shifts the processing load from the central processing unit to distributed processing elements in memory, which greatly reduces memory access while increasing performance and energy efficiency. IMC finds it difficult to support complex processing operations such as multiply-accumulate (MAC); therefore, the binary neural network (BNN) has recently emerged [7], with the aim of simplifying network operations. Interestingly, IMC approaches and BNNs apparently exhibit much synergy. Indeed, a BNN typically performs calculations based on bitwise XNOR operations (for multiplication), and bit counting (for accumulation) can be fully implemented in memory, as proposed in some prior works. The proposed IMC designs in [8], [9] employ resistive RAM (RRAM), yielding low standby power and improving array density. However,

The associate editor coordinating the review of this manuscript and approving it for publication was Yuh-Shyan Hwang.

in these works, the accumulation process based on the current summing approach could require significant energy in large networks. As a further development, STT-MRAM was also adopted for IMC-based neural networks. STT-MRAM exhibits outstanding advantages over other types of memory in terms of endurance [10]; thus, it is well suited for use as on-chip memory. The STT-MRAM-based BNN accelerator architecture in [11] also adopts the current sensing approach and is only reasonable for small arrays (8×4). Furthermore, in [12], the authors introduced a scalable and fully parallel in-memory BNN structure, which supports MAC operations in a single memory access phase via a voltage sensing technique and achieves improved scalability and energy efficiency. Nonetheless, the accuracy of BNNs strongly depends on process variations, which could be quite severe in finer technologies.

Spiking neural networks (SNNs), known as the third generation of DNNs, not only better mimic biological neural behaviors but also exhibit great fault tolerance and can potentially overcome the persistent drawback of binarized networks [13]. Therefore, SNNs have been actively studied recently. Among them, an MTJ was also adopted for SNNs [14]–[19]. In [14]–[16], the authors leveraged the binary switching of an MTJ to map the sigmoid function in an artificial neural network (ANN) to an SNN. However, this mapping may cause significant accuracy degradation. Additionally, the accuracy suffers from unstable MTJ switching and bias current variation. Inhibitory and excitatory spike-timing-dependent plasticity was processed for on-chip learning with MTJ-synaptic [17] and MTJ-neuron [18] implementations. In [17], a stochastic synapse was introduced, in which synaptic propagation was modulated stochastically by a full-precision weight. Then, each neuron accumulated incoming synaptic events sequentially using a spike counter, which significantly affected the network latency. In [18], the leaky-integrate-fire spiking model was used to emulate biological neuron dynamics, but this work focused only on neuron circuitry and did not cover the impact of variation. On the other hand, in [19], the author presented a compact probabilistic Poisson method based on back-hopping oscillation in an MTJ, where the number of spikes was exponentially proportional to the synaptic current in the utilized sampling time (within a time step). However, the classification accuracy of this approach is highly sensitive to the sampling period.

In this work, we propose a complete design approach with multilevel optimization for a BSNN that leverages high-energy and efficient IMC based on STT-MRAM. For network training, we first propose a direct BSNN training approach using a surrogate gradient augmented with residual connections. While directly training a deep SNN with binary weights can cause accuracy degradation, as reported in [20], our training method improves the accuracy and reduces the network latency. In the BSNN inference process, we also provide a mathematical formulation to justify that the MAC function of the BSNN can fully utilize in-memory with XNOR-based computation. Specifically, we propose an equivalent IF model with a dynamic threshold instead of a fixed threshold. This permits the MAC operation–the most computationally intensive operation of the synapse – to be realized solely in the XNOR IMC array. Additionally, the fundamental operation of the IF neuron is based only on an accumulation function. The latter not only simplifies the neuron circuitry but also enhances its calculation accuracy. Additionally, considering that the threshold is dynamic, we adjust the initial value (i.e., after firing) to emulate the impact of BN.

Our contributions can be summarized as follows

- We propose a BSNN with residual connections and train the network with a surrogate gradient, which enables higher classification accuracy with fewer time steps.
- The proposed dynamic threshold mechanism allows neural synapses to be mapped to XNOR cells based on the STT-MRAM subarray in [12] and reduces the complexity of the neuron circuit by incorporating BN.
- The proposed approaches are built for circuit-level simulations. The accuracy and other performance metrics of the network are then evaluated based on parameters realistically extracted from circuit simulations, including the nonlinearity and process variations.

The rest of this paper is organized as follows. Section II presents the BSNN training model. The BSNN IMC model with a novel MAC operation using an XNOR cell circuit is introduced in Section III. Section IV expresses the models of the STT-BSNN array and introduces the sense amplifier used in the proposed architecture. Section V evaluates the accuracy of the proposed training method, and the energy efficiency of our design is compared with that of previous studies. Then, Section VI concludes this work.

II. TRAINING A BSNN WITH A SURROGATE GRADIENT AND RESIDUAL CONNECTIONS

In this section, we present the training process for BSNNs using surrogate gradient backpropagation with residual connections. In our training model, the binarized weights are represented in bipolar format (i.e., ± 1), as introduced in [21].

A. BACKGROUND

In the conventional IF model, the membrane potential $u_i^{t,l}$ for the i-th neuron at time step t in layer l is defined as follows

$$u_i^{t,l} = u_i^{t-1,l} + \frac{\gamma}{\sigma}\left(\sum_{j=1}^{M} w_{ij}^{l} \cdot s_j^{t,l-1} - \mu\right) + \beta \qquad (1)$$

Here, γ and β are scaling and shifting parameters, respectively; σ and μ correspond to the standard deviation and mean of BN, respectively. M denotes the number of presynaptic spikes, and $s_j^{t,l-1}$ is the presynaptic spike in the j-th neuron. $w_{ij}^{l} = \alpha \cdot w_{ij}^{b,l}$ represents the latent weight of the BSNN, where $w_{ij}^{b,l} = sign(w_{ij}^{l})$ is the corresponding binary weight and $\alpha = \left|w_{ij}^{l}\right|$ is the latent weight scaling factor.

For the spike representation, we use the rate encoding method introduced in [22]. Specifically, the real input data are converted into spike format using a Poisson random number generator. The generated value is proportional to the total number of spikes within T time steps. According to the IF model in (1), if the membrane potential $u_i^{t,l}$ surpasses the firing threshold θ_i^l, a postsynaptic spike $o_i^{t,l}$ is generated. Then, the membrane potential is reset to zero before being accumulated again. Furthermore, the cross-entropy loss function is calculated through the last output membrane potential $u_i^{T,L}$, which is expressed as follows

$$\mathcal{L}_p = -\sum_{i=1}^{C} y_i log\left(\frac{e^{u_i^{T,L}}}{\sum_{k=1}^{C} e^{u_k^{T,L}}}\right) \tag{2}$$

where $Y = (y_1, y_2, \ldots, y_C)$ is a label vector and C is the total number of network outputs.

During the training process, the loss function $\mathcal{L}_p$ is minimized by gradient descent, and the latent weight is updated as follows [22]

$$w_{ij}^l = w_{ij}^l - \eta \cdot \sum_t \frac{\partial \mathcal{L}_p}{\partial w_{ij}^{t,l}} \tag{3}$$

where η is a learning rate. $\sum_t \frac{\partial \mathcal{L}_p}{\partial w_{ij}^{t,l}}$ is the accumulated gradient over all time steps, which is calculated as in [22]:

$$\sum_t \frac{\partial \mathcal{L}_p}{\partial w_{ij}^{t,l}} = \begin{cases} \sum_t \frac{\partial \mathcal{L}_p}{\partial o_i^{t,l}} \frac{\partial o_i^{t,l}}{\partial u_i^{t,l}} \frac{\partial u_i^{t,l}}{\partial w_{ij}^{t,l}}, & if\ 1 \le l < L \\ \sum_t \frac{\partial \mathcal{L}_p}{\partial u_i^{T,L}} \frac{\partial u_i^{T,L}}{\partial w_{ij}^{t,L}}. & if\ l = L \end{cases} \tag{4}$$

In (4), the gradient calculation suffers from non-differentiable spiking activities. To address this issue, an approximate gradient (i.e., a surrogate gradient) was introduced in [22], [23], which is formally expressed as

$$\frac{\partial o_i^{t,l}}{\partial u_i^{t,l}} = \delta \cdot max\left\{0, 1 - \left|\frac{u_i^{t,l} - \theta_i^l}{\theta_i^l}\right|\right\} \tag{5}$$

where δ typically is set to 0.3. The surrogate gradient is effective for solving non-differentiable spiking activity. However, when the network gets deeper, the training process based on gradient descent in (3)-(4) suffers from the degradation problem [20], [24]. In the following, we present how a residual connection [20], [24] can be adopted for our BSNN to tackle the degradation issue.

B. PROPOSED BSNN WITH RESIDUAL CONNECTIONS

Residual connection is an effective technique that helps to stabilize the training processes and improve the classification accuracies of deep networks [20], [24]. We hence propose a BSNN training model using a surrogate gradient in conjunction with residual connections. As illustrated in Fig. 1, relative to the conventional BSNN network in Fig. 1(a), each convolutional (Conv) layer in the residual structure in Fig. 1(b) has an additional connection from layer $(l-1)$ to layer l via the inverter-AND spike-element-wise (SEW) function g.

