TFSRAM: A 249.8TOPS/W Timing-to-First-Spike Compute-in-Memory Neuromorphic Processing Engine With Twin-Column SRAM Synapses

Ziru Li, *Graduate Student Member, IEEE*, Qilin Zheng, *Graduate Student Member, IEEE*, Jonathan Ku, *Graduate Student Member, IEEE*, Brady Taylor, *Graduate Student Member, IEEE*, and Hai Li, *Fellow, IEEE*

***Abstract*—Spiking neural networks (SNNs) have shown high efficiency in information processing. Time-to-first-spike (TTFS) coding, which encodes the information to the times of first spikes, has been introduced in SNNs due to its simple data representation. However, there is a lack of studies in synapse mapping and neuron circuits for compute-in-memory (CIM) neuromorphic design for low-power TTFS-based SNN inference. This work presents a CIM neuromorphic processing engine for TTFS-based SNNs, which comprises a 64x64 8T-SRAM array for synapse storage and current-based post-neuron circuits. We propose to map the positive and negative synaptic weights to adjacent columns in the SRAM array and accumulate the membrane potential in the dedicated post-neuron circuits. Two techniques are presented through hardware-software co-design: "Multi-Level Firing Threshold Adjustment" to mitigate the impact of process variations, and "Timing Threshold Adjustment" to further speed up the computation. The proposed processing engine achieves the energy efficiency 249.8 TOPS/W under 8-bit inputs and signed 4-bit weights.**

***Index Terms*—Compute-in-memory, neuromorphic computing, spiking-neural-network, time-to-first-spike, temporal coding.**

I. Introduction

WIDE applications of deep neural networks (DNNs) prompt dedicated hardware accelerators for better performance under tight power consumption constraints [5], [6]. Even though the state-of-the-art (SoTA) domain-specific accelerators have raised the energy efficiency to a higher level compared with general-purpose CPUs and GPUs, the continued advance in hardware efficiency becomes difficult without game-changing technologies [2], [14]. In contrast, the fast-growing artificial intelligence and internet of things (AIoT) field require recklessly higher efficiency under sub-watt power supplies, calling for the next-generation computing schemes for the AIoT cognitive tasks as well as their hardware implementations.

The existing DNN algorithms are mainly *frame-driven*. The data of each frame are evaluated by the model at a constant frame rate. In contrast, spiking neural networks (SNNs), regarded as the third-generation neural networks [18], offer an alternative *event-driven* solution to realize high computational efficiency, especially when the input data are encoded in a sparse format. A neuron in an SNN communicates with another neuron using a sequence of spiking events. Unlike DNNs, neurons in an SNN are activated only when generating or receiving spikes, resulting in very low energy consumption.

There are several ways to modulate information on spike events. Generally, fewer spike events generated during execution imply less energy consumption. A common data encoding scheme of SNNs is rate-coding, in which data are carried by the spike frequency [4], [8]. For example, a spike train with a higher frequency propagates a stronger signal to its destination neuron. Usually, a large amount of spike events are needed to convey a datum via averaging the temporal information of the spike events. Another limitation of rate-coding-based networks is that the inference accuracy is significantly affected by parameters such as the firing threshold [28]. Temporal coding scheme, which encodes data to spike times, creates a new path to achieve high efficiency by enabling the data representation with sparse spikes [11]. Particularly, the time-to-first-spike (TTFS) encoding [24] modulates the input data in the arrival time of the first spike, which significantly reduces spike numbers in propagation and thus saves the dominating dynamic computing energy consumption. Even though the restriction of spike number in TTFS limits the number of neurons that propagate spikes in the SNN inference, which may degrade the overall accuracy, it incurs much less energy consumption compared to the conventional rate-coded SNNs [20].

Built with different data encoding schemes, the spiking neuromorphic systems, i.e., the specific hardware that emulate SNN models, can improve energy efficiency and computation parallelism by leveraging asynchronous circuits that work in the

Received 20 May 2024; revised 28 July 2024; accepted 27 August 2024. Date of publication 4 September 2024; date of current version 8 November 2024. This work was supported in part by the National Science Foundation under Grant 2328805, Grant 2112562, Grant 2328712, and Grant 2332744 and in part by the Army Research Office under Grant W911NF-19-2-0107. The review of this article was arranged by Associate Editor Yang Yi. *(Corresponding author: Ziru Li.)*

The authors are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27707 USA (e-mail: ziru.li@duke.edu; qilin.zheng@duke.edu; jonathan.ku@duke.edu; brady.g.taylor@duke.edu; hai.li@duke.edu).

Digital Object Identifier 10.1109/TCASAI.2024.3452649

event-driven manner. These designs can thus eliminate power-hungry global clock networks [1]. Among CMOS-based practices, digital neuromorphic systems, such as GoldenGate [19], TrueNorth [1], SpiNNaker [10] and Loihi [7], have good programmability to support multiple encoding schemes, including rate-coding and temporal-coding. However, they approximate the continuous neuron dynamics and spike propagation in discrete time steps, which is not biologically plausible and degrades the inference accuracy. Mixed-signal neuromorphic systems such as BrainScaleS [22] and Neurogrid [3] are built with analog neuron circuits. Power-hungry digital-to-analog/analog-to-digital converters (DACs & ADCs) are required as the interface between the SRAM-based synapses and analog neuron circuits. Confronted with the limitations of those neuromorphic systems using conventional digital and analog CMOS technologies, researchers introduce new technologies and architectures to boost the computational efficiency vastly.

The compute-in-memory (CIM) spiking neuromorphic designs based on either conventional static random-access-memory (SRAM) [13] or emerging non-volatile memory (NVM), e.g. resistive random-access-memory (ReRAM) [15], [16], [27], are promising candidates featuring low-power implementation, which perfectly meets the requirements of AIoT cognitive tasks. Regarding implementing SNNs with CIM design paradigm, the following questions need to be clarified:

- How to map the excitatory and inhibitory synapses (i.e. positive and negative weights) to the memory crossbar array?
- How to leverage the event-driven property of SNNs to reduce energy consumption as most neurons are in the idle state?

In order to answer these questions and maximize the efficiency gain of the emerging neuromorphic computing paradigm, in this paper, we propose TFSRAM, a TTFS CIM processing engine that comprises a 64x64 8T-SRAM array for twin-column synapse storage, and DAC-free current-based post-neuron circuits to implement the integrate-and-fire (IF) neuron model. The proposed engine achieves 249.8 TOPS/W energy efficiency featuring the twin-column synapse mapping scheme and a power-efficient 2-rail post-neuron circuit design. Two SNN threshold adjusting techniques are provided to further boost the power efficiency and robustness of the proposed processing engine. More specifically, the main contributions of this work include:

1) We use a twin-column excitatory/inhibitory synaptic weight mapping scheme (contrast to the conventional row-wise mapping scheme) and design a brand-new set of neuron circuits. To our best knowledge, TFSRAM is the first SRAM-based CIM neuromorphic design that elaborates a complete set of peripheral circuits to compute TTFS-based SNNs and leverages the event-driven characteristics to speed up the computation.
2) Regarding the widely-concerned circuit process variations, we propose a multi-level "firing threshold adjustment" method that effectively recovers the inference accuracy degradation with only 1.5% hardware overhead.
3) We also devise the "timing threshold adjustment" method, a design-automation technique specifically for the proposed post-synaptic neuron circuits. It removes a considerable amount of unnecessary spike generation between adjacent neural network layers and thus speeds up the inference.

We fabricated the proposed processing engine with 65nm process. The prototype chip measurement results show that the proposed engine achieves 249.8 TOPS/W energy efficiency. Our simulation results on the proposed SNN threshold adjusting techniques also present an outstanding resilience to variations by maintaining 90.1% accuracy under the process variations with a 20% standard deviation.

