

Digital-memory Hybrid Counter-based SRAM In-memory Computing

Siyeol Lee¹, Dasom Ahn¹, Sung Hun Jin^{3,*}, and Taehui Na^{1,2,*}

Abstract: To address the bottleneck caused by the memory wall in conventional computing systems, in-memory computing (IMC) has gained significant research interest. While the digital approach to IMC can achieve high accuracy, its energy efficiency is often lower than that of analog IMC approaches. To address this limitation, the Ternary-output Binary Neural Network-based IMC (ToBNN-IMC) has gained attention. ToBNN-IMC can achieve both high accuracy and energy efficiency through its digital nature and the use of a zero-skipping technique. However, ToBNN-IMC has been experienced only with STT-MRAM memory macro. Moreover, its accumulating strategy, which counts multiplication results using a counter with a fixed bit-width, can suffer from degraded accuracy when the number of multiply-accumulate (MAC) operations exceeds the counter's range. To overcome this issue, the proposed digital-memory hybrid counter-based SRAM IMC (hybrid SRAM-IMC) employs a counter design referred to as the scalable exponent counter, which enables accurate accumulation and direct extraction of neuron activation regardless of the number of MAC operations. By implementing this design, the hybrid SRAM-IMC achieved 96.58% inference accuracy on the MNIST dataset, 88.64% on the CIFAR-10 dataset, and 94.00% on the SVHN dataset, demonstrating accurate MAC operation without any accuracy loss, regardless of the number of accumulations.

Index terms: Binary neural network (BNN), counter, deep learning, energy efficiency, in-memory computing (IMC), low power, MAC, memory wall, MRAM, SRAM, scalable exponent counter (scalable EC)

I. INTRODUCTION

To overcome the conventional computing architecture's memory wall [1-3], in-memory computing (IMC) has been actively studied for its ability to perform multiply-accumulate (MAC) operations directly within memory macro, thereby reducing data transfer overhead [4]. IMC architectures are generally categorized into analog and digital types. Although analog IMC offers high energy efficiency, it often sacrifices computational accuracy [5-7] and area efficiency [8].

In response, recent research has investigated methods to enhance the energy efficiency of digital IMC while maintaining high accuracy. One promising approach to efficient digital IMC is the Ternary-output Binary Neural Network-based IMC (ToBNN-IMC) [9].

Like conventional BNNs, ToBNN binarizes both weights and inputs. However, it differs by encoding inputs as 0 and 1 instead of -1 and $+1$, leading to ternary multiplication output of -1 , 0, or $+1$. This ternary behavior enhances numerical diversity and improves the precision of the MAC operation while preserving the simplicity of binary input encoding.

ToBNN-IMC has previously been explored only in MRAM-based IMC architecture. However, ToBNN is also well suited for SRAM-based IMC architecture, where the memory array remains identical to that of a conventional 6T-SRAM design, owing to its use of only one bit-cell per weight, in contrast to other MRAM-based BNN-IMC designs [9].

Manuscript received Jul. 23, 2025; revised Nov. 12, 2025; accepted Jan. 20, 2026

¹Department of Intelligent Semiconductor Engineering, Incheon National University, Incheon 22012, South Korea

²Department of Electronics Engineering, Incheon National University, Incheon 22012, South Korea

³Department of Information Display, Kyung Hee University, Seoul 02447, South Korea

*E-mail : jinsh@khu.ac.kr, taehui.na@inu.ac.kr

To accumulate multiplication results, ToBNN-IMC systems employ counters. Although the area overhead of the digital counter is over ten times smaller than that of the ADC used in analog IMC, it is still more than ten times larger than the overhead of the Sense Amplifier (SA) [9-11,15]. Reducing counter bit-width can alleviate this issue, but it risks degrading MAC accuracy, making bit-width optimization a complex task dependent on model and dataset variability. Thus, this paper proposes a digital-memory hybrid counter that repurposes a portion of the SSRAM cells, which would otherwise be used solely for weight storage, to store intermediate computation results and support neuron-activation extraction. And by utilizing this method, the proposed digital-memory hybrid counter-based SRAM IMC can utilize low bit-width counter while eliminating the degradation of MAC accuracy.