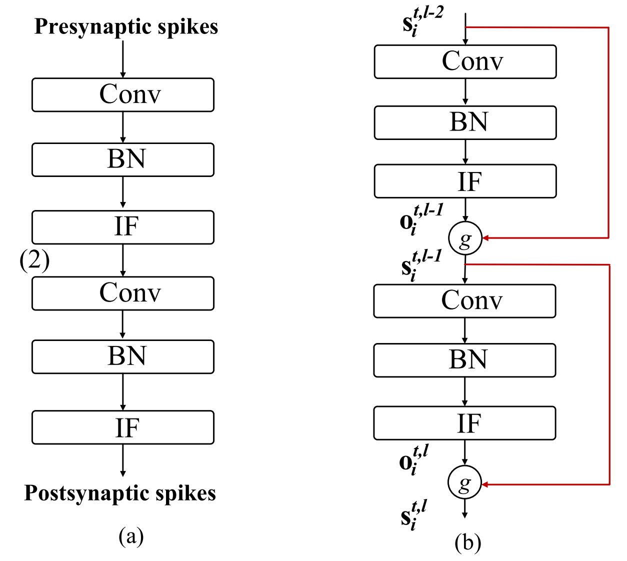


FIGURE 1. (a) A conventional BSNN topology, (b) a BSNN topology with residual connections using an inverter-AND spike-element-wise function.

The spike $s_i^{t,l}$ of layer l is now dependent on the IF output $o_i^{t,l}$ and the spike $s_i^{t,l-1}$ as follows

$$s_i^{t,l} = g\left(o_i^{t,l}, s_i^{t,l-1}\right) = (1 - o_i^{t,l}) \bigwedge s_i^{t,l-1} \tag{6}$$

By supporting the SEW function in (6), if the output in (1) is $o_i^{t,l} = 0$, the output of the element-wise function becomes $s_i^{t,l} = s_i^{t,l-1}$, which satisfies the identity mapping condition. The algorithm that describes the whole training process can be found in Appendix A.

III. XNOR-BASED BSNN INFERENCE WITH A DYNAMIC THRESHOLD

This section presents a BSNN inference model that utilizes the MAC operation based on an XNOR array; the latter is suited for implementation in memory using emerging technologies [12], [25]. To simplify the inference model in (1) without accuracy degradation, we set $\gamma = 1$ and $\beta = 0$ in the BN layers [26]. The original IF model for a BSNN is expressed as

$$\begin{aligned} &Integration: u_i^{t,l} = u_i^{t-1,l} + \frac{\alpha}{\sigma}\left\{\sum_{j=1}^{M} w_{ij}^{b,l} s_j^{t,l-1} - \frac{\mu}{\alpha}\right\} \\ &Firing: o_i^{t,l} = \begin{cases} 1, & if\ u_i^{t,l} > \theta_i^l \\ 0, & otherwise \end{cases} \\ &Resetting: u_i^{t,l} = 0 \end{aligned} \tag{7}$$

In this model, during every time step, the membrane potential $u_i^{t,l}$ accumulates with $\alpha/\sigma \left\{\sum_{j=1}^{M} w_{ij}^{b,l} s_j^{t,l-1} - \frac{\mu}{\alpha}\right\}$ and then is compared with a threshold θ_i^l for the firing decision. To avoid multiplication in (7), which could be costly to implement at the circuit level, we scale both the membrane potential and the threshold by a factor of α/σ.

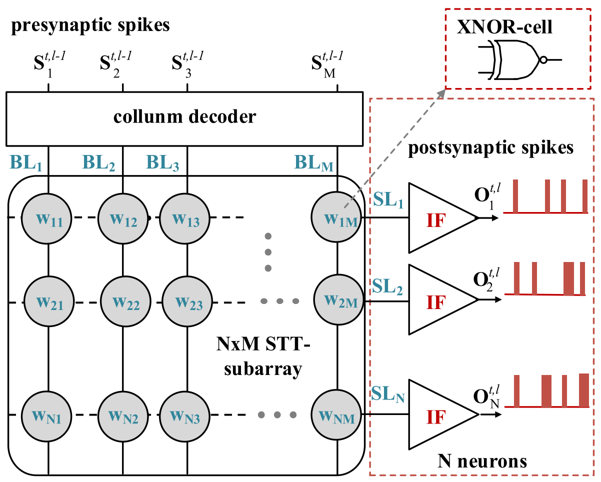


FIGURE 2. BSNN architecture for intra-layer processing using an XNOR cell array.

Given that, the scaled membrane potential $\hat{u}_i^{t,l}$ and the threshold $\hat{\theta}_i^l$ can be rewritten as

$$\hat{u}_i^{t,l} = \hat{u}_i^{t-1,l} + \sum_{j=1}^{M} w_{ij}^{b,l} s_j^{t,l-1} - \frac{\mu}{\alpha}$$
$$\hat{\theta}_i^l = \frac{\sigma}{\alpha} \cdot \theta_i^l \tag{8}$$

The MAC component $\sum_{j=1}^{M} w_{ij}^{b,l} s_j^{t,l-1}$ in (8) is essentially the most computationally extensive operation. On the other hand, in the binary spiking model, a spike signal is represented in unipolar format (0, 1) [27], [28], while the weights $w_{ij}^{b,l}$ are trained with a bipolar format $(-1, 1)$. Therefore, unlike in the case of a BNN [12], [25], it is not possible to directly utilize an XNOR array for a BSNN MAC function. To overcome this issue, assuming that the M weights in (8) have M_1 negative weights (w_{ij}^{-b}) and $M - M_1$ positive weights (w_{ij}^{+b}), the MAC function in (8) can be reformulated as

$$\sum_{j=1}^{M} w_{ij}^{b,l} s_j^{t,l-1} = \sum_{j=1}^{M_1} \overline{(1 + w_{ij}^{-b}) \oplus s_j^{t,l-1}} - M_1$$
$$+ \sum_{j=1}^{M-M_1} \overline{w_{ij}^{+b} \oplus s_j^{t,l-1}} = \sum_{j=1}^{M} \overline{w_{ij}^{u,l} \oplus s_j^{t,l-1}} - M_1 \tag{9}$$

In (9), $w_{ij}^{u,l}$ is represented in unipolar format ($w_{ij}^{u,l} = (w_{ij}^{b,l} + 1)/2$). Substituting the expression of (9) into (8) and denoting $\rho = M_1 + \frac{\mu}{\alpha}$, the scaled membrane potential in (8) is now expressed as

$$\hat{u}_i^{t,l} = \hat{u}_i^{t-1,l} + \sum_{j=1}^{M} \overline{w_{ij}^{u,l} \oplus s_j^{t,l-1}} - \rho \tag{10}$$

In (10), the component $\sum_{j=1}^{M} \overline{w_{ij}^{u,l} \oplus s_j^{t,l-1}}$ can be realized entirely in memory by using an XNOR array.

However, this transformation introduces a constant ρ in (10) implies that the hardware implementation must support both accumulation and subtraction. The latter leads to undesirable complexity in the circuit implementation. To solve this problem, we propose an equivalent IF model with a dynamic threshold mechanism. Specifically, instead of a fixed threshold, we can transform the negative component ρ of the scaled membrane in (10) into a positive component of the threshold $\hat{\theta}_i^l$, which is now considered the time-dependent quantity $\hat{\theta}_{dyn,i}^{t,l}$. Therefore, the proposed IF model can be formulated as follows

$$\begin{aligned}
&\textit{Integration}: \begin{cases} \hat{u}_{xnor,i}^{t,l} = \hat{u}_{xnor,i}^{t-1,l} + \sum_{j=1}^{M} \overline{w_{ij}^{u,l} \oplus s_j^{t,l-1}} \\ \hat{\theta}_{dyn,i}^{t,l} = \hat{\theta}_{dyn,i}^{t-1,l} + \rho, \hat{\theta}_{dyn,i}^{t=0,l} = \hat{\theta}_i^l \end{cases} \\
&\textit{Firing}: o_i^{t,l} = \begin{cases} 1, & \textit{if } \hat{u}_{xnor,i}^{t,l} > \hat{\theta}_{dyn,i}^{t,l} \\ 0, & \textit{otherwise} \end{cases} \\
&\textit{Resetting}: \hat{u}_{xnor,i}^{t,l} = 0 \textit{ and } \hat{\theta}_{dyn,i}^{t,l} = \hat{\theta}_i^l
\end{aligned} \tag{11}$$

Compared to the IF model in (7), the IF model in (11) requires only accumulation operations. This means that the latter model can help to simplify the subsequent hardware implementation.

However, the model in (11) is correct only when ρ is positive. Although it rarely occurs, the value of ρ can theoretically be negative, i.e., the subtraction in (10) is actually an addition. In that case, we retain the original expression in (10) and keep the threshold constant. Accordingly, a small modification is required in the IF neuron circuit implementation (detailed in Section IV) for covering both positive and negative values of ρ.

The pseudocode of the XNOR-based BSNN inference process with a dynamic threshold is presented in Appendix B.

IV. IN-MEMORY STT-MRAM BITCELL AND SUBARRAY CIRCUITS

This section presents the circuit implementation for the BSNN model presented in Section II and Section III. The general architecture for intra-layer processing using the proposed model is shown in Fig. 2. First, the binarized weight $w_{ij}^{u,l} \equiv w_{ij}$ is mapped into the memory of the $N \times M$ STT subarray. Then, the digital presynaptic spikes $s_j^{t,l-1}$ are encoded by the column decoder and fed to the array through bit line BL$_j$ ($j = 1 \div M$) to fit the XNOR-MAC computation. Finally, the source line (SL) SL$_i$ ($i = 1 \div N$) voltage, which represents the output of the MAC operation, is passed into the IF model in (11) to generate postsynaptic spikes $o_i^{t,l}$.

All of the circuit implementation and simulation steps are completed on a 65-nm CMOS with the MTJ parameters presented in Table 1.