The rest of the paper is organized as follows. We first provide preliminary knowledge about prior CIM neuromorphic designs and the inference TTFS-based SNNs in Section II. Section III describes our proposed processing engine TFSRAM, as well as the two threshold adjustment schemes. In Section V, we evaluate the performance of our proposed threshold adjustment schemes. In Section V-D, we provide the chip measurement results of TFSRAM. Finally, we conclude this work in Section VII.

II. Preliminaries

A. TTFS-Based SNNs

In SNNs, the classical integrate-and-fire (IF) neuron model with a current-based (CuBa) activation follows the following equation [12]:

$$C_{mem}\frac{dV_{mem}}{dt} = \sum_i w_i \sum_{t_i} \delta(t - t_i) \quad (while\ V_{mem} < V_{th}), \tag{1}$$

where C_{mem} is the membrane capacitance, w_i are synaptic weights, t is time, t_i are pre-spike times[1], $\delta(t)$ is the Dirac delta function. The membrane potential V_{mem} is accumulated by CuBa activation, which is the sum of pre-spikes modulated by synapses, during the inference time window. When V_{mem} reaches the preset threshold V_{th}, a post-spike will be generated ("fired") by the post-neuron, and V_{mem} will be reset. The biologically-plausible TTFS coding scheme [24] encodes the data to the firing time of the first spike in the time window, i.e. t_i in Eq. (1). Single spike firing per neuron is adequate to complete inference in each compute time window. Because of this unique property, TTFS needs dedicated circuits to realize single-spike t_i.

The inference process of the TTFS-based SNN with the standard IF model is depicted in Fig. 2. Two properties of this event-driven inference process can be observed. First, since each neuron only fires a single spike during the inference, the neuron will be in an idle state after the post-spike is fired. Second, during the membrane potential accumulation, the earlier pre-spikes contribute more to the membrane potential, while the pre-spikes coming later than the single post-spike have no

[1]Pre- and Post-spikes are short for "pre-synaptic spikes" and "post-synaptic spikes", respectively.

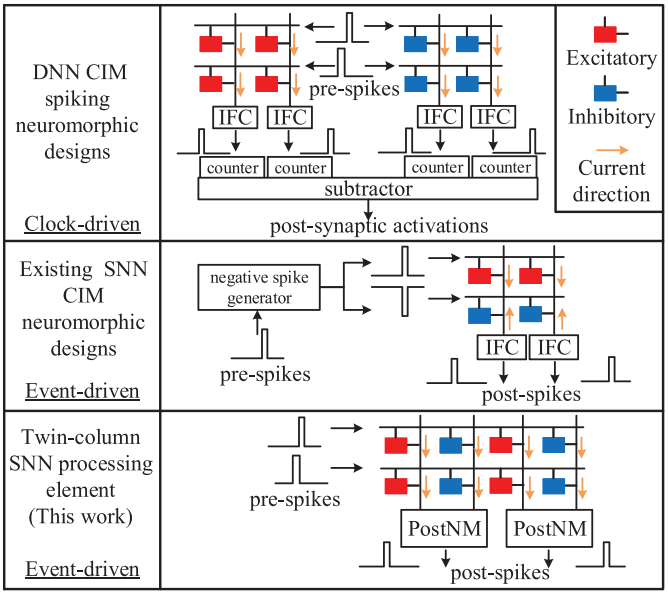


Fig. 1. Comparison of CIM neuromorphic designs: DNN spiking scheme (top), the conventional SNN scheme (middle) and the proposed SNN scheme (bottom).

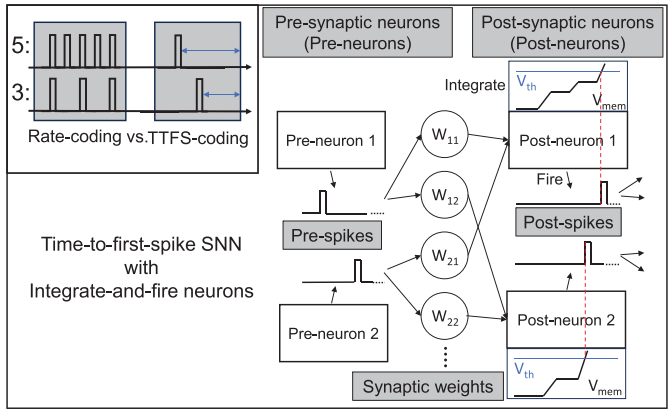


Fig. 2. Time-to-first-spike SNN with integrate-and-fire neurons.

impact on its firing time. In other words, an earlier spike time represents stronger activation and later spike time represents weaker activation. We take advantage of these properties during the SNN inference in our neuromorphic designs to greatly reduce unnecessary power consumption and computation latency.

B. CIM Neuromorphic Design

CIM neuromorphic designs perform neural network computation (DNNs or SNNs) as specialized energy-efficient AI platforms. Prior CIM neuromorphic designs for DNN inference borrow the spike-based data encoding from SNNs to simplify the computation circuits and remove the power-consuming DACs and ADCs. There have been rate-coding-based DNN process engine (PE) designs [17], [27], which count the spike numbers in fixed-time windows to decode the data. These designs suffer from the large quantization errors caused by the averaging-distributed spikes of the rate-coding scheme. Li et al. [15] proposed a single-spiking ReRAM-based DNN PE, which adopted TTFS encoding by taking the arrival time of a single spike to represent a datum. This greatly reduced the spike number and power consumption. However, the frame-driven DNNs cannot fulfill the potential of TTFS because the computation is still driven by a fixed clock cycle repeatedly. When applying TTFS to the event-driven SNNs, the data format of a single spike event allows the possibility of eliminating unnecessary idle periods without the restriction of a fixed clock cycle. As such, the circuit implementation of event-driven SNNs should adopt different computation principles from the DNN spiking neuromorphic designs.

As depicted in the top row of Fig. 1, the conventional DNN spiking neuromorphic designs [15], [27] adopt the clock-driven dataflow. The excitatory and inhibitory synapses are mapped to different spatial positions of CIM memory arrays. The computation with positive and negative synaptic weights are executed separately, and the post-synaptic activations are calculated from the difference of the bitline outputs at the clock edge. This is suitable for the clock-driven DNN hardware but not viable in the event-driven SNN scheme in that the latter does not have clock edges.

In the event-driven SNN hardware, the post-synaptic activations are accumulated throughout the time window. Hence, the activations from the excitatory synapses should be subtracted by those from the inhibitory synapses every moment and accumulated by the post-neuron modules. In some existing NVM-based SNN neuromorphic designs [25], [26], the synapses are mapped to adjacent excitatory and inhibitory synapse rows to form a differential pair, and input positive and negative spike voltages are fed into the two rows respectively, as depicted in Fig. 1 (middle). It requires additional circuits to duplicate and reverse the pre-synaptic spikes, which excessively complicates the computing peripheral circuits. [13] presents an SRAM-based neuromorphic processors for the inference of rate-coded SNNs with 28nm technology node. The chip stores 4-bit/8-bit synaptic weights in the 8T-SRAM arrays, takes in input spikes along the read wordlines and accumulates the membrane potential from the weighted spikes along the read bitlines. The multi-bit signed synapses are mapped to the same row instead of adjacent rows. Two sets of area-consuming capacitor-DAC adders are used to accumulate potential from excitatory and inhibitory synapses, controlled by the signed bit, as shown in Fig. 3(a).

III. Twin-Column SNN Processing Engine

In the following two sections, we first introduce the basic circuit components of TFSRAM, our proposed TTFS CIM neuromorphic processing engine and its synapse mapping scheme (Section III). Then, we propose two novel computer-aided design methods in Section IV, including "multi-level firing threshold adjustment" and "timing threshold adjustment" (Section IV), to improve the inference robustness and energy efficiency of the TTFS neuromorphic design, respectively.