In this paper, Section II first introduces ToBNN, the SRAM-based ToBNN-IMC, the counter architecture used in ToBNN-IMC, and the direct extraction method for neuron activation. Section III then proposes the core of the presented IMC architecture, the exponent counter, and introduces the scalable exponent counter structure, which enables simplified and scalable operation using registers. In Section IV, we propose the digital-memory hybrid counter-based SRAM IMC architecture by applying the scalable exponent counter to the SRAM-based ToBNN-IMC and discuss its circuit design and SRAM memory mapping strategy. Subsequently, in Section V, we present software-level simulations, post-layout simulations, and performance metrics derived from throughput calculation. In Section VI, we compare the proposed architecture with prior works, and finally, Section VII concludes the paper.

II. PRELIMINARY

1. Binarization and Optimization of ToBNN

As described in the introduction, ToBNN differs from conventional BNNs in the binarization of inputs and weights. Conventional BNNs binarize both input and weight values as shown in Eq. (1), where x^b is the binarized value and x is real value.

$$x^b = \text{sign}(x) = \begin{cases} +1, & \text{if } x \geq 0. \\ -1 & \text{otherwise.} \end{cases} \quad (1)$$

However, in ToBNN, only the weight binarization follows Eq. (1), while the input is binarized using a step func-

tion, as shown in Eq. (2) below.

$$x^b = \text{step}(x) = \begin{cases} 1, & \text{if } x > 0, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Due to the different binarization methods for input and weight, the multiplication results between an input and a weight can take one of three values. Unlike conventional BNNs, this increase in possible output states leads to improved inference accuracy in ToBNN.

In addition, since the input is binarized to 1 or 0, the multiplication operation can be simplified. when the input is 1, the binarized weight directly becomes the multiplication result, and when the input is 0, the output is 0. Moreover, because the zero output does not affect the accumulation, the computation can be skipped when the input is 0. This technique, referred to as zero-skipping, can significantly reduce the computational power required for the neural network.

2. SRAM-based ToBNN-IMC

The SRAM-based ToBNN-IMC utilizes a conventional 6T-SRAM macro, where the weights are stored in the SRAM bit-cells, and the input is directly applied to word lines (WLs) through the WL decoder. Since the inputs are binarized to either 1 or 0, multiplication can be implemented by sequentially activating the WLs and reading the corresponding bit-cells using the SA when the input is 1. In the case of input 0, the corresponding WL read operation is skipped using the zero-skipping technique, thereby minimizing the power and latency of the MAC operation. The write driver (WD) is used to write weight values to the SRAM array. However, it is not used during neural network inference computation. The MUX is connected to multiple SRAM column bit-lines (BLs) and selects a BL to connect to the SA and WD. This configuration increases the number of weights that a single counter can handle. The shared controller controls the overall peripherals, and the simultaneous operation of these peripherals enables parallel MAC operation.

3. Up/Down Counter in SRAM-based ToBNN-IMC

Due to the serial activation of WLs, the multiplication results are generated sequentially. Accordingly, the accumulation result must also be stored and updated in a sequential manner, which is most efficiently implemented using a counter. The counter is implemented after the SA,

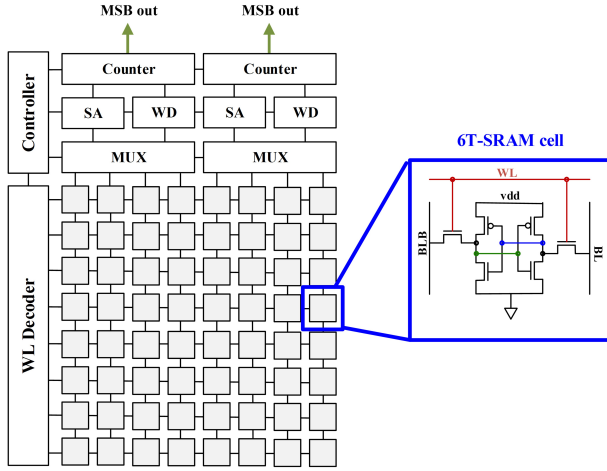


Fig. 1. Overall architecture of SRAM-based ToBNN-IMC.

and the read weights are propagated to the counter input. Since the SA output becomes either +1 or 0, corresponding to multiplication results of +1 or -1, the counter must support both increment and decrement operations. Therefore, ToBNN-IMC employs an up/down counter. The overall architecture of SRAM-based ToBNN-IMC is shown in Fig. 1.

4. Counter Initialization Strategy of ToBNN-IMC

In ToBNN-IMC, the neuron's non-linear activation is implemented by the counter macro, eliminating extra activation hardware [9]. Since the activation is a step function, its output can be taken directly from the MSB of an N-bit counter.