The detailed implementations of individual subcircuits are subsequently described in the following sections.

TABLE 1. BSNN circuit parameters.

MTJ size (W×L)	60 nm × 60 nm
MTJ thickness (Tm)	1.5 nm
oxide thickness (TMgO)	1.15 nm
MTJ resistance variability	5%
nominal R_P (R_{AP})	2 KΩ (4 KΩ)
tunnel magnetoresistance (TMR)	100%
max MTJ read current (1E-9 bitflip rate)	50 µA
65-nm CMOS ($W_{min} \times L_{min}$)	135 nm × 60 nm

A. MAC OPERATIONS USING A COMPLEMENTARY 2T-2R STT-MRAM BITCELL AND SUBARRAY

To map an MAC computational unit into the IMC memory, we employ a 2T-2R STT-MRAM-based XNOR cell. The details of this circuit and model can be found in [12]; here, we give brief descriptions of the bitcell and row operations. Updating the binary weights is performed at the beginning. Since SL is shared among the cells in the same row, each MTJ has to be written individually. Specifically, apart from the BL corresponding to the active MTJ, other BLs are left in the high-impedance state while an appropriate voltage is applied across (BL, SL) for flipping the MTJ magnetization. The write peripheral circuit is omitted for clarity, details can be found in [29], [30]. As seen in Fig. 3(a), for a single XNOR bitcell, the binarized weight (0, 1) is encoded by the MTJ states (R_{AP} and R_P), and the presynaptic spike is encoded to a pair of BL voltages as follows

$$s_j^{t,l-1} \equiv \begin{cases} 0, & if\ BL_{0,j}^{t,l-1} = 0\,(V)\,, BL_{1,j}^{t,l-1} = V_{BL} \\ 1, & if\ BL_{0,j}^{t,l-1} = V_{BL}\,, BL_{1,j}^{t,l-1} = 0\,(V) \end{cases} \tag{12}$$

Here, V_{BL} is set to 0.3 V to limit the bitcell reference current to less than 50 μA and avoid a high disturbance rate and high power consumption [31]. The BL driver encodes incoming spikes using a pair of complementary transistors (an n-channel MOS (NMOS) and a p-channel MOS (PMOS)). The SL voltage represents the output of a single XNOR operation (see Fig. 3(b)). Fig. 3(a) shows the circuit implementation for a single-row XNOR-based BSNN using STT-MRAM. Each high-load WL is driven by a buffer (an 4-stage inverter chain) that guarantees the fast transition and stable level value during the MAC operation on the memory row. Additionally, from [12], the MAC product in the output of the i-th row connection is represented by the merged SL voltage $V_{SL,i}^{t,l}$ (i.e., all bitcells in a row share the same SL), which is equal to [12]

$$V_{SL,i}^{t,l} = V_{BL}\left(\frac{M-K}{M}\cdot\frac{R_{bitcell}}{R_{AP}+R_{access}}+\frac{K}{M}\cdot\frac{R_{bitcell}}{R_P+R_{access}}\right) \tag{13}$$

where $R_{bitcell}$ is the overall resistance seen from the SL terminal of the bitcell, which is data-independent and equal to $(R_{AP}+R_{access})\ ||\ (R_P+R_{access})$; K denotes the number of XNOR outputs (i.e., $s_j^{t,l-1} \oplus w_{ij}$) equal to $+1$ across the entire row of M bitcells. The SL voltage linearly depends on K and ranges from $V_{BL}\frac{R_{bitcell}}{R_{AP}+R_{access}}$ ($K = 0$) to $V_{BL}\frac{R_{bitcell}}{R_P+R_{access}}$ ($K = M$) [12].

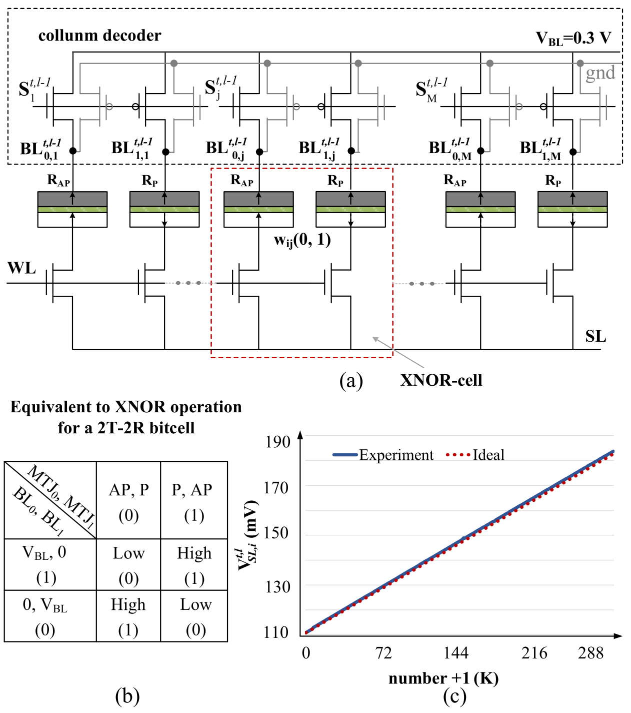


FIGURE 3. (a) A single STT-MRAM row based on 2T-2R STT-MRAM bitcells for realizing binarized MAC operations, (b) the SL voltage level corresponding to the XNOR operation for a single 2T-2R bitcell [12] and (c) the dependence of the SL voltage on the number of (+1) values among the XNOR outputs (K) of the circuit simulation for a row of 288 bitcells with V_{BL} = 0.3 V and V_{WL} = 0.9 V.

In Fig. 3(c), we plot the circuit simulation of $V_{SL,i}^{t,l}$ with respect to K. The simulation results show that $V_{SL,i}^{t,l}$, which ranges from $110mV$ to $184mV$, is linearly dependent on K. This confirms that the MAC calculation (11) can be performed within a single in-memory access phase.

In the following, we introduce an approach for implementing an IF neuron mechanism (the model in (11)) using circuit computation in the charge domain, whose input is the SL voltage $V_{SL,i}^{t,l}$ from the MAC operation.

B. IF NEURON AND SEW CIRCUIT DESIGNS

1) GENERAL ARCHITECTURE

Fig. 4 shows the proposed implementation of the IF neuron and SEW circuit architecture. The IF neuron circuit consists of two charge-based accumulations (ACC1 and ACC2), a capacitive voltage booster (CB), and firing and shaping (FS) circuits. The charge-based accumulations are used to update the membrane potential and the dynamic threshold. As discussed earlier, the result of an MAC operation is represented in the form of a voltage at the merged SL ($V_{SL,i}^{t,l}$). Subsequently, this voltage must be accumulated in every time step, followed by the IF model in (11). However, since $V_{SL,i}^{t,l}$ varies from $110mV$ to $184mV$, it must be amplified to an adequate level to limit the charging current.[1]

[1]If $V_{SL,i}^{t,l}$ is used directly to control the charging current (the drain current of the PMOS M_1), the accumulated charge on C_1 can quickly reach the saturation level (i.e., $V_{DD} \times C$, where C is the capacitance of C_1) because the M_1 transistor is almost at the full-driving state.

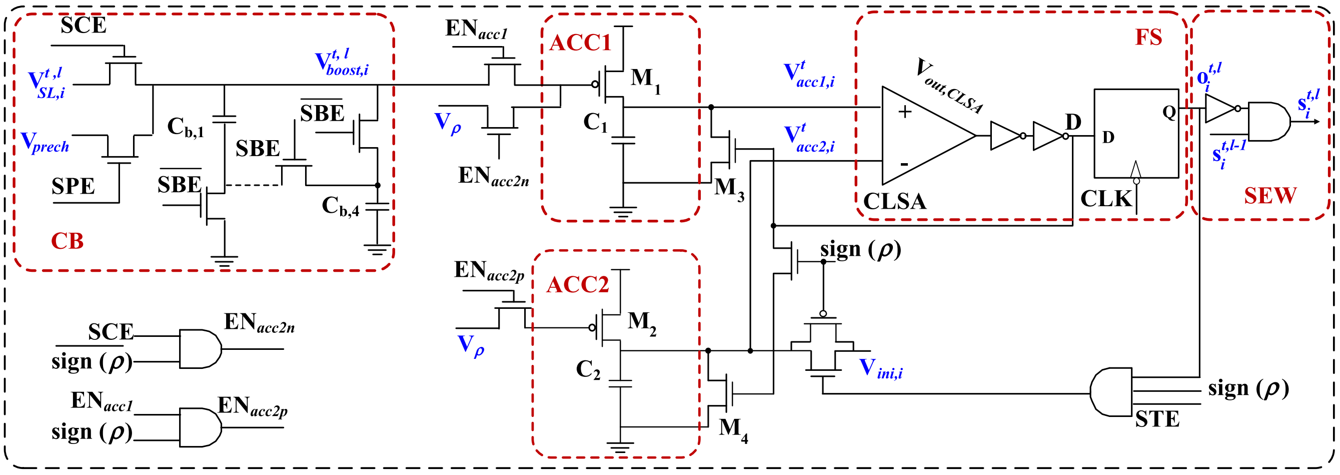


FIGURE 4. The IF neuron and SEW circuit of the proposed XNOR-based BSNN inference with STT-MRAM synapses.