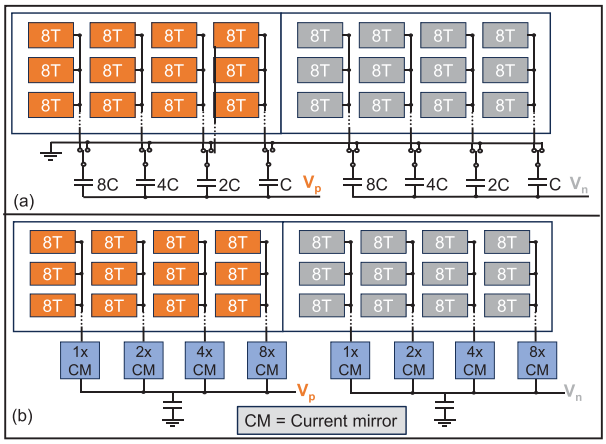


Fig. 3. Comparison between (a) prior capacitor-based neuromorphic macros [13] and (b) our current-based TFSRAM.

A. Fundamental Circuit Components

TFSRAM adopts the TTFS coding scheme to pursue high energy efficiency. Fig. 4(a) presents the overview of our twin-column SNN processing engine (PE), namely TFSRAM. The main components of TFSRAM include a 64×64 SRAM array with read/write interface, 64 spike generators to generate pre-synaptic spikes, 8 post-neuron circuits, and spike timing counters to capture the spike times of post-synaptic spikes. The SRAM array contains a 64×8 8-bit weight matrix.

B. Twin-Column Synapse Mapping on the 8T-SRAM Array

The synapses stored in the 8T-SRAM synaptic array modulate the input wordline voltages and generate CuBa activations in the form of bitline currents. When the pre-synaptic spikes are horizontally fed to the read wordlines (RWLs) of the 8T-SRAM array, spiking currents are generated along the read bitlines (RBLs) vertically from the cells that store '1's. Assume that the weights stored in the 8T-SRAM cells along $j-th$ RBL are $G_{i,j}$, $i = 0, 1, \cdots, 63$. The strength of CuBa activation along each bitline is calculated as:

$$I_{RBL,j} = \sum_{i=0}^{N-1} W_{i,j} I_D \delta(t - t_i), \tag{2}$$

where I_D is the current flowing through the read devices of the 8T-SRAM cell when the RWL is activated by the pre-synaptic spike $\delta(t - t_i)$. Note that there are multi-bit excitatory and inhibitory synapses in SNN models. We need a novel synapse mapping scheme and corresponding neuron circuit designs to support it.

In our proposed twin-column SNN processing element, we use the eight adjacent columns of 8T-SRAM cells to represent the multi-bit excitatory and inhibitory synapses. To be specific, one signed 8-bit synaptic weight is partitioned to a 4-bit positive part and a 4-bit negative part, and programmed to the SRAM cells in adjacent 4 RBLs along the same RWL, respectively. The positive and negative RBLs form a differential pair. Each 8 RBLs belonging to the same synaptic weight are connected to the same post-neuron circuit.

The twin-column synapse mapping provides an alternative solution compared to the row-wised synapse mapping scheme adopted in [25], [26]. In the row-wised synapse mapping scheme, the excitatory and inhibitory synapses modulating the same inputs are mapped to different rows, hence additional circuits are required to duplicate and reverse the pre-synaptic spikes and feed into the positive and negative rows, leading to area overhead and inevitable delay of spike timing. In the column-wise mapping, the excitatory and inhibitory synapses share the same inputs. The subtraction between the potential accumulated from the excitatory and inhibitory synapses is executed in the neuron circuits which will be introduced in the following paragraphs.

C. 2-Rail Post-Neuron Circuit

The post-neuron circuit (Fig. 5) computes the difference of the complementary spiking currents from twin-column synapses and implements neuron dynamics. Each post-neuron module contains 8 current mirrors with proportional gains along the 8 RBLs, and one 2-rail IF circuit (IFC). As shown in Fig. 5, the 2-rail IFC contains two capacitors $\mathrm{C_p}$ and $\mathrm{C_n}$ with identical capacitance. The current mirrors from the positive part are connected to $\mathrm{C_p}$, while the current mirrors from the negative part are connected to $\mathrm{C_n}$. $\mathrm{C_p}$ accumulates membrane potential ($\mathrm{V_{mem,p}}$) from 0V, while $\mathrm{C_n}$ accumulates membrane potential ($\mathrm{V_{mem,n}}$) from $\mathrm{V_{th}}$. Once $\mathrm{V_{mem,p}}$ is larger than $\mathrm{V_{mem,n}}$, the output post-synaptic spike ($\mathrm{V_{out}}$) is generated with its pulse width controlled by $\mathrm{V_{ref}}$. The differential membrane potential on $\mathrm{C_p}$/$\mathrm{C_n}$ will then be reset to 0V/$\mathrm{V_{th}}$ by the following inverter chain. The post-synaptic spikes are converted to the digital domain by the subsequent 8-bit spike timing counters.

In each post-neuron circuit, assume $\mathrm{C_p} = \mathrm{C_n} = \mathrm{C_m}$, the input spiking currents along RBLs from the positive part are denoted as $I^p_{RBL,j}$, the input spiking currents along RBLs from the negative part are denoted as $I^n_{RBL,j}$, and the current gains of the corresponding current mirrors are $A_{cm,j}$ (j=0, 1, 2, 3). The neural dynamics performed in the post-neuron module are formulated as:

$$\begin{aligned} V_{mem} &= V_{mem,p}(t) - (V_{mem,n}(t) - V_{th}); \\ V_{mem,p}(t) &= \int_0^t \tfrac{1}{C_p} \textstyle\sum_{j=0}^{3} A_{cm,j} I^p_{RBL,j}(t) dt; \\ V_{mem,n}(t) &= V_{th} + \int_0^t \tfrac{1}{C_n} \textstyle\sum_{j=0}^{3} A_{cm,j} I^n_{RBL,j}(t) dt; \\ C_m \tfrac{dV_{mem}}{dt} &= \textstyle\sum_{j=0}^{3} A_{cm,j} (I^p_{RBL,j}(t) - I^n_{RBL,j}(t)). \end{aligned} \tag{3}$$

Eq. (3) resembles Eq. (1), indicating that the twin-column PE is able to compute the IF neural dynamics. Fig. 5 shows the post-layout transient simulation waveform of the post-neuron module under $\mathrm{V_{th}} = 400\mathrm{mV}$ and $\mathrm{V_{ref}}$=150mV, validating the function of twin-column PE. The key circuitry parameters are summarized in Table I. In the simulation waveform, $V_{mem,p}$ takes in two spikes at $10ns$ and $170ns$ while $V_{mem,n}$ takes in on spike at $10ns$. $V_{mem,p}$ is accumulated from 0 and increases at $10ns$ and $170ns$, while $V_{mem,n}$ is accumulated from $V_{th} = 400\mathrm{mV}$ and increases at $10ns$. When $V_{mem,p}$ exceeds $V_{mem,n}$ ($\sim 179ns$), the IFC generates an output post-spike. The pulse width of the output post-spike is $25ns$ ($179 \sim 204ns$), which is large enough to be captured by the subsequent 100MHz spike timing counter.