The counter is initialized to the midpoint value (MSB = 0, all lower bits = 1), which is defined as zero. With this configuration, the MSB serves as the decision bit. After accumulation, MSB = 1 (count > midpoint) maps to output 1, whereas MSB = 0 (count ≤ midpoint) maps to output 0.

5. Limitation of Conventional Counter in ToBNN-IMC

Since a digital counter has a fixed bit-width, its counting range is also limited. If an increment or decrement operation exceeds this range, overflow or underflow can occur, resulting in a loss of the accumulated value. To address this issue, the counter's bit-width must be sufficiently large to support the target neural network application in the IMC system. However, increasing bit-width proportionally increases the number of FFs and the associated digital logics. Consequently, to support a wide range

of neural network models in ToBNN-IMC, the counter's bit-width must be designed to accommodate the largest expected model. This leads to significant area and energy overheads due to the enlarged counter structure.

III. THE EXPONENT COUNTER

To minimize the area and energy overheads caused by the large-bit-width counter required to support various NN models in ToBNN-IMC, this paper proposes a modified counter structure, termed the Scalable Exponent Counter (scalable EC), which is discussed below.

1. Preliminaries of the Exponent Counter (EC)

Consider an up/down counter initialized to zero, operating within a symmetric range of $\pm N$. When the count value exceeds the defined range, the counter generates a plus carry or minus carry signal and resets its count to zero. Each carry represents a value weighted by $(N+1)$ relative to the up/down input, corresponding to the counter's range.

This counter design is referred to as an Accumulative Carry Counter (ACC) due to its ability to accumulate inputs and output carries. When M such ACCs are connected in a cascaded manner, where the carry output of a lower-level counter serves as the input to the next higher-level counter, the final counter receives an input weighted by $(N+1)^{M-1}$ relative to the original input. This cascaded chain of ACCs is termed EC, as mentioned earlier, due to the exponential scaling of the propagated carry through successive stages.

An illustration of the ACC and EC is shown in Fig. 2. After the counting process is complete, each ACC retains a residual count (RC) after propagating any carries to the next ACC. The overall accumulated value of the EC can be obtained by multiplying each ACC's RC by its corresponding exponential weight and summing across all levels. Eq. (3) describes this operation.

$$\text{Accumulated value} = \sum_{i=0}^{M-1} RC_i \times (N+1)^i. \quad (3)$$

In ToBNN-IMC, however, to extract the neuron activation output from the counter, the required information is not the full accumulated value, but simply whether the accumulated value is greater than zero or not. Accordingly, Eq. (3) does not need to be computed to determine the neuron activation in the EC.

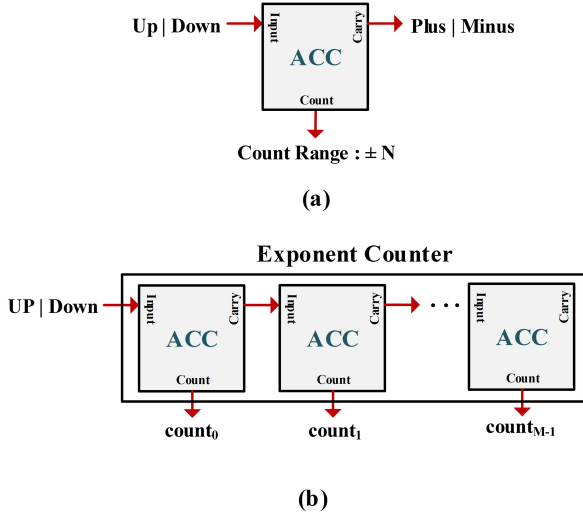


Fig. 2. (a) Diagram of the accumulative carry counter (ACC). (b) Diagram of the exponent counter (EC) composed with multiple ACCs.

Algorithm 1: Determining neuron activation from ACC's RC values.

1. **Input:** $RC_{M-1}, RC_{M-2}, \dots, RC_0$ from M cascaded ACCs
 2. **Output:** Neuron activation A (0 or 1)
 3. **for** $i = (M - 1)$ **to** 1 **do**
 4. **if** $RC_i > 0$ **then**
 5. $A \leftarrow 1$; **return** A ;
 6. **else if** $RC_i < 0$ **then**
 7. $A \leftarrow 0$; **return** A ;
 8. **end for**
 9. $A \leftarrow 0$; **return** A ;
-

To decide whether the accumulated value is greater than zero, the RC_i are compared with zero in sequence. Since the exponential weighting makes a higher-level RC dominate lower-level RCs, if the comparison at a higher-level yields $RC > 0$ or $RC < 0$, the neuron activation is immediately determined as 1 or 0, and comparisons at lower levels are skipped. A lower-level comparison is performed only when the higher-level RC are equal zero. If all RCs are equal to zero, the activation is 0 by the nature of the step function. The EC's activation decision method can be found in Algorithm 1.