The amplification process is performed by the CB circuit introduced in [31], considering the amplification factor (G) does not require high precision, and the CB circuit is very compact and energy efficient. As seen in Fig. 5, $V_{SL,i}^{t,l}$ is sampled before being fed into the booster circuit, the sampling time τ_{sample} for this array size is chosen to be 1 ns (when SCE $=$ 1). The capacitors in the CB are precharged by V_{prech} up to the middle level of $V_{SL,i}^{t,l}(150mV)$ to optimize the timing, as detailed in [31]. The precharging time $\tau_{precharge}$ is set to 0.5 ns (when SPE $=$ 1). After boosting, $V_{boost,i}^{t,l}$ is maintained for the duration of $\tau_{boost} = 4.5ns$ (when SBE $=$ 1). During that time, when the signal EN$_{acc1}$ is activated, $V_{boost,i}^{t,l}$ is connected to the gate of M$_1$ transistor for charging C$_1$ within a fixed duration of time $\tau_{charge} = 1ns$. The additional amount of charge in C$_1$ hence is proportional to $V_{boost,i}^{t,l}$ and to $V_{SL,i}^{t,l}$. The voltage level across C$_1$ is equal to $V_{acc1,i}^{t} = Q_{acc1,i}^{t} \big/ C$, which indirectly represents the accumulation of $V_{SL,i}^{t,l}$ at time step t.

Similarly, ACC2 is utilized for dynamic threshold accumulation. Assume that $\rho > 0$, and V_ρ represents the value of ρ in the voltage domain. As shown in Fig. 5, when EN$_{acc2p}$ is activated, V_ρ is used to charge C$_2$ through M$_2$(EN$_{acc2p} \equiv$ EN$_{acc1}$). The amount of additional charge corresponds to the voltage increment at the output $V_{acc2,i}^{t}$ of capacitor C$_2$. In such a way, $V_{acc2,i}^{t}$ represents the dynamic threshold accumulation corresponding to the model in (11).

Finally, $V_{acc1,i}^{t}$ and $V_{acc2,i}^{t}$ are fed into the current latched sense amplifier (CLSA) [32] circuit for a voltage level comparison. If the firing condition in (11) is satisfied ($V_{acc1,i}^{t} > V_{acc2,i}^{t}$), a spike is generated in the output of the CLSA ($V_{out,CLSA}$), as an example shows in Fig. 5. Subsequently, $V_{out,CLSA}$ is shaped by two inverters before being fed into a D-flip-flop (D-FF) for the postsynaptic generation of spike $o_i^{t,l}$. The frequency of the clock (CLK) signal determines the postsynaptic spike period ($T_{spike} = 6ns$). Additionally, after two inverters, a signal D, which is $V_{out,CLSA}$ delayed by two inverters, is used for resetting ACC1 and ACC2 before starting the next step operation.

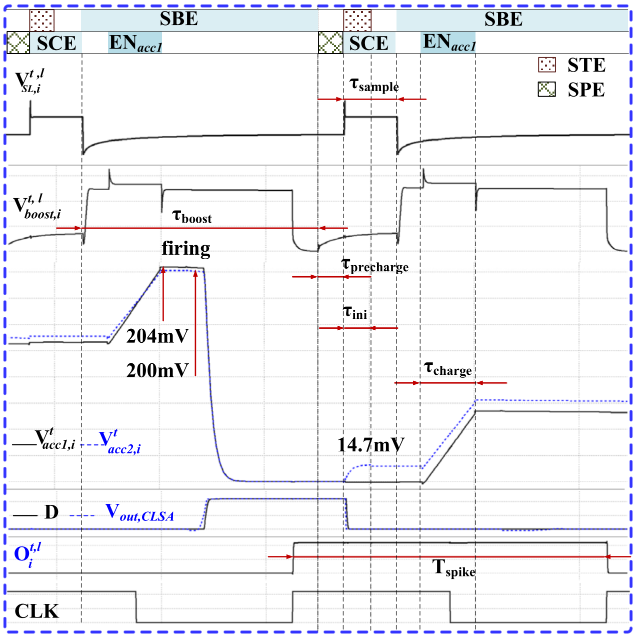


FIGURE 5. The IF neuron circuit simulation waveform within 2 time steps for a row with 288 bitcells.

To reset the dynamic threshold (after firing and during initialization), C$_2$ is precharged by $V_{ini,i}$, which represents the threshold $\hat{\theta}_i^l$ (see (11)) in the voltage domain. According to Fig. 4, that C$_2$ is precharged when both the STE and output of the D-FF are equal to 1, where STE is a periodic signal that enables the threshold precharging circuit for a fixed duration within a time step. In the normal working mode, the STE is activated when a generated postsynaptic spike occurs. For example, $V_{acc2,i}^{t}$ is precharged to $V_{ini,i}$ ($14.7mV$) within the duration τ_{ini} ($0.5ns$) after firing, as seen in Fig. 5. During the initialization process, the D-FF output is manually set to '1' to preset the threshold. After IF processing, the output of the D-FF is fed into the SEW circuit to perform residual connection according to the model in (6). Specifically, $o_i^{t,l}$ is added with the spike from the previous layer $s_i^{t,l-1}$ using a single inverter-AND gate, as shown in Fig. 4.

Note that in some rare cases, if $\rho < 0$ ($sign(\rho) \equiv 0$) (see (11)), EN$_{acc2p}$ is deactivated, and EN$_{acc2n}$ is active. In such cases, V_ρ accumulates in ACC1 instead of ACC2. In other words, the output of ACC2 is fixed as $V_{ini,i}$. This switching mechanism recalls our discussion about ρ in Section III.

The detailed charge-based accumulation, analysis and circuit implementation of the core functions are described below.

2) CHARGE-BASED ACCUMULATION

As shown in Fig. 4, the charge-based accumulation architecture consists of a PMOS transistor and a capacitor. According to the well-known α-power law model [33], the charging current I_{ds}^t passing through transistors M$_1$ and M$_2$ is equal to

$$I_{ds}^t = \frac{\mu C_{ox} W}{L} (V_{GS} - V_{TH})^\alpha \tag{14}$$

where α is the model power index, which ranges from 1-2 depending on the adopted technology. In our chosen technology (a 65-nm CMOS), the transistor is considered a short channel; i.e., theoretically, $\alpha \approx 1$.

We further experimentally verify that the charging current is quasi-linear and dependent on V_{GS} in the range of interest (V_{GS} from $-702mV$ to $-443mV$). This fact is very important and allows the proposed model in (11) to be directly mapped to the circuit solution. Note that the value of $V_{SL,i}^{t,l}$ and its boosted value $V_{boost,i}^{t,l} = G{\cdot}V_{SL,i}^{t,l}$ ($G \approx 3.6$) are constants within a time step duration. The charging current I_{ds}^t in (14) can therefore be considered unchanged during the accumulation time. Thus, the amount of charge accumulated in C$_1$ in the time step from $t-1$ to t is equal to

$$\begin{aligned} \Delta Q_{acc1,i} &\approx \int_{t-1}^{t} I_{ds}^t d_t \\ &= \frac{\mu C_{ox} W \tau_{charge}}{L} \left(G V_{SL,i}^{t,l} - V_{DD} - V_{TH} \right) \\ &= \beta V_{SL,i}^{t,l} + \varphi \end{aligned} \tag{15}$$

Here, $\beta = G\frac{\mu C_{ox} W \tau_{charge}}{L}$ and $\varphi = -\frac{\mu C_{ox} W \tau_{charge}}{L}(V_{DD} + V_{TH})$. Accordingly, the amount of charge in C$_1$ at time step $t$equals to

$$Q_{acc1,i}^t \approx Q_{acc1,i}^{t-1} + \Delta Q_{acc1,i} \tag{16}$$

By replacing $V_{acc1,i}^t = Q_{acc1,i}^t \big/ C$ and $\Delta V_{acc1,i}^t = \Delta Q_{acc1,i} \big/ C$ (in (16)), the accumulated voltage at the second plate of the capacitor equals

$$V_{acc1,i}^t \approx V_{acc1,i}^{t-1} + \Delta V_{acc1,i}^t \tag{17}$$

Note that $\Delta V_{acc1,i}^t$ represents the voltage increment in ACC1 when $V_{SL,i}^{t,l}$ is applied to the input analog accumulation. For a dynamic threshold, the circuit implementation is almost the same, but $V_{SL,i}^{t,l}$ is replaced with V_ρ in (15)-(17). We have

$$V_{acc2,i}^t \approx V_{acc2,i}^{t-1} + \frac{\beta}{C} V_\rho + \frac{1}{C}\varphi \tag{18}$$

The dynamic threshold can be reformulated as follows

$$V_{acc2,i}^t \approx V_{acc2,i}^{t-1} + \Delta V_{acc2,i}^t \tag{19}$$

where $\Delta V_{acc2,i}^t = \frac{\beta}{C} V_\rho + \frac{1}{C}\varphi$. The equations for $V_{acc1,i}^t$ and $V_{acc2,i}^t$ in (17) and (19), respectively, in the circuit domain hence can be mapped to the model in (11).

3) THE EFFECTS OF NONLINEARITY AND PROCESS VARIATION

As introduced in (17) and (19), the IF model in (11) can be directly mapped to the charge-based accumulation circuit, which is introduced in Fig. 4. However, this model suffers from inevitable nonlinearity, and process variation comes from both the CMOS and MTJ devices. These effects essentially degrade the IF accuracy, as well as the overall system performance. In this section, we quantify this non-ideal impact based on actual circuit simulations, aiming to have a realistic evaluation of the proposed BSNN model at the system level in the subsequent section.