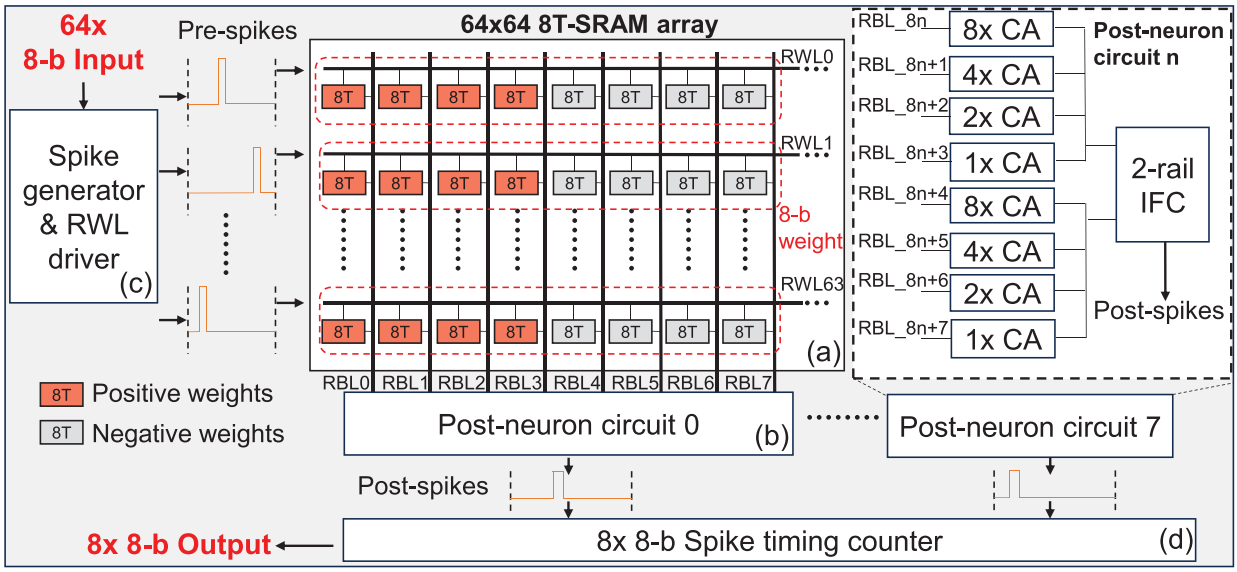


Fig. 4. Design overview of TFSRAM. The main components of TFSRAM include the 64×64 8T-SRAM array (a) for synaptic weight storage, 8 post-neuron circuits (b) to accumulate membrane potential and generate post-synaptic spikes, spike generators (c) and spiking timing counters (d) for the conversion between pre-/post-synaptic spikes and digital inputs/outputs.

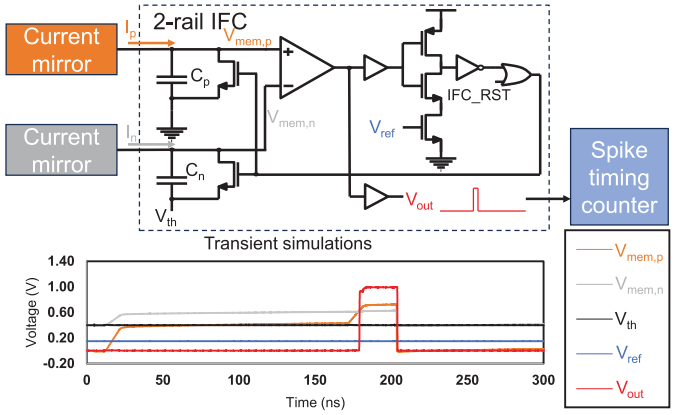


Fig. 5. Post-neuron circuit and transient simulation waveform.

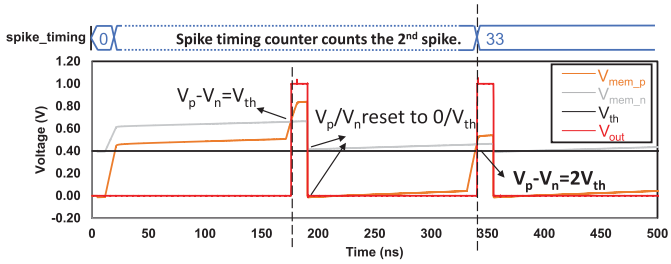


Fig. 6. Threshold voltage scaling increases the dynamic range of V_{mem}.

TABLE I
CRITICAL CIRCUITRY PARAMETERS IN THE POST-NEURON CIRCUIT

Symbol	Value	Symbol	Value
V_{DD}	1V	Frequency	100MHz
C_p	110fF	C_n	110fF
V_{ref}	150mV	$V_{th,0}$	400mV

D. Threshold Voltage Scaling

In the 2-rail IFC, the equivalent V_{mem} cannot surpass $V_{DD} - V_{th}$. To expand the dynamic range of V_{mem}, we introduce the threshold voltage scaling (TVS) mechanism specifically tailored for temporal-coded SNNs. The spike timing counter takes in a control signal named vth_scale to decide which spike from each neuron within the inference time window is regarded as the "first spike", whose spike timing is counted as the output. During the inference, V_{mem} repetitively increases towards V_{th} to trigger post-synaptic spikes and is reset to 0 after it reaches V_{th}. Therefore, the n-th post-synaptic spike indicates that the membrane potential has increased to the equivalent n-fold V_{th} (Fig. 6). If V_{th} is relatively high so that V_p reaches V_{DD} before V_{mem} reaches V_{th}, the applied threshold voltage can be reduced to 1/n V_{th}, and the n-th post-synaptic spike will be detected as the "first spike" to get equivalent threshold voltage of V_{th}.

IV. FIRING/TIMING THRESHOLD ADJUST

Aiming to further improve the efficiency and robustness of the twin-column SNN PE, we propose two schemes, i.e. multi-level firing threshold adjustment (MFTA) and timing threshold adjustment (TTA). MFTA and TTA are orthogonal to each other and can be simultaneously applied to the SNN processing engine.

A. Multi-Level Firing Threshold Adjustment

Process variations of transistors in the proposed SNN PE are inevitable. They impose noises on the CuBa activation

Algorithm 1: MFTA scheme

Input: Target firing time of neurons $t_{exp}(i,l,n)$
Input: No. of MFTA each neuron should complete C
Output: The threshold of each neuron $V_{th}(l,n)$
No. of MFTA each neuron has completed $c(l,n)=0$;
The initial threshold of each neuron $V_{th}(l,n)=V_{th,N}$;
while *Any* $c(l,n)<C$ **do**
  for $(l=0; l<L; l++)$ **do**
    Convergence detector $CD(l,n)=0$;
    Value indicating last adjustment result $r(l,n)$;
    while $CD(l,n)=0$ **do**
      $t(i,l,n)=$ **TTFS** $(Input_i)$ //$Input_i$ is processed;
      if $t(i,l,n)<t_{exp}(i,l,n)$ *and* $c(l,n)<C$ **then**
        $V_{th}(l,n)++$; $c(l,n)++$;
        if $r(l,n)=0$ **then** $CD(l,n)=1$;
        else $r(l,n)=1$;
      else if $t(i,l,n)>t_{exp}(i,l,n)$ *and* $c(l,n)<C$ **then**
        $V_{th}(l,n)--$; $c(l,n)++$;
        if $r(l,n)=1$ **then** $CD(l,n)=1$;
        else $r(l,n)=0$;

from the SRAM-based synapses, affect the post-spike firing times, and subsequently degrade the computational accuracy. This problem can be effectively solved by calibrating firing thresholds V_{th} to correct the firing times. When the post-spike of a post-neuron is fired earlier than expected due to the process variations, we deliberately raise its V_{th} to postpone the post-spikes, and vice versa.

MFTA is applied to SNN processing element right after deploying the pre-trained SNN model to the SRAM array. In MFTA, V_{th} of each post-neuron module can be selected from a preset group of evenly-distributed discrete values $\{V_{th,1}, V_{th,2}, ..., V_{th,2N}\}$ rather than continuous variables.

MFTA is applied to SNN processing element only *once* for each SNN deployment before iterative workloads begin. A set of input test patterns are fed to adjust the thresholds. MFTA starts from $V_{th}=V_{th,N}$, which is the expected global threshold voltage in the pre-trained SNN model. Before MFTA starts, the expected firing time of the $n\text{-}th$ neuron in Layer l for the $i\text{-}th$ test input, denoted as $t_{exp}(i,l,n)$, is derived from the pre-trained SNN and stored in the cache.