2. The Scalable EC

A scalable implementation of the EC can be realized by reusing a single ACC. In this design, two registers are uti-

lized: Carry Register for storing the carry values and RC register for storing the RCs of the ACC. As described in Algorithm 1, since only the sign of the RC is required, the MSB of the binary counted value, referred to as the RC-sign, is stored in the register. This architecture is termed the scalable EC and is illustrated in Fig. 3(a).

The operation of the scalable EC can be divided into three cycles. In the first cycle, the ACC receives up/down signals and performs counting operations, accordingly, as shown in Fig. 3(b). Whenever a carry is generated during this process, the counting is paused, and the carry is stored in the carry register for use in the next cycle, after which the ACC is reset to zero. After all inputs have been processed, the RCsign is stored in the RC register, and the ACC is again reset in preparation for the next accumulation, as illustrated in Fig. 3(c).

In the second cycle, the ACC begins accumulation by receiving the carry values stored during the first cycle as input, as shown in Fig. 3(d). Any new carries generated during this accumulation overwrite the obsolete carries from the previous counting. Once the accumulation is complete, the RCsign is again stored in the RC register. The RCsigns are saved in the latest order in the register so that they can be checked in that order when extracting the neuron activation. After storing the RCsign, ACC is reset. This cycle is repeated until no further carry is generated.

In the final cycle, the neuron activation is determined based on the sequence of RCsigns stored in the RC register, as illustrated in Fig. 3(e). The RCsigns are examined in the latest order, and the first non-zero RC encountered is taken as the neuron activation.

The timing diagram for the scalable EC with ACC counting range of ± 3 , write-enable (WE) of the Carry register, and the RC register is shown in Fig. 4. The Carry register's read address is synchronized to the ACC input signal, and the write addresses of both registers are synchronized to their respective WE signals.

IV. DIGITAL-MEMORY HYBRID COUNTER-BASED SRAM IMC

1. Overall Architecture of Digital-memory Hybrid Counter-based SRAM IMC

The scalable EC can be integrated into ToBNN-IMC by replacing its registers with reserved memory cells. Therefore, by modifying the SRAM-based ToBNN-IMC to use the SRAM memory array as a replacement of registers, the scalable EC functionality can be implemented, realiz-

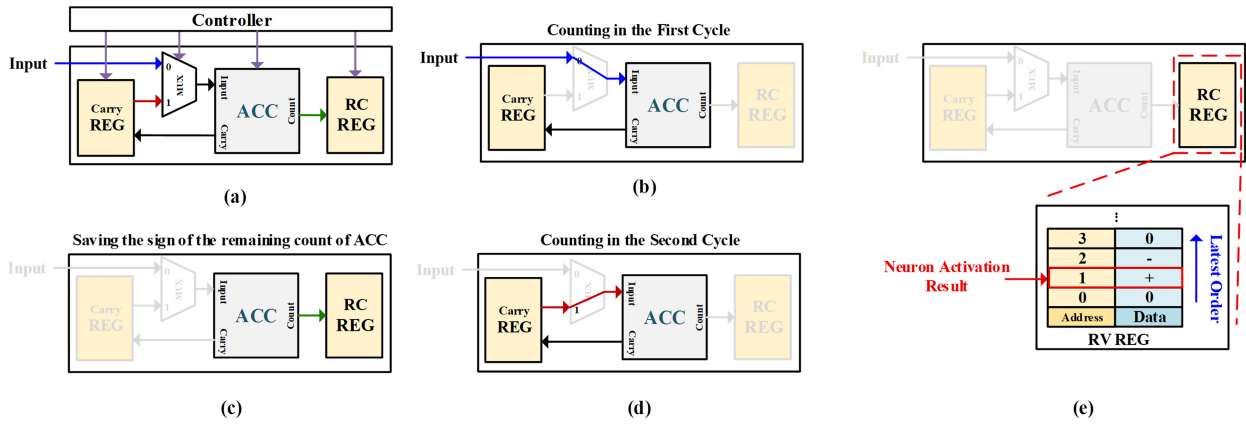


Fig. 3. (a) Circuit diagram of scalable EC. (b) Scalable EC's counting operation in the first cycle. (c) Scalable EC's RCsign saving operation after the counting. (d) Scalable EC's counting operation in the second cycle. (e) Determination of neuron activation in the final cycle.