To capture the effect of process variation, we run Monte Carlo simulations for a synaptic array (a row of STT-MRAM) in Fig. 3(a) and the proposed IF neuron circuit in Fig. 4. The variation in the CMOS device is set according to the provided by foundry models. For the MTJ, the variability in the MTJ resistance is approximately 5% according to [34]. The accumulated output $\Delta V_{acc1,i}^t$ depends on K (the number of XNOR outputs that are equal to $+1$), as presented in Section IV.A. Each IMC row is designed for 288 bitcells that later fit with the BSNN inference model. Monte Carlo simulations have been performed for 289 cases of K. Fig. 6 plots the results of mapping K to $\Delta V_{acc1,i}^t(K)$ under nonlinear and variation effects. The capacitances C in ACC1 and ACC2 are set to 150 fF and 100 fF for all capacitors in the booster circuit. The PMOS sizes in ACC1 and ACC2 are set to $W = 8 \times W_{min}$, $L = 7 \times L_{min}$. The NMOS size in the array is set to $W = 4 \times W_{min}$, $L = L_{min}$.

From Fig. 6, we can see the effect of nonlinearity on $\Delta V_{acc1,i}^t(K)$ (the solid line). As shown in Fig. 6, $\Delta V_{acc1,i}^t(K)$ is quasi-linearly dependent on K. From the simulation data, the difference $\delta_1(K) = \Delta V_{acc1,i}^t(K) - \Delta V_1^t(K)$ between the ideal linear value $\Delta V_1^t(K)$ (the dashed line) and the simulated $\Delta V_{acc1,i}^t(K)$ is negligible in the middle but becomes significant near the boundary. For example, $\delta_{1,max}(0) = 7.27mV$, and $\delta_{1,min}(144) = 0.01mV$.

In Fig. 6, we also present the statistical analysis of $\Delta V_{acc1,i}^t(K)$. The statistical results show that the variations of $\Delta V_{acc1,i}^t(K)$ can be approximately fit to Gaussian distributions with a standard deviation $\sigma_1(K)$.

The results indicate that $\sigma_1(K)$ is not the same for every K. This effect is understandable from the circuit perspective, where the working points of the M$_1$ transistor for different values of K are not the same. For example, $\sigma_1(K)$ reaches

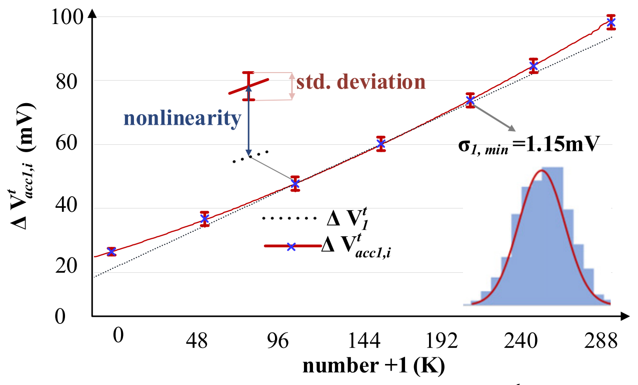


FIGURE 6. The effects of nonlinearity and variations on $\Delta V^t_{acc1,i}(K)$ with nonlinear errors $\delta_1(K)$ and standard deviations $\sigma_1(K)$ with respect to the number of XNOR outputs (equal to +1 (K) in a row with 288 bitcells).

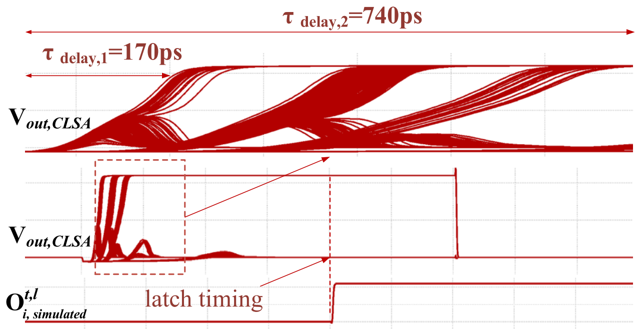


FIGURE 7. The effect of process variations on the output of the CLSA within a single time step.

its maximum value at 3.60 mV for $K = 2$ and its minimum value at 1.15 mV for $K = 221$.

The noise profile, represented by a normal distribution $n\{0, \sigma_1(K)\}$ is added to the ideal value $\Delta V^t_{acc1,i}$. The actual voltage level in the output of ACC1 can be approximated as

$$\Delta V^t_{acc1,i} = \Delta V^t_1(K) + \delta_1(K) + n\{0, \sigma_1(K)\} \tag{20}$$

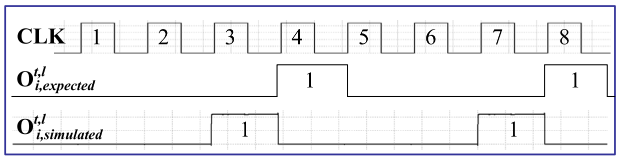
A similar analysis and model are conducted for $\Delta V^t_{acc2,i}$. The difference is that the variation $\Delta V^t_{acc2,i}$ comes from the ACC2 circuit itself without contributions from the synapse circuits (i.e., the memory array). We have

$$\Delta V^t_{acc2,i} = \Delta V^t_2(\rho) + \delta_2(\rho) + n\{0, \sigma_2(\rho)\} \tag{21}$$

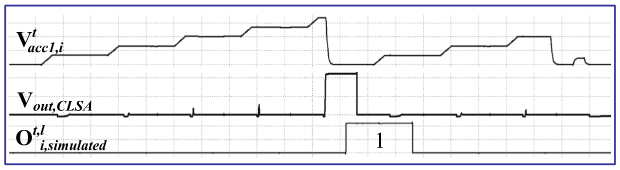
where $\Delta V^t_2(\rho)$ is the ideal linear; $\delta_2(\rho)$ and $n\{0, \sigma_2(\rho)\}$ are the ρ-dependent nonlinear errors and random variation of $\Delta V^t_{acc2,i}$. Finally, the nonlinear and variation effects on $\Delta V^t_{acc1,i}$ and $\Delta V^t_{acc2,i}$ are added into the accumulations in (17) and (19). The effects of nonlinearity and process variation on the full circuit simulation are studied in the following subsection.

4) IF NEURON CIRCUIT SIMULATION

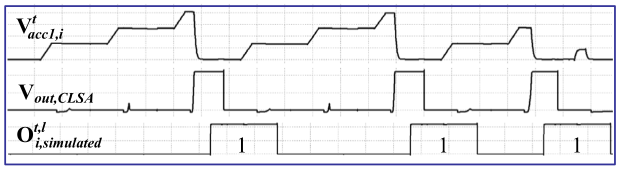
In this subsection, we conduct a full circuit simulation for an IF neuron circuit, which is directly connected with the M synapses of 288 STT memory cells. We extract the trained network parameters (i.e., weights, BN parameters, and threshold) from the PyTorch model and feed them to the circuit model, which is encapsulated in the HSPICE netlist.



(a)



(b)



(c)

FIGURE 8. The effect of process variations on postsynaptic spike generation: (a) a spike drift, (b) a missing spike, (c) adding an additional spike.

The network hyperparameters are set so that the kernel size is 3×3 and the number of input channels is 32, corresponding to an IMC array size of 1×288. Fig. 7 shows an example waveform of the CLSA output ($V_{out,CLSA}$) for a single time step considering the process variation effect. From the figure, the shift delay can fluctuate from $\tau_{delay,1}$ ($\sim$170ps) to $\tau_{delay,2}$ ($\sim$740ps). However, this variation barely affects the output spike because $V_{out,CLSA}$ is shaped and latched in a fairly stable position.

Furthermore, for multiple time steps, the process variation also affects the spike position and the total number of spikes. Fig. 8 presents the postsynaptic spike waveform for $T = 8 \times T_{spike}$ corresponding to 288 input bitstrings. From multiple Monte Carlo samples, it is very common that the positions of the spikes in the output pattern $o^{t,l}_{i,simulated}$ are shifted back and forth from the positions of the expected software simulation output $o^{t,l}_{i,expected}$. However, since we use rate encoding, this effect does not affect the final result, which is determined by the total number of spikes within T period. It can also be observed that the output pattern may miss or introduce one spike, as shown in Figs. 8(b) and 8(c). Nevertheless, the undesirable missing or addition of one spike has little impact on the final result also thanks to the use of the rate encoding. In rare cases, we also observe two missing or introduced two spikes, but that is the maximum number of incorrect spikes observed thus far in our simulations.

To quantify the impact of process variation on the robustness of the neuron implementation, we extracted the mean square error (MSE) of the output bitstreams from the multiple Monte Carlo simulations as follows

$$MSE = \frac{1}{N_{Monte}} \sum_{j=1}^{N_{Monte}} \frac{1}{T^2} \left(\sum_{t=1}^{T} o^{t,l}_{ij,expected} - \sum_{t=1}^{T} o^{t,l}_{ij,simulated} \right)^2 \tag{22}$$

where N_{Monte} is the number of Monte Carlo simulations. According to (22), the MSE for 500- Monte Carlo simulations in this experiment has a value of 1.45%. Thus, process variation essentially has an impact on the classification accuracy, but this impact is small and can be reasonably accepted. The results obtained from the above analysis could be a solid evidence for the SNN fault-tolerant nature [13].

In fact, it is not possible to simulate the whole network at the circuit level with massive input patterns. However, as we have mentioned earlier, the statistical error models of $\Delta V^t_{acc1,i}$ and $\Delta V^t_{acc2,i}$ in (20) and (21) can be exactly injected into the system model to realistically estimate the system accuracy. The details of the model and evaluation process are discussed in the next section.