Algorithm 1 shows the principle of our proposed MFTA scheme. For each test input $Input_i$, MFTA starts from the first layer ($l=1$). The threshold voltage of all the neurons in the first layer will be adjusted according to the difference between $t_{exp}(i,1,n)$ and the actual post-spike time $t(i,1,n)$. If $t(i,1,n)>t_{exp}(i,1,n)$, the threshold voltage of the $n\text{-}th$ neuron lowers one discrete step of preset threshold values; otherwise it rises up to the next higher preset threshold value. Through experiments, we observe that it takes no more than 10 iterations in most cases to converge into an optimal threshold value. When all the neuron thresholds in the SNN model have been adjusted, the next input test pattern will be taken in. By repeatedly conducting MFTA with different inputs, the thresholds are adjusted to proper levels to compensate for the firing time deviation caused by process variations.

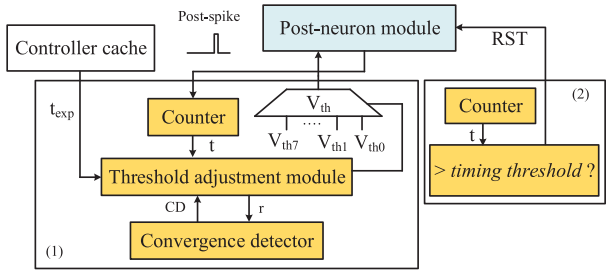


Fig. 7. The block diagram of (1) the multi-level firing threshold adjustment module and (2) the timing threshold adjustment controller.

Apparently, how to feed input test patterns for threshold adjustment is critical for MFTA scheme. By setting C (adjustment number allowed by each neuron), Algorithm 1 guarantees that all the firing thresholds can be involved in the adjustment. This method also prevents unnecessary threshold changes and dynamically schedules the number of required test inputs.

Based on the proposed MFTA method, we implement the circuit for MFTA control logic at TSMC 65nm technology node. The block diagram is depicted in Fig. 7 (1). A counter is used to convert the post-spike timing to the digital value t. The expected firing time, i.e. t_{exp}, is fetched from the controller cache. A threshold adjustment module compares t versus t_{exp} and adjusts the threshold guided by Algorithm 1. A convergence detector gets the last adjustment result r from the threshold adjustment module, and determines if the adjustment has already converged. It disables the threshold adjustment module once the convergence condition in Algorithm 1 is fulfilled. The circuit results in only 1.5% energy overhead for our SNN processing element. The efficacy of MFTA scheme is given in Section V-B.

B. Timing Threshold Adjustment

While MFTA improves the resilience against device process variations, TTA scheme speeds up the inference and enables further energy savings. In each TTFS-SNN layer, neurons fire spikes earlier, representing stronger activations; while other neurons fire spikes later and contribute less to the membrane potential accumulation of the neurons in the subsequent layers. We propose TTA scheme for our neuromorphic designs by taking advantage of the event-driven nature of TTFS-based SNN models. A *timing threshold* will be set. During each iteration, the post-spikes that fire before the *timing threshold* will be propagated. At the *timing threshold*, the activation propagation of all the neurons will cease, and the next iteration starts.

TTA scheme cuts the duration of iterations by allowing only a portion of neurons in each layer to propagate their spikes. It helps reduce energy consumption as it shortens the time windows and eliminates unnecessary spike generation. On the other hand, the inference accuracy could be affected due to the loss of some post-spikes, but the degradation is marginal because the propagated strong activations have contributed to the

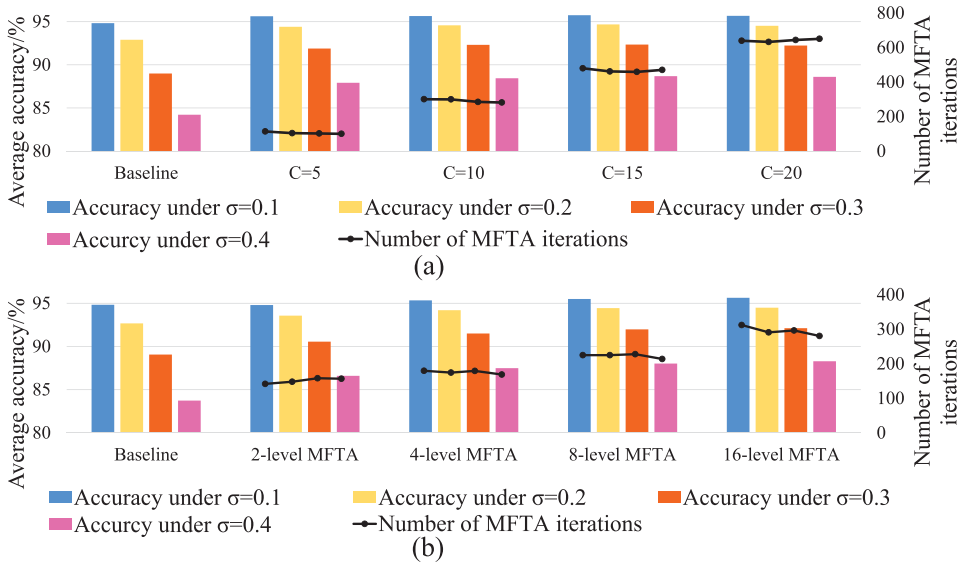


Fig. 8. Tests of (a) C and (b) $2N$ affecting the recovering inference accuracy (higher is better) and how many iterations that MFTA needs (lower is better).

majority of the destination neurons' potential. We will discuss the detailed trade-off between the benefits of TTA scheme and the accuracy degradation in Section V-C.

V. Experiments

A. Experimental Setup

We first evaluate the performance of our proposed threshold adjustment schemes based on the simulation. The simulation of the circuit components of TFSRAM is conducted in Cadence Virtuoso with 65nm CMOS process. Gaussian-distributed noises are injected into the synaptic weights to verify the performance of MFTA scheme.

In this paper, we present our investigation for the hardware on the well-studied lightweight TTFS-based SNN perceptrons on the MNIST dataset. As long as the mechanisms of deeper TTFS-based SNN models are fully investigated, more work can be conducted with the same computing principles because the basic operations are identical regardless of the model size. The pre-trained SNN model has 400 hidden-layer neurons and 10 output neurons. Its ideal inference accuracy without considering process variations is 95.4%. For the standard IF neuron module, a $2.56\mu s$ time window are adopted. The average inference energy consumption is calculated as the arithmetic mean of the energy consumption of the MNIST test set.

B. Accuracy Analysis of MFTA

MFTA scheme is to rescue the inference accuracy from circuit process variations. The efficacy of MFTA lies in the two key parameters: C and $2N$.

Dependency on C: C is the iteration number of threshold adjustment neurons should complete. A larger C means that each firing threshold is involved in more adjustments triggered by more input testing patterns. To quantitatively analyze this, the simulation injects process variations with different standard deviation $\sigma = 10\%, 20\%, 30\%$ and 40% to the synaptic weights. The standard deviation is chosen to cover the typical process variations of CMOS and emerging devices [29], [30]. For each process variation setup, we conduct 50 experiments and obtain the average inference accuracy. We set the number of available threshold levels $2N = 16$ for this set of experiments.

Fig. 8(a) presents the results. The "baseline" group refers to the inference accuracy with the process variation injection and without applying MFTA. As C increases, the iteration number of MFTA (black dots) increases in an approximately linear relation, and shows no correlation to the process variations. This trend validates that C is the knob to control the MFTA iterations. A larger C induces more iterations which imply higher compensation overhead.

The inference accuracy (bars) is boosted by applying MFTA compared to the baseline. With a larger C, MFTA leads to better accuracy given the process variation, and there is a clear saturation of accuracy recovery as C keeps increasing. Therefore, there is a sweet point that balances the iteration number and inference accuracy. For our pre-trained model, we set $C = 10$ for both satisfactory accuracy recovery and relatively small hardware costs.