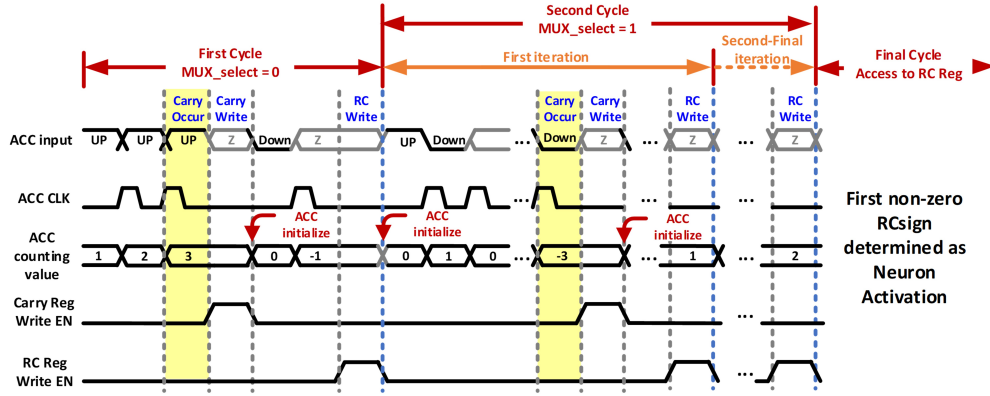


Fig. 4. Timing diagram of scalable EC with the ACC, counting range of ± 3 .

ing the digital-memory hybrid counter-based SRAM IMC (hybrid SRAM-IMC). Specifically, the up/down counter of SRAM-based ToBNN-IMC is replaced with the ACC, and a register write circuit (RW) is added, connected to the ACC and WD to determine the SRAM write values. All peripheral and digital circuits, including scalable EC, are managed by a single controller, as in the SRAM-based ToBNN-IMC. The circuit diagram of hybrid SRAM-IMC is shown in Fig. 5.

In this architecture, a subset of SRAM bit-cells per ACC is repurposed as registers for the corresponding scalable EC. Although various strategies can be adopted for SRAM mapping to serve as registers, this work employs a dedicated column-based allocation strategy, where a specific SRAM column is reserved exclusively for register use. The carry outputs of the scalable EC are sequentially stored at the beginning of the corresponding SRAM column, and the RCsigns are stored following the carry section. The operation of the digital-memory hybrid counter-

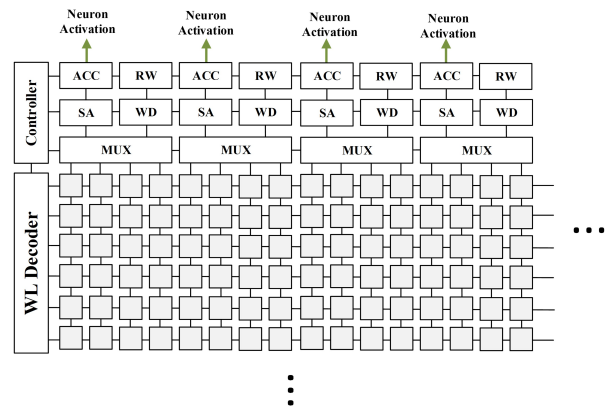


Fig. 5. Circuit diagram of hybrid SRAM-IMC.

based SRAM IMC follows the same three-cycle procedure as the scalable EC, with the addition of memory read and write operations.

During the first cycle of the proposed hybrid SRAM-IMC, WLs are activated only when the input is 1, thereby

enabling multiplication operations. In this cycle, each ACC updates its count according to the multiplication result between the input and the weight. If a carry is generated during counting, counting is paused, and the generated carry is sequentially stored in the SRAM array. And after all the multiplied results are fed to the ACC, the RCsign is stored in the SRAM array at the address following the carry entry. The controller stores the address of the last-written carry and RCsign entries, so it can correctly point to the write address for the next carry or RCsign value.