V. SYSTEM-LEVEL EVALUATION

A. BSNN TRAINING MODEL

To evaluate the performance of the BSNN at the system level, we use the training method proposed in Section II and Appendix A. The network structures and major parameters are shown in Table 2. The training and inference models are written in the Python language using the PyTorch library. We use the MNIST and CIFAR-10 [35] datasets for training and evaluation. The networks are trained for 300 (50) epochs with a batch size of 128 (100) for the CIFAR-10 (MNIST) dataset. The base learning rate is set to 0.3, and the stochastic gradient descent (SGD) optimizer has a momentum of 0.9. The learning rate is scheduled with a decay factor of 10 at 50%, 70%, and 90% of the total epochs. For the CIFAR-10 dataset, we adopt BSNNs using 7 Conv layers for both networks (with and without the residual connections). Since the Conv layers occupy most of the computational workload and latency (more than 90%) [36], [37] in the deep neural network, only the hidden Conv layers are binarized in the BSNN to balance the accuracy with that of the conventional SNN [16]. Therefore, in our work, the IMC XNOR-based STT subarrays substitute for all hidden Conv layers in the BSNN evaluation.

B. MAPPING THE BSNN TO THE CIRCUIT MODEL

Fig. 9 shows the mapping of a BSNN Conv layer to the STT subarray. The main calculation workload to shift from presynaptic spikes to postsynaptic spikes is done by the IMC and IF circuits described in Section IV. As seen in the figure, we unroll each sliding window calculation for the given input into a vector of presynaptic spikes.

Then, the unrolled vector is passed to an $N \times M$ subarray through M BL decoders. The kernel is mapped and stored in the M bitcells to perform convolution between unrolled spikes and the kernel weights. If N output channels are generated simultaneously, the $N \times M$ subarray performs one sliding window, as illustrated in Fig. 9. Furthermore, intra-layer parallelism can be realized by utilizing P subarrays that calculate P parallel sliding windows [38]. P hence indicates the level of parallelism, which reflects the tradeoff between

TABLE 2. Three network structures and simulation parameters for two different datasets.

	Network structure	
MNIST (2 Conv layers)	32 Conv (rate encoding) – AvrPool2-32 Conv -AvrPool2-128FC1-512FC2-10FC3	
CIFAR-10 (7 Conv layers)	32 Conv (rate encoding)- 32 Conv- 32 Conv-32 Conv- 32 Conv- 32 Conv - 256 Conv -AvrPool4-512FC1-10FC2	
CIFAR-10 (7 Conv layers with SEW[a])	Input layer l= g (Output layer $l-1$, Output layer $l-2$) $l = 3 \div 7$	
parameters	time steps	$T = 4, 8$
	learning rate	$\eta = 0.3$
	epochs size CIFAR-10/MNIST	300 (50)
	momentum (SGD optimizer)	0.9
	batch size CIFAR-10/MNIST	128/100

[a]SEW: the networks with residual connections [20]

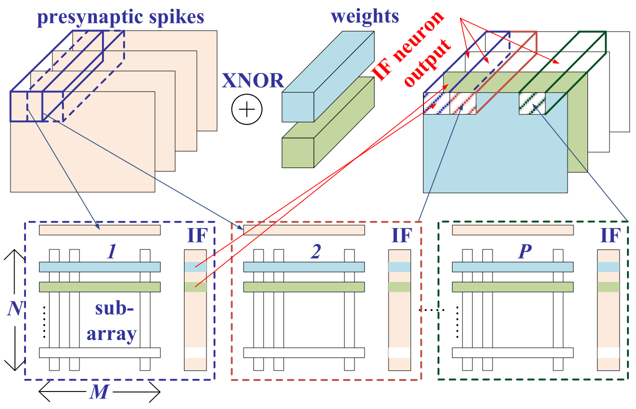


FIGURE 9. Mapping BSNN Conv layers to STT-MRAM subarrays.

the hardware cost and the speed of computation. For the practical implementation in this work, each Conv layer has $N = 32$ input and output channels, and a kernel size of 3×3 corresponds to a subarray size of 32×288 ($M = 288$).

C. THE IMPACTS OF THE NUMBER OF TIME STEPS AND PROCESS VARIATION ON CLASSIFICATION ACCURACY

We investigate the impact of the number of time steps on the classification accuracy for both the MNIST and CIFAR-10 datasets. The classification accuracies of the BSNN inferences with six network configurations are shown in Table 3.

From the table, using more time steps essentially improves accuracy. Specifically, for networks without SEW, increasing the number of time steps from 4 to 8 results in an accuracy increase of 0.64% (3.24%) for MNIST (CIFAR-10). Additionally, it is clear that the networks with SEW achieve 2.52% (1.82%) better accuracy with 4 (8) time steps for CIFAR-10 than the conventional networks [27]. These results confirm that the training method using a surrogate gradient with SEW significantly reduces the required number of time steps compared with the that of the ANN-SNN conversion method [28].

Furthermore, the non-ideal charge increments in (20) (21) are injected into the Python BSNN inference model to

TABLE 3. Classification accuracies of BSNN model on the MNIST, CIFAR-10 datasets for four and eight time steps.

Datasets	Architecture	Accuracy (%)	Timesteps
MNIST	2 Conv layers	97.50	4
MNIST	2 Conv layers	98.14	8
CIFAR-10	7 Conv layers	79.70	4
CIFAR-10	7 Conv layers	82.94	8
CIFAR-10	7 Conv layers +SEW	82.22	4
CIFAR-10	7 Conv layers +SEW	84.76	8

TABLE 4. The effect of process variation on the classification accuracy of the BSNN model for 8 time steps.

Datasets	Architecture	Accuracy (%)	
		No variations	With variations μ_{BSNN}/σ_{BSNN}
MNIST	2 Conv layers	98.14	97.92/0.23%
CIFAR-10	7 Conv layers +SEW	84.76	83.85/0.03%

evaluate the effect of process variation on the classification accuracy. This is completed by replacing the linear models of the membrane potential and threshold by the actual models with an incorporated nonlinearity bias and a Gaussian random quantity,[2] which are characterized by the Monte Carlo simulations in Section IV.B.3. The models are evaluated on the test set multiple times (a 100-variation netlist) with different variation seeds. The means (μ_{BSNN}) and standard deviations (σ_{BSNN}) are reported in Table 4. From the table, the mean classification accuracy is slightly reduced by 0.22% (0.91%) for MNIST (CIFAR-10) with 8 time steps compared to the reported accuracies for the models without variation. Additionally, the accuracies evaluated with different configurations are not much different, with standard deviations of 0.23% (MNIST) and 0.03% (CIFAR-10). In the extreme case at the $6 \times \sigma_{BSNN}$ point, the classification accuracy is degraded by 1.38% (MNIST) and 0.18% (CIFAR-10). Overall, these results permit us to conclude that process variation has a minor impact on classification accuracy and the proposed BSNN exhibits a very good level of fault tolerance.

D. EVALUATION OF THE ENERGY, THROUGHPUT, AND AREA OF THE SUBARRAY

In the proposed STT-BSNN architecture, the energy per spiking operation is provided by the energy for synapses and for the IF neuron circuit:

$$\begin{aligned} E_{spike} &= E_{synapses} + E_{neuron} \\ &= E_{wl} + E_{bitcell} + E_{neuron} \end{aligned} \tag{23}$$

where $E_{synapses}$ is the energy consumed by the synapse operations of the IMC circuit (i.e., the MAC operations). This includes the precharged energy required for the high-load word lines E_{wl} and the energy consumed by the M bitcells $E_{bitcell}$. The latter essentially accounts for the main portion

[2]To ensure high accurate model, we use look-up tables for storing the nonlinear bias and standard deviation for 289 points in Fig. 6 to implement the corresponding Python models.

TABLE 5. Comparison between the proposed BSNN and methods in previous works.

	IJCNN'19 [16]	TNNSL'20 [17]	VLSI'20 [19]	This work
synapse	MTJ	MTJ	MTJ	MTJ
neuron	MTJ	Digital	MTJ	Analog
technology	45 nm	28 nm	N/A	65 nm
network type	BSNN	SNN	BSNN	BSNN
structure	3 Conv	FC layers	2 Conv	2/7 Conv
neuron	sigmoid	IF	Possion	IF
weights	+1/-1	+1/0[c]	+1/-1	+1/0
spiking rate (*MHz*)	N/A	83	0.1	166
energy/ synapse (*fJ*)	36[b]	8.87	N/A	5.48
area/neuron (F^2)[a]	N/A	~15× 10^5	6× 10^3	32× 10^3
accuracy MNIST/ CIFAR-10	N/A/ 70.3%	91.5%/ N/A	~97.4%/ N/A	97.92%/ 83.85%

[a]The area is normalized to F^2, with F is the technology feature size.
[b]The maximum energy consumption per spiking event for a synapse, as reported in [39], [40].
[c]The full-precision weights are converted into stochastic bits (+1/0) in each time step.

of the total energy, which is proportional to the sampling time and the accumulated current drawn from the BL source voltage V_{BL}. E_{neuron} is the energy spent on the IF neuron circuit, which includes the energy needed for the CB, ACC1, ACC2 and FS subcircuits in Fig. 4. Since these circuits are all charge-based circuits, they consume energy only during switching, i.e., without any direct currents. Therefore, their energy proportions are small compared to $E_{synapses}$, which normally accounts for the main contribution in E_{spike}. Specifically, for an IMC array row of size 288, the $E_{synapses}$ for 288 synaptic elements is found to be 1.58 pJ (where $E_{wl} = 0.064pJ$ and $E_{bitcell} = 1.52pJ$), where the optimal sampling time is set to $1ns$ when using the precharged technique for the booster [31]. E_{neuron} accounts for only $0.052pJ$, which results in a total spiking operation energy of $E_{spike} = 1.63pJ$.