Dependency on $2N$: $2N$ is the number of available levels for V_{th} candidates. A larger $2N$ value means finer granularity of MFTA scheme, projecting to more precise MFTA tuning. Yet more iterations are required to complete MFTA since the step change of the threshold voltage in each iteration is smaller.

The initial global voltage threshold of MFTA is set to 400mV based on the original circuit-level configuration, and the range of the available threshold levels is set to $260\text{mV} \sim 560\text{mV}$. Fig. 8(b) shows how the average accuracy and total MFTA iteration number change with the number of threshold levels under different process variation setups. We select the number of threshold levels to be 2 (160mV interval), 4 (80mV interval),

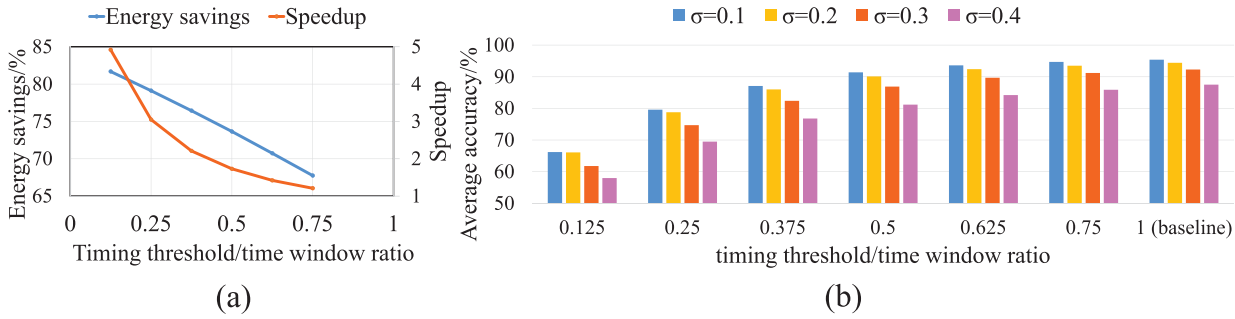


Fig. 9. (a) Energy savings and speedup vs. *timing threshold*; (b) Inference accuracy and *timing threshold* under different process variations. Notably, the x-axes of (a) and (b) are the ratio of timing threshold to time window size.

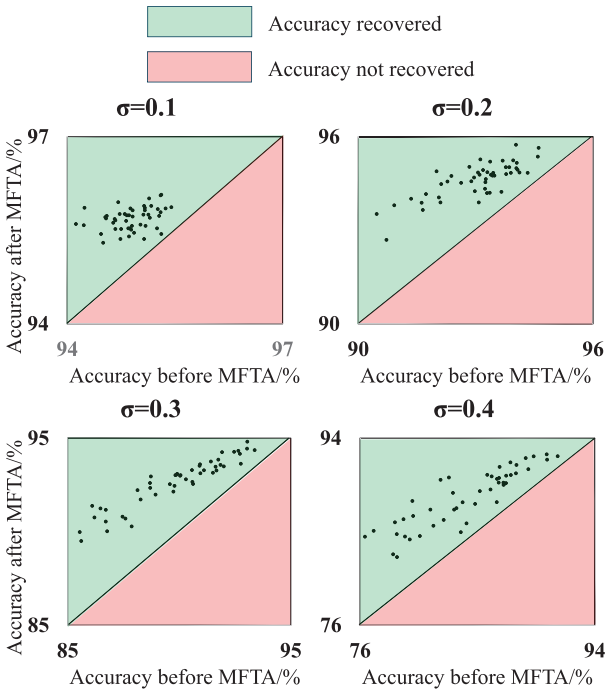
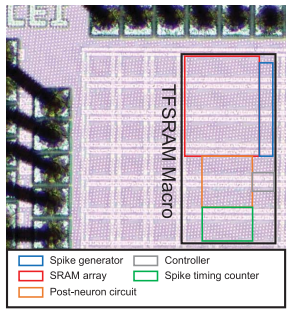


Fig. 10. Accuracy recovered by MFTA under various process variations. x- and y-axis are the percentage accuracy before and after applying MFTA, respectively.

8 (40mV interval), 16 (20mV interval). The average accuracy after applying MFTA increases with more threshold levels from the baseline accuracy. Under these four process variations σ, the accuracy after applying MFTA increases significantly when the number of threshold levels jumps from 2 to 4, whereas gains only marginally in the cases over 4. Besides, the numbers of iterations to complete MFTA in all the process variation cases increase from ~ 150 to ~ 300. This validates the trend that finer granularity of MFTA levels ($2N$) would cause more iterations. In this set of experiments, the results also show that the number of iterations to complete MFTA has no correlation to process variations. Based on these observations, we set the default $2N$ parameter as 4 levels to achieve satisfactory accuracy recovery as well as small hardware cost.

Fig. 10 shows the Monte Carlo simulation to validate the efficacy of MFTA scheme. It is the result of 50 experiments under



Technology	65nm
Voltage	1V
Clock Frequency	100MHz
Time Window	2.56μs
Die Area	0.98mm x 0.98mm
Macro Area	336μm x 106μm
Macro CIM Capacity	4Kb
Macro Power	93.1μW @ 0.65V 406.7μW @ 1.0V

Fig. 11. Die photo and summary of TFSRAM.

different process variation levels, including the accuracy before threshold adjustment (x-axis) and the accuracy after threshold adjustment (y-axis), under the aforementioned MFTA configuration. As shown, all the experiments under $\sigma = 10\% \sim 40\%$ obtain accuracy improvement. On average, TFSRAM remains at 95.3% accuracy under 10% variations, and at 94.2% accuracy under 20% variations.

C. Trade-Off in TTA

We investigate the trade-off between the energy savings and speedup of TTA scheme and the accuracy degradation under different *timing threshold* setups. Fig. 9(a) shows the trade-off between energy savings and speedup. The baseline is the proposed twin-column PE without TTA. Smaller *timing threshold* results in faster inference as well as better energy savings. Fig. 9(b) shows the accuracy degradation induced by TTA. When the *timing threshold* is half of the *time window size*, TFSRAM remains above 91.4% accuracy under 10% variations, and above 90.1% accuracy under 20% variations. Under this *timing threshold* setup, more than $1.7\times$ speedup and 68.8% energy savings can be achieved. As such, the optimal setup of "*timing threshold* is $0.5\times$ time window size = $1.28\mu s$" for TFSRAM.

D. Prototype Chip Results

Fig. 11 presents the die photo and chip summary of TFSRAM, fabricated in 65nm CMOS process. Under the condition of 1V/100MHz, the TFSRAM macro consumes 406.7μW and 98.2μW during computation and during idle state, respectively. Fig. 12 shows the measured power breakdown and area

TABLE II
COMPARISON WITH PRIOR SPIKE-BASED CIM MACROS

Work	**This work**	VLSI'19 [27]	VSLI'22 [13]	ASSCC'22 [23]	TCSI'24 [9]
Process	**65nm**	150nm	28nm	65nm	40nm
Computing cell	**8T-SRAM**	1T1R RRAM	8T-SRAM	8T-SRAM	SOT-MRAM
Voltage (V)	**1**	-	1.1	-	-
Clock frequency (MHz)	**100**	50	200	-	-
Macro size (bit)	**64x64**	256x256	64x160	32Kb	512x512
Macro area (mm^2)	**0.036**	-	0.048	0.25	~0.064
Die area (mm^2)	**0.96**	-	2.91	-	-
Input precision (bit)	**1-8**	1-8	1-8	6-12 spikes	2
Weight precision (bit)	**4**	1	1/4/8	1.5b	4
Output precision (bit)	**8**	8	8	-	-
Macro power (mW)	**0.41**	1.52	15.8	0.56	-
Energy efficiency (TOPS/W)	**249.8 (8b, 4b)**	16.9 (1b, 8b)	413.8 (4b, 4b)	290 (2.8b, 1.5b)	17.56 (2b, 4b)