In the second cycle, since all of the carries should be accumulated into the counter, the zero-skipping operation is not adopted. Thus, all WLs are sequentially activated, and the stored carry values from the previous cycle are fed back into the ACCs. Like the first cycle, if a carry is generated during the accumulation of the last cycle's carry values, those values are overwritten sequentially in the SRAM array. After all the carries are accumulated, the current RCsign is written after the carry entries. The previously stored RCsigns are then read and rewritten to follow the current RCsign. This cycle is repeated until no further carry is generated.

In the last cycle, each ACC reads the RCsigns stored in the SRAM array to generate the neuron activation output.

The overall operation of the hybrid SRAM-IMC follows the Scalable EC timing diagram in Fig. 4. However, its specific behavior differs due to the hybrid SRAM-IMC's counter structure and control logic.

2. Carry Encoding and End-Flag Mechanism

Since multiple ACCs are controlled by a shared single controller, the ACCs cannot operate independently. Therefore, even if one ACC generates a carry, the other ACCs are still required to write a value to the SRAM memory, regardless of whether they generated a carry. In such cases, a neutral value should be written to the memory instead of plus or minus carry.

Furthermore, since the amount of carry is not fixed and depends on the neural network size, the combination of weight values, and the input data, the controller in the subsequent counting cannot determine at which WL address to stop reading the carry value stored from the previous accumulation. Therefore, an end-flag is required to indicate when to terminate reading the carry values.

These two additional states that need to be written in the SRAM memory make a single bit-cell insufficient to

represent four possible states. Therefore, two SRAM bit-cells are allocated to represent the four states: 11 for plus carry, 00 for minus carry, 01 for neutral value, and 10 for the end-flag.

With this approach, when the ACC reads a neutral value or end-flag, it receives one plus and one minus inputs, effectively canceling each other out and leaving no impact on the accumulated value. Additionally, the controller checks the two bit-cells, and if their values are 10, it can stop the accumulation process.

3. Design of ACC circuit

The counter utilized in the ACC employs a 3-bit ripple carry adder (RCA)-based design, as shown in Fig. 6 [12]. The ACC is initialized to its midpoint value, which is interpreted as zero, similar to the counter used in the SRAM-based ToBNN-IMC. To support symmetric range and carry output, two AND gates and one OR gate are used to generate the Carry signal and carry occurrence (Occ) signal. These Carry and Occ signals are sent to the RW, which determines the value to be written to the SRAM memory based on them. Although the RW control signal is shared across all ACCs, the different counter outputs allow each ACC to write distinct values.

As discussed in Section III.2, four possible states are written to memory. When reading these two bit-cells sequentially, the data volume doubles, increasing the number of carry events. To address this, the ACC in the hybrid

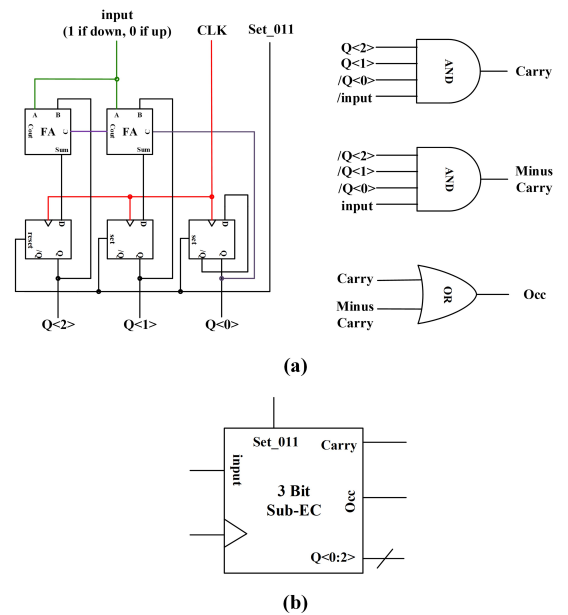


Fig. 6. (a) Circuit of 3-bit ACC. (b) Symbol of 3-bit ACC.

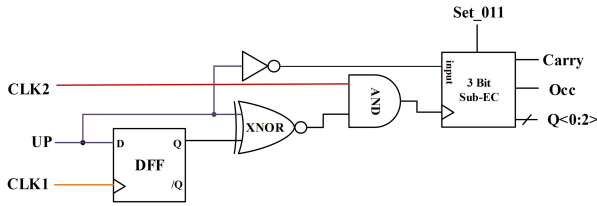


Fig. 7. Full ACC circuit for the hybrid SRAM-IMC.