In this work, we define the number of operations to be equal to the size of the MAC function. This means that 288 operations are executed within one time step. Therefore, the energy efficiency E_{eff} (TOPS/W) for an IMC array row of size 288 is estimated to be $288/1.63 = 176.6$ TOPS/W.

The rough estimation of the subarray area is equal to 608[3] μm^2 for a single-row implementation with 288 bitcells. The estimation area for a neuron is equal to 115 μm^2. For a subarray of size 32×288, the number of operations is equal to $32 \times 288 = 9216$ (operations) over 8 time steps with a period of $T_{spike} = 6ns$. The throughput efficiency $T_{subarray}$ is equal to $(9216/8 \times 6) = 192$ GOPS.

If P sliding windows are processed simultaneously, the throughput efficiency increases by P times ($P \cdot T_{subarray}$).

[3]The area is estimated roughly in this work only because such design with this emerging technology is not fully available for utilization in its current state. In detail, the array size is equal to the size of a laid out bitcell multiplied by the array length, the area of the IF circuit is taken as the areas of individual devices.

Algorithm 1 BSNN Training Model With a Surrogate Gradient and Residual Connections

```
Input: input $X$, label vector $Y$, set threshold $\theta = 1$
Output: update weights $W^1$, $W^L$, $W^{b,l}$
1:  IF function:
2:      function $O = IF(u, I)$
3:          for $t = 1$ to $T$ do
4:              $u^t = u^{t-1}(1 - O^t) + I(t)$
5:              if $u^t > \theta$ then
6:                  $O^t = 1$
7:              else
8:                  $O^t = 0$
9:              end if
10:         end for
11:     end function
12: Forward:
13:     for $l = 1$ do
14:         for $t = 1$ to $T$ do
15:             $X^t = Poisson\ Generator(X)$
16:             $y^{t,l} = BN\left(W^1 * X^t\right)$
17:         end for
18:         $O^l \leftarrow IF\left(u^l, y^l\right)$
19:     end for
20:     for $l = 2$ to $L - 1$ do
21:         for $t = 1$ to $T$ do
22:             $y^{t,l} = BN\left(\alpha \cdot W^{b,l} * g\left(O^{t,l-1}, S^{t,l-2}\right)\right)$
23:         end for
24:         $O^l \leftarrow IF\left(u^l, y^l\right)$
25:     end for
26:     for $l = L$ do
27:         for $t = 1$ to $T$ do
28:             $u^{t,l} = u^{t-1,l} + W^L * O^{t,l-1}$
29:         end for
30:     end for
31: Calculate the loss and backpropagation:
32:     $\mathcal{L}_p \leftarrow ComputeLoss\left(Y, u^{T,L}\right)$
```

E. COMPARISON WITH RELATED WORKS

Table 5 summarizes a comparison with previous work on IMC architectures for SNNs. Although there are many similar works [14]–[19] on this topic, we have selected the most relevant studies with using spintronic memory and spiking networks for comparison.

Regarding accuracy, using a deeper network and direct training method, the accuracy level of BSNN in this work is ~13.8% higher than that in [16], evaluated using CIFAR-10 dataset. The implementation in [17] with spintronic synapse and digital neuron also achieves lower accuracy (by 6.3% for MNIST dataset) compared to our work. This is partly because their results are reported for a fully connected network. An all-spin SNN in [19] exhibits similar accuracy for MNIST dataset, though their spiking rate is much lower than ours. The reason is that their method requires a large sampling time to convert synaptic current into a spike duty cycle in each time step.

Algorithm 2 BSNN Inference With a Dynamic Threshold

```
Input: presynaptic spike $s_j^{t,l-1}$, initialize $\hat{\theta}_{dyn,i}^{t=0,l} = \hat{\theta}_i^l$,
    $\hat{u}_{xnor,i}^{t=0,l} = 0$
Output: postsynaptic spike $o_i^{t,l}$
1:  In-memory MAC computation:
2:      for $t = 1$ to $T$ do
3:          $\sum_{j=1}^{M} \overline{w_{ij}^{u,l} \oplus s_j^{t,l-1}}$
4:      end for
5:  Accumulation:
6:      for $t = 1$ to $T$ do
7:          $\hat{u}_{xnor,i}^{t,l} = \hat{u}_{xnor,i}^{t-1,l} + \sum_{j=1}^{M} \overline{w_{ij}^{u,l} \oplus s_j^{t,l-1}}$
8:          $\hat{\theta}_{dyn,i}^{t,l} = \hat{\theta}_{dyn,i}^{t-1,l} + \rho$
9:      end for
10: Firing and resetting:
11:     for $t = 1$ to $T$ do
12:         if $\hat{u}_{xnor,i}^{t,l} > \hat{\theta}_{dyn,i}^{t,l}$ then
13:             $o_i^{t,l} = 1$
14:             $\hat{\theta}_{dyn,i}^{t,l} = \hat{\theta}_i^l$
15:             $\hat{u}_{xnor,i}^{t,l} = 0$
16:         else
17:             $o_i^{t,l} = 0$
18:         end if
19:     end for
```

Regarding the energy, the energy consumption per synapse calculated for our BSNN is ~6.5 times lower than that of the MTJ synapse reported in [16], not considering their synapse is implemented in a smaller technology node (45nm). Similarly, our work is still 1.6X more energy-efficient than the reported synapse energy in [17], even though their work is implemented on a 28nm CMOS, theoretically ~2 times more energy-efficient than the same 65nm implementation.

Finally, for area comparison reported in F^2 (F is the technology feature size), the area per neuron of our model is essentially much better than digital implementation [17]. Still, it is not as good as all spintronic one in [19]. That advantage comes with the clear trade-offs in energy and latency, as mentioned earlier.

VI. CONCLUSION

This paper presents an in-memory BSNN based on STT-MRAM for low-power, low-latency on-edge AI applications. We propose a direct BSNN training approach using a surrogate gradient with residual connections that achieves high classification accuracy with much fewer time steps than the number of steps required in prior works. Furthermore, we propose a full-circuit solution for IMC MAC operations based on a STT-MRAM array, which allows ultrafast vector multiplication to be performed within one memory access phase. Furthermore, we propose a dynamic threshold approach for the IF circuit that mimics the neuron spiking behavior and consumes very low power. The BSNN system model is then re-evaluated using realistic circuit parameters and exact circuit simulations. The results indicate that device mismatches and nonlinearity essentially affect the misclassification accuracy of the model. Nonetheless, the accuracy degradation is insignificant, and the proposed BSNN still offers decent performance in comparison with other methods. The proposed design approach with a practical circuit solution could potentially pave the way for ultralow-power DNNs to be applied in on-edge AI applications. We are working forward to improve the architecture and design to fit deeper and larger networks.

APPENDIX A

See Algorithm 1.

APPENDIX B

See Algorithm 2.

ACKNOWLEDGMENT

The authors would like to thank the VLSI Design and Education Centre (VDEC) of The University of Tokyo in collaboration with the Rohm Corporation, the Toppan Printing Corporation, Synopsys, Inc., Cadence Design Systems, Inc., and Mentor Graphics, Inc.

REFERENCES

[1] Y. H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017, doi: 10.1109/jssc.2016.2616357.

[2] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "14.5 Envision: A 0.26-to-10tops/w subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28 nm FDSOI," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2017, pp. 246–247.

[3] D. Shin, J. Lee, J. Lee, and H.-J. Yoo, "14.2 DNPU: An 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2017, pp. 240–241.

[4] P. Narayanan, S. Ambrogio, A. Okazaki, K. Hosokawa, H. Tsai, A. Nomura, T. Yasuda, C. Mackin, S. C. Lewis, A. Friz, and M. Ishii, "Fully on-chip MAC at 14 nm enabled by accurate row-wise programming of PCM-based weights and parallel vector-transport in duration-format," in *Proc. Symp. VLSI Technol.*, Kyoto, Japan, Jun. 2021, pp. 1–2.

[5] A. Mukherjee, K. Saurav, P. Nair, S. Shekhar, and M. Lis, "A case for emerging memories in DNN accelerators," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Grenoble, France, Feb. 2021, pp. 938–941.

[6] S. Jain, L. Lin, and M. Alioto, "Broad-purpose in-memory computing for signal monitoring and machine learning workloads," *IEEE Solid-State Circuits Lett.*, vol. 3, pp. 394–397, Feb. 2020, doi: 10.1109/lssc.2020.3024838.

[7] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," 2016, *arXiv:1602.02830*.

[8] X. Sun, X. Peng, P.-Y. Chen, R. Liu, J.-S. Seo, and S. Yu, "Fully parallel RRAM synaptic array for implementing binary neural network with (+ 1, −1) weights and (+ 1, 0) neurons," in *Proc. 23rd Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jeju, South Korea, Jan. 2018, pp. 574–579.

[9] X. Sun, S. Yin, X. Peng, R. Liu, J.-S. Seo, and S. Yu, "XNOR-RRAM: A scalable and parallel resistive synaptic architecture for binary neural networks," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2018, pp. 1423–1428.

[10] S. Mittal, J. S. Vetter, and D. Li, "A survey of architectural approaches for managing embedded DRAM and non-volatile on-chip caches," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 6, pp. 1524–1537, Jun. 2015, doi: 10.1109/tpds.2014.2324563.