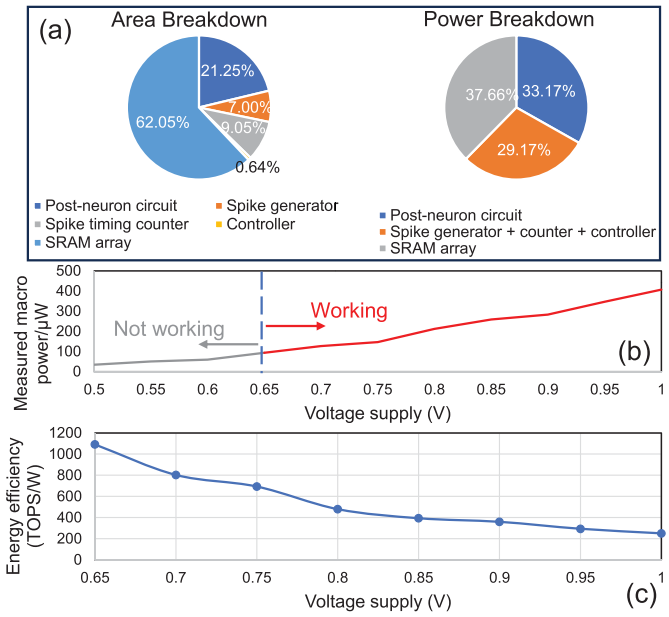


Fig. 12. (a) Area and power breakdown of TFSRAM. (b) Measured macro power under different voltage supply. (c) Measured energy efficiency under different voltage supply.

breakdown of the TFSRAM macro. The synaptic SRAM array dominates the area by 62.05%, with the 8T-SRAM cell area of $2.4\mu\text{m} \times 0.9\mu\text{m}$. The neuron circuits incur only 21.25% area. During the computation, the neuron circuits, the synaptic SRAM array, and the digital circuit components each consume about one-third of the overall power.

We also measure the macro power under the voltage supply from 0.5V to 1V under the clock frequency of 100MHz. Tested with a pre-trained 2-layer TTFS-SNN on Sklearn's 8×8 digit dataset, TFSRAM is able to generate correct outputs while applying the voltage over 0.65V, whereas temporal variations are observed in multiple cycles. Under 1V/100MHz, we select the first four post-neurons in the first layer and compare their firing times to the expected firing times from the algorithm in 50 continuous runs. The mean absolute error of the measured post-spike firing times is 7.11 time steps out of the 256-time-step time window. These dynamic errors stem from thermal noises and cycle-to-cycle variations of spiking currents that charge the membrane capacitors [21]. Table I shows the comparison results with prior spike-based CIM neuromorphic macros. TFSRAM achieves an energy efficiency of 249.8 TOPS/W and an area efficiency of 2.85 TOPS/mm^2 under 8-bit inputs and signed 4-bit weights, measured under 1V/100MHz.[2] Fig. 12(c) depicts the change in energy efficiency with different voltage supply.

Thanks to the great potential of TTFS-based SNNs and the dedicated energy-efficient neuron circuit designs, TFSRAM outperforms prior spike-based CIM macros [9], [27] in terms of energy efficiency. It achieves comparable energy efficiency with the rate-coded macro [23], whereas [23] only supports the inputs of 6-12 spikes, equivalently 2.8b input precision. TFSRAM does not demonstrate higher energy efficiency than the rate-coded macros [13] due to less advanced technology node.

VI. FUTURE WORK

Despite the fact that TFSRAM achieves a competitive improvement of energy efficiency when executing SNN inference, exploring additional methodologies that can further enhance its efficiency and expand its compatibility with diverse real-life scenarios is still ongoing. First, potential optimization lies in the sparsity of SNN models mapped to the CIM memory array. In our twin-column mapping scheme, excitatory and inhibitory synapses equally occupy the adjacent portions of the array. However, ordinary SNN models usually have imbalanced excitatory and inhibitory synapse distribution. When excitatory synapses dominate, the mapping scheme leads to significant zero redundancy of data stored in the inhibitory portion and degrades memory utilization and area efficiency. This calls for improved synapse mapping schemes to reduce the data sparsity, as well as modified neuron circuit designs to support the reorganized data layout.

Second, our proposed neuromorphic design focuses on the efficient inference of TTFS-based SNNs. Another critical advantage of temporal-coded SNNs over conventional DNNs is the capability of exploiting temporal dynamics and performing

[2]OP is defined as the number of MAC operations within each SNN inference time step. TOPS is defined as MACs×Frequency×2.

unsupervised learning rules such as spike-timing-dependency-plasticity (STDP). Compared to supervised learning algorithms, unsupervised learning rules eliminate the costs of gradient computation and data communication of backpropagation and enable local learning. While introducing these learning mechanisms to the CIM neuromorphic designs, the key challenge is how to capture the spatiotemporal information of spike events from the inference and program the synaptic arrays accordingly. An efficient solution should take full advantage of both the local learning capability (from the algorithm perspective) and the parallelism of the crossbar structure (from the hardware perspective) to avoid unnecessary data caching, conversion and transmission. We are now targeting unsupervised learning support on the basis of our neuromorphic processing engine design. The major issue to be resolved include the methodology and circuit implementation to translate the firing times of the single spikes from the post-neuron circuits to update the synaptic weight values stored in the CIM arrays. Enabling unsupervised learning in the CIM neuromorphic designs is a promising direction that will benefit the edge computing AI hardware in online learning and model fine-tuning tasks.

Third, TFSRAM is tailored for lightweight SNN inference in power-constrained and area-constrained edge computing applications. Nevertheless, the design paradigm of TFSRAM can be migrated to various memory technologies, hardware configurations and network structures. With a higher area and power budget, TFSRAM could achieve higher efficiency by boosting the computation parallelism with increasing synaptic array size, holding the potential to function as basic processing elements in large-scale neuromorphic computing systems that execute deeper SNNs for more complex cognitive tasks. However, this leads to the increasing number of neurons in each macro and thus complicates the data communication between neurons in different macros. How to deploy different partitions of the SNN workload to multiple analog spiking CIM macros and how to manage the inter-macro data communication are valuable questions that need addressing in future research.

VII. Conclusion

Neuromorphic computing hardware that harnesses the spike-based computation and compute-in-memory significantly boosts energy efficiency, paving the way towards next-generation AI hardware that pursues the capabilities of the human neural system. In this paper, we present TFSRAM, a timing-to-first-spike compute-in-memory neuromorphic processing engine with twin-column SRAM Synapses. TFSRAM features the novel twin-column synapse mapping scheme and dedicated neuron circuits. Specifically for the TTFS spiking encoding scheme and the fabricated neuromorphic processing engine, we introduce the multi-level firing threshold adjustment methodology to improve the resilience against process variations and the timing threshold adjustment methodology to exploit the advantage of event-driven SNN execution models. These two methods enable highly efficient execution of SNN inference. Measurement results show that TFSRAM achieves an energy efficiency of 249.8 TOPS/W and an area efficiency of 2.85 TOPS/mm^2 under 8-bit inputs and signed 4-bit weights.

References

[1] F. Akopyan et al., "TrueNorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 34, no. 10, pp. 1537–1557, Oct. 2015.

[2] A. Ankit, A. Sengupta, P. Panda, and K. Roy, "RESPARC: A reconfigurable and energy-efficient architecture with memristive crossbars for deep spiking neural networks," in *Proc. 54th ACM/IEEE Des. Automat. Conf. (DAC)*, 2017, pp. 1–6.

[3] B. V. Benjamin et al., "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proc. IEEE*, vol. 102, no. 5, pp. 699–716, May 2014.

[4] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *Int. J. Comput. Vis.*, vol. 113, pp. 54–66, 2015.

[5] T. Chen et al., "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *ACM SIGARCH Comput. Archit. News*, vol. 42, no. 1, pp. 269–284, 2014.