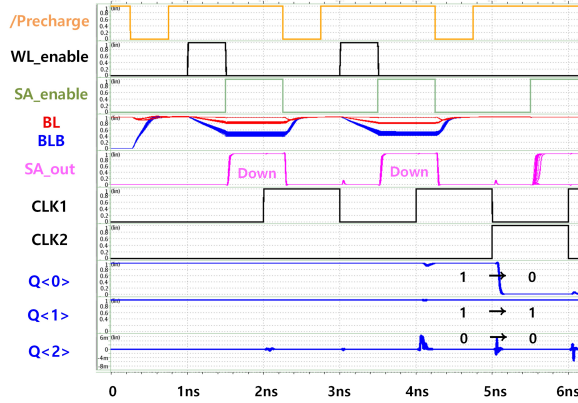


Fig. 8. Timing diagram of SA and Full ACC circuit.

SRAM-IMC is designed to read and accumulate the values of both bit-cells simultaneously in a single operation.

To achieve this, the ACC operates half the speed of SA. This allows the ACC to process two SRAM bit-cell values at once. Additionally, the input signal to the counter is latched using a DFF and compared to the previous counting cycle's input using an XNOR gate. If the input values differ between consecutive clock cycles, the clock signal is held low, preventing unnecessary counter updates. The complete circuit implementation of the ACC is illustrated in Fig. 7, and the timing diagram of SA and ACC is illustrated in Fig. 8.

4. RCsign Utilizing Strategy

In the final cycle of hybrid SRAM-IMC, the neuron activation is derived by the ACC reading the RCsigns stored after the end-flag. As in the previous two cycles, the ACC is initialized to its midpoint value, interpreted as a zero, and sequentially reads each RCsign. If the counted value after processing a specific RCsign is non-zero, the sign of the counted value extracted from the MSB is taken as the neuron activation. If the counted value is zero, the ACC continues to read the next RCsign until it encounters a non-zero result.

However, a challenge arises due to the shared controller

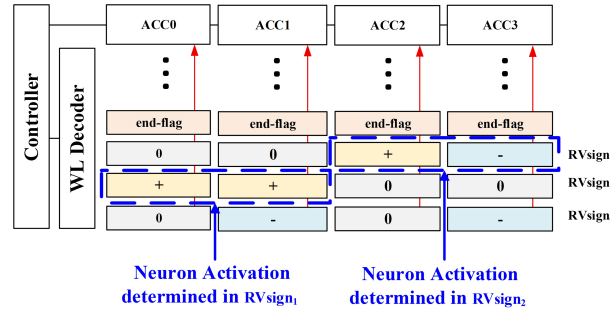


Fig. 9. The determinant RCsign's address difference between the ACCs.

and WL decoder across the ACCs. Because the WL address of the first non-zero RCsign can differ among multiple ACCs, the counter cannot determine the neuron activation at the same RCsign address for all ACCs. For example, as shown in the simplified hybrid SRAM-IMC diagram in Fig. 9, the neuron activation of ACC0 and ACC1 is determined at the RCsign1 address, while that of ACC2 and ACC3 is determined at RCsign2. This address difference makes it difficult to determine the neuron activation of multiple ACC through serial memory reading.

To mitigate this issue, the activation-decision point must be aligned across all ACCs. This can be achieved by sequentially reading the RCsigns and updating earlier RCsigns as it progresses. Since each ACC holds the same number of RCsigns, this rolling update ensures that, when the earliest RCsign is finally read, all ACCs yield a correct activation result.

This algorithm can be realized by storing RCsigns as 4-bit values. As each RCsign is read, the MSB 3 bits of the next RCsign are overwritten with updated data. This process continues until the earliest RCsign, and by reading the modified RCsign in the final counting step, all ACCs can simultaneously determine the neuron activation.

The RCsign is initially encoded as 1111 for a positive RCsign, 0000 for a negative RCsign, and 1010 for a zero RCsign. These values are written into the SRAM memory during the first and second cycles and are accumulated by the ACC during the final cycle of the hybrid SRAM-IMC. Each of the 4 bits in an RCsign is treated as an individual up or down signal.

During the reading and overwriting processes, if the ACC's MSB becomes 1 after accumulating the current RCsign, the RW circuit writes 111 to the three MSBs of the next RCsign. If the MSB is 0, it writes 101, as summarized in Algorithm 2.

Algorithm 2: Determining Neuron Activation from 4-bit RC values.