[11] S. Gao, B. Chen, Y. Qu, and Y. Zhao, "MRAM acceleration core for vector matrix multiplication and XNOR-binarized neural network inference," in *Proc. Int. Symp. VLSI Technol. Syst. Appl. (VLSI-TSA)*, Hsinchu, Taiwan, Aug. 2020, pp. 153–154.

[12] T. N. Pham, Q. K. Trinh, I. J. Chang, and M. Alioto, "STT-MRAM architecture with parallel accumulator for in-memory binary neural networks," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Daegu, South Korea, May 2021, pp. 1–5.

[13] W. Guo, M. E. Fouda, A. M. Eltawil, and K. N. Salama, "Neural coding in spiking neural networks: A comparative study for robust neuromorphic systems," *Frontiers Neurosci.*, vol. 15, Mar. 2021, Art. no. 638474, doi: 10.3389/fnins.2021.638474.

[14] A. Sengupta, M. Parsa, B. Han, and K. Roy, "Probabilistic deep spiking neural systems enabled by magnetic tunnel junction," *IEEE Trans. Electron Devices*, vol. 63, no. 7, pp. 2963–2970, Jul. 2016, doi: 10.1109/ted.2016.2568762.

[15] A. Sengupta, G. Srinivasan, D. Roy, and K. Roy, "Stochastic inference and learning enabled by magnetic tunnel junctions," in *IEDM Tech. Dig.*, San Francisco, CA, USA, Dec. 2018, pp. 15.16.11–15.16.14.

[16] A. Jaiswal, A. Agrawal, I. Chakraborty, D. Roy, and K. Roy, "On robustness of spin-orbit-torque based stochastic sigmoid neurons for spiking neural networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Budapest, Hungary, Jul. 2019, pp. 1–6.

[17] S. N. Pagliarini, S. Bhuin, M. M. Isgenc, A. K. Biswas, and L. Pileggi, "A probabilistic synapse with strained MTJs for spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1113–1123, Apr. 2020, doi: 10.1109/tnnls.2019.2917819.

[18] A. Jaiswal, S. Roy, G. Srinivasan, and K. Roy, "Proposal for a leaky-integrate-fire spiking neuron based on magnetoelectric switching of ferromagnets," *IEEE Trans. Electron Devices*, vol. 64, no. 4, pp. 1818–1824, Apr. 2017, doi: 10.1109/TED.2017.2671353.

[19] M. H. Wu, M.-S. Huang, Z. Zhu, F.-X. Liang, M.-C. Hong, J. Deng, and J.-H. Wei, "Compact probabilistic Poisson neuron based on back-hopping oscillation in STT-MRAM for all-spin deep spiking neural network," in *Proc. IEEE Symp. VLSI Technol.*, Honolulu, HI, USA, Jun. 2020, pp. 1–2.

[20] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian, "Deep residual learning in spiking neural networks," 2021, *arXiv:2102.04159*.

[21] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. ECCV*, Oct. 2016, pp. 525–542.

[22] Y. Kim and P. Panda, "Revisiting batch normalization for training low-latency deep spiking neural networks from scratch," 2020, *arXiv:2010.01729*.

[23] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, "Direct training for spiking neural networks: Faster, larger, better," in *Proc. AAAI Conf. Artif. Intell.*, Jul. 2019, vol. 33, no. 1, pp. 1311–1318, doi: 10.1609/aaai.v33i01.33011311.

[24] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li, "Going deeper with directly-trained larger spiking neural networks," 2020, *arXiv:2011.05280*.

[25] M. A. Lebdeh, H. Abunahla, B. Mohammad, and M. Al-Qutayri, "An efficient heterogeneous memristive xnor for in-memory computing," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 9, pp. 2427–2437, Sep. 2017, doi: 10.1109/tcsi.2017.2706299.

[26] T. Hirtzlin, B. Penkovsky, M. Bocquet, J.-O. Klein, J.-M. Portal, and D. Querlioz, "Stochastic computing for hardware implementation of binarized neural networks," *IEEE Access*, vol. 7, pp. 76394–76403, 2019, doi: 10.1109/access.2019.2921104.

[27] H.-H. Lien, C.-W. Hsu, and T.-S. Chang, "VSA: Reconfigurable vector-wise spiking neural network accelerator," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Daegu, South Korea, May 2021, pp. 1–5.

[28] Y. Wang, Y. Xu, R. Yan, and H. Tang, "Deep spiking neural networks with binary weights for object recognition," *IEEE Trans. Cognit. Develop. Syst.*, vol. 13, no. 3, pp. 514–523, Sep. 2021, doi: 10.1109/tcds.2020.2971655.

[29] Q. K. Trinh, S. Ruocco, and M. Alioto, "Voltage scaled STT-MRAMs towards minimum-energy write access," *IEEE J. Emerg. Select. Topics Circuits Syst.*, vol. 6, no. 3, pp. 305–318, Sep. 2016, doi: 10.1109/JETCAS.2016.2547702.

[30] Q. K. Trinh, "STT-MRAMS circuit techniques for enhanced robustness in low power embedded applications," Ph.D. dissertation, ECE Dept., Nat. Univ. Singapore, Singapore, 2017. [Online]. Available: https://scholarbank.nus.edu.sg/handle/10635/141255

[31] Q. K. Trinh, S. Ruocco, and M. Alioto, "Novel boosted-voltage sensing scheme for variation-resilient STT-MRAM read," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 10, pp. 1652–1660, Oct. 2016, doi: 10.1109/tcsi.2016.2582203.

[32] M. van Elzakker, E. van Tuijl, P. Geraedts, D. Schinkel, E. A. M. Klumperink, and B. Nauta, "A 10-bit charge-redistribution ADC consuming 1.9 μW at 1 MS/s," *IEEE J. Solid-State Circuits*, vol. 45, no. 5, pp. 1007–1015, May 2010, doi: 10.1109/jssc.2010.2043893.

[33] T. Sakurai and A. R. Newton, "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas," *IEEE J. Solid-State Circuits*, vol. 25, no. 2, pp. 584–594, Apr. 1990, doi: 10.1109/4.52187.

[34] E. Chen, D. Apalkov, and Z. Diao, "Advances and future prospects of spin-transfer torque random access memory," *IEEE Trans. Magn.*, vol. 46, no. 6, pp. 1873–1878, Jun. 2010, doi: 10.1109/tmag.2010.2042041.

[35] A. Krizhevsky and G. Hinton, *Learning Multiple Layers of Features from Tiny Images*. Toronto, ON, Canada: Univ. Toronto, 2009.

[36] Y. Chen, J. Zhang, Y. Xu, Y. Zhang, R. Zhang, and Y. Nakashima, "An efficient ReRAM-based inference accelerator for convolutional neural networks via activation reuse," *IEICE Electron. Exp.*, vol. 16, no. 18, Jun. 2019, Art. no. 20190396, doi: 10.1587/elex.16.20190396.

[37] L. Jiang, M. Kim, W. Wen, and D. Wang, "XNOR-POP: A processing-in-memory architecture for binary convolutional neural networks in wide-IO2 drams," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Taipei, Taiwan, Jul. 2017, pp. 1–6.

[38] L. Song, X. Qian, H. Li, and Y. Chen, "PipeLayer: A pipelined ReRAM-based accelerator for deep learning," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Austin, TX, USA, Feb. 2017, pp. 541–552.

[39] G. Srinivasan, A. Sengupta, and K. Roy, "Magnetic tunnel junction enabled all-spin stochastic spiking neural network," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Lausanne, Switzerland, Mar. 2017, pp. 530–535.

[40] G. Srinivasan, A. Sengupta, and K. Roy, "Magnetic tunnel junction based long-term short-term stochastic synapse for a spiking neural network with on-chip STDP learning," *Sci. Rep.*, vol. 6, no. 1, pp. 1–13, Jul. 2016, doi: 10.1038/srep29545.

VAN-TINH NGUYEN (Graduate Student Member, IEEE) received the B.E. degree from the Belarusian State University of Informatics and Radioelectronics, Minsk, Belarus, in 2012. He is currently pursuing the Ph.D. degree with the Division of Information Science, Nara Institute of Science and Technology, Japan. His research interests include machine learning, stochastic computing, and emerging memory technologies.

QUANG-KIEN TRINH (Member, IEEE) received the Ph.D. degree in computer engineering from the National University of Singapore, Singapore, in 2018. He is working as the Deputy Head of the Department of Microprocessor Engineering, Le Quy Don Technical University, Hanoi, Vietnam. His research interests include low-power integrated circuit design, emerging memory technologies, and hardware security.

RENYUAN ZHANG (Senior Member, IEEE) received the M.E. degree from Waseda University, in 2010, and the Ph.D. degree from The University of Tokyo, in 2013. He was an Assistant Professor with the Japan Advanced Institute of Science and Technology, from 2013 to 2017. He has been an Assistant Professor and an Associate Professor with the Nara Institute of Science and Technology, since 2017 and 2021, respectively. His research interests include analog–digital mixed circuits and approximate computing. He is a member of IEICE.

YASUHIKO NAKASHIMA (Senior Member, IEEE) received the B.E., M.E., and Ph.D. degrees in computer engineering from Kyoto University, in 1986, 1988, and 1998, respectively. He was a Computer Architect with the Computer and System Architecture Department, Fujitsu Ltd., from 1988 to 1999. From 1999 to 2005, he was an Associate Professor at the Graduate School of Economics, Kyoto University. Since 2006, he has been a Professor with the Graduate School of Information Science, Nara Institute of Science and Technology. His research interests include processor architectures, emulation, CMOS circuit design, and evolutionary computation. He is a fellow of IEICE and a member of ACM and IPSJ.