[6] 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.

[7] M. Davies et al., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan./Feb. 2018.

[8] P. U. Diehl, G. Zarrella, A. Cassidy, B. U. Pedroni, and E. Neftci, "Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware," in *Proc. IEEE Int. Conf. Rebooting Comput. (ICRC),* Piscataway, NJ, USA: IEEE Press, 2016, pp. 1–8.

[9] H. Fu et al., "DS-CIM: A 40nm asynchronous dual-spike driven, MRAM compute-in-memory macro for spiking neural network," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 71, no. 4, pp. 1638–1650, Apr. 2024.

[10] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker project," *Proc. IEEE*, vol. 102, no. 5, pp. 652–665, May 2014.

[11] W. Gerstner, R. Kempter, J. L. Van Hemmen, and H. Wagner, "A neuronal learning rule for sub-millisecond temporal coding," *Nature*, vol. 383, no. 6595, pp. 76–78, 1996.

[12] J. Göltz et al., "Fast and deep neuromorphic learning with time-to-first-spike coding," 2019, *arXiv:1911.10124*.

[13] S. Kim, S. Kim, S. Um, S. Kim, K. Kim, and H.-J. Yoo, "Neuro-CIM: A 310.4 TOPS/W neuromorphic computing-in-memory processor with low WL/BL activity and digital-analog mixed-mode neuron firing," in *Proc. IEEE Symp. VLSI Technol. Circuits (VLSI Technol. Circuits)*, Piscataway, NJ, USA: IEEE Press, 2022, pp. 38–39.

[14] H. Kwon, P. Chatarasi, M. Pellauer, A. Parashar, V. Sarkar, and T. Krishna, "Understanding reuse, performance, and hardware cost of DNN dataflow: A data-centric approach," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchit.,* Piscataway, NJ, USA: IEEE Press, 2019, pp. 754–768.

[15] Z. Li, B. Yan, and H. Li, "ReSiPE: ReRAM-based single-spiking processing-in-memory engine," in *Proc. 57th ACM/IEEE Des. Automat. Conf. (DAC),* Piscataway, NJ, USA: IEEE Press, 2020, pp. 1–6.

[16] Z. Li, Q. Zheng, B. Yan, R. Huang, B. Li, and Y. Chen, "ASTERS: Adaptable threshold spike-timing neuromorphic design with twin-column ReRAM synapses," in *Proc. 59th ACM/IEEE Des. Automat. Conf.*, 2022, pp. 1099–1104.

[17] C. Liu et al., "A spiking neuromorphic design with resistive crossbar," in *Proc. 52nd Annu. Des. Automat. Conf.*, 2015, pp. 1–6.

[18] W. Maass, "Networks of spiking neurons: The third generation of neural network models," vol. 10, no. 9, pp. 1659–1671, 1997.

[19] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC),* Piscataway, NJ, USA: IEEE Press, 2011, pp. 1–4.

[20] S. Oh et al., "Spiking neural networks with time-to-first-spike coding using TFT-type synaptic device model," *IEEE Access*, vol. 9, pp. 78098–78107, 2021.

[21] S. Park, D. Lee, and S. Yoon, "Noise-robust deep spiking neural networks with temporal information," in *Proc. 58th ACM/IEEE Des. Automat. Conf. (DAC),* Piscataway, NJ, USA: IEEE Press, 2021, pp. 373–378.

[22] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS),* Piscataway, NJ, USA: IEEE Press, 2010, pp. 1947–1950.
[23] J. Song et al., "Spike-CIM: A 290TOPS/W spike-encoding sparsity-adaptive computing-in-memory macro with differential charge-domain integrate-and-fire," in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC),* Piscataway, NJ, USA: IEEE Press, 2022, pp. 1–3.
[24] S. Thorpe, A. Delorme, and R. Van Rullen, "Spike-based strategies for rapid processing," *Neural Netw.*, vol. 14, nos. 6–7, pp. 715–725, 2001.
[25] Z. Wang et al., "Fully memristive neural networks for pattern classification with unsupervised learning," *Nature Electron.*, vol. 1, no. 2, pp. 137–145, 2018.
[26] P. Wijesinghe, A. Ankit, A. Sengupta, and K. Roy, "An all-memristor deep spiking neural computing system: A step toward realizing the low-power stochastic brain," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 5, pp. 345–358, Oct. 2018.
[27] B. Yan et al., "RRAM-based spiking nonvolatile computing-in-memory processing engine with precision-configurable *in situ* nonlinear activation," in *Proc. Symp. VLSI Technol.,* Piscataway, NJ, USA: IEEE Press, 2019, pp. T86–T87.
[28] L. Zhang, S. Zhou, T. Zhi, Z. Du, and Y. Chen, "TDSNN: From deep neural networks to deep spike neural networks with temporal-coding," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 1, 2019, pp. 1319–1326.
[29] M. Zhao et al., "Investigation of statistical retention of filamentary analog RRAM for neuromorphic computing," in *Proc. IEEE Int. Electron Devices Meeting (IEDM),* Piscataway, NJ, USA: IEEE Press, 2017, pp. 39–4.
[30] W. Zhao et al., "Rigorous extraction of process variations for 65-nm CMOS design," *IEEE Trans. Semicond. Manuf.*, vol. 22, no. 1, pp. 196–203, Feb. 2009.

Ziru Li (Graduate Student Member, IEEE) received the B.Eng. degree in electronic engineering from Tsinghua University, Beijing, China, in 2019, and the Ph.D. degree in electrical and computer engineering from Duke University, Durham, NC, USA, in 2024. His research interest includes integrated circuit design for advanced artificial intelligence algorithms.

Qilin Zheng (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees from KU Leuven and Peking University, and the Ph.D. degree in electrical and computer engineering from Duke University, in 2024. His research interests include machine learning accelerator, in-memory computing, and non-volatile memory design.

Jonathan Ku (Graduate Student Member, IEEE) received the B.S. degree in electrical engineering and computer science from National Tsing Hua University, in 2022. He is currently working toward the Ph.D. degree in electrical and computer engineering with Duke University, Durham, NC, USA. His research interests include VLSI design, hardware acceleration in cryptography and machine learning.

Brady Taylor (Graduate Student Member, IEEE) received the B.S. degree in electrical engineering from Rice University, in 2019, and the M.S. degree in electrical and computer engineering from Duke University, in 2022. He is currently working toward the Ph.D. degree with Duke University, researching mixed-signal circuits for implementing biologically-plausible learning rules. His research interests include mixed-signal VLSI, neuromorphic computing, computer architecture, and emerging nonvolatile memories for artificial intelligence.

Hai Li (Fellow, IEEE) received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, and the Ph.D. degree from the Department of Electrical and Computer Engineering. She is the Clare Boothe Luce Professor and Department Chair of the Electrical and Computer Engineering Department with Duke University. Purdue University, West Lafayette, IN, USA. Prior to joining Duke University, she worked with Qualcomm Inc., Intel Corporation, Seagate Technology, the Polytechnic Institute of New York University, and the University of Pittsburgh. Her research interests include neuromorphic circuits and systems for brain-inspired computing, machine learning acceleration and trustworthy AI, conventional and emerging memory design and architecture, and software and hardware co-design. Dr. Li served/serves as an Associate Editor for multiple IEEE and ACM journals. She was the General Chair or Technical Program Chair of numerous IEEE/ACM conferences and the Technical Program Committee Member of over 30 international conference series. She is a Distinguished Lecturer of the IEEE CAS Society (2018–2019) and a Distinguished Speaker of ACM (2017–2020). She is a recipient of the NSF Career Award, DARPA Young Faculty Award, TUM-IAS Hans Fischer Fellowship from Germany, ELATE Fellowship, Ten Year Retrospective Influential Paper Award from ICCAD, nine best paper awards, and another nine best paper nominations. Dr. Li is a Fellow of ACM.