1. **Input:** $RCsign_{M-1}, RCsign_{M-2}, \dots, RCsign_0$ (4-bit values)
 2. where: $1111 \rightarrow +4, 0000 \rightarrow -4, 1010 \rightarrow 0, \dots$;
 3. **Output:** Neuron activation A (0 or 1);
 4. **for** $i = (M - 1)$ **to** 0 **do**
 5. Initialize $ACC \leftarrow 0$;
 6. $ACC \leftarrow ACC + RCsign_i$;
 7. **if** $i > 0$ **then**
 8. **if** $ACC > 0$ **then**
 9. overwrite
 $RCsign_{i-1}[MSB : MSB - 2] \leftarrow 111$;
 10. **else**
 11. overwrite
 $RCsign_{i-1}[MSB : MSB - 2] \leftarrow 101$;
 12. **end**
 13. **if** $ACC > 0$ **then**
 14. $A \leftarrow 1$;
 15. **else**
 16. $A \leftarrow 0$;
 17. **return** A ;
-

This algorithm can be explained with a simple example. Suppose two $RCsign$ values following the end-flag are initially stored as 1111 (positive) and 0000 (negative). After reading the first $RCsign$, the ACC 's MSB becomes 1, causing the second $RCsign$ to be updated to 1110. When the updated second $RCsign$ is read, the counter accumulates three up counts and one down count, resulting in $MSB = 1$, and the neuron activation is correctly inferred as 1.

As another example, if the stored $RCsign$ s are 1010 (zero) and 0000 (negative), the ACC 's MSB after accumulating the first $RCsign$ remains 0. Therefore, the next $RCsign$ is updated to 1010, and when read, the counter still holds $MSB = 0$, resulting in a correctly inferred neuron activation output of 0.

5. Register Mapping in the Hybrid SRAM-IMC

Fig. 10 shows the proposed hybrid SRAM-IMC, which uses one column of memory for the register of scalable EC, out of 4 columns per MUX. The blue and yellow memory cells represent the values stored in the SRAM memory for the register use, while the gray memory cells

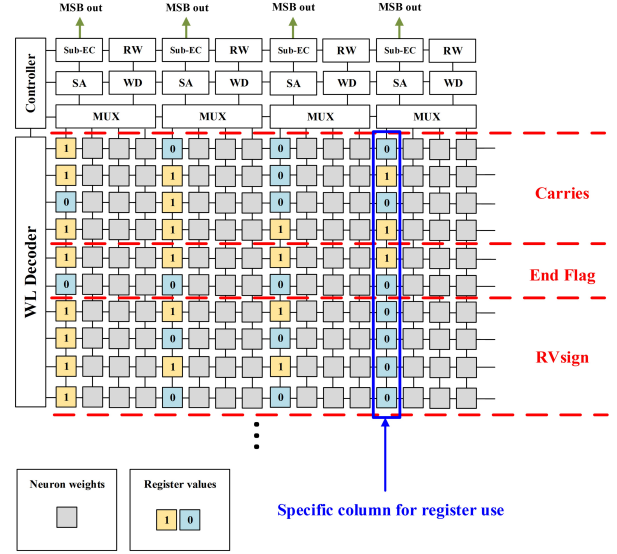


Fig. 10. Memory mapping of hybrid SRAM-IMC.

indicate the neural network weights stored in the memory array. The figure illustrates two scalable EC carries, each consisting of two bit-cells, an end flag also composed of two bit-cells, and a single $RCsign$ value represented by four bit-cells.

V. SIMULATION

Software-level simulations were conducted on the proposed hybrid SRAM-IMC, achieving an accuracy of 96.58% on the MNIST dataset using a multi-layer perceptron (MLP) model. When evaluated with a ResNet-34 on CIFAR-10, the system achieved an accuracy of 88.64%. Furthermore, using the ResNet-34 model and the SVHN dataset, the system reached an accuracy of 94.00%.

The proposed hybrid SRAM-IMC circuit is laid out with 256 bit-cells per column, and four columns are fed into each MUX. There are 64 such MUX sets, resulting in a total SRAM memory capacity of 64 kb. The design was implemented using industry-compatible 28 nm process parameter, as shown in Fig. 11. Regarding area composition, the WL decoder accounts for 8.95%, the read/write and ACC circuit for 9.63%, and the SRAM array for 81.42%. The post-layout simulations were conducted under the condition of a 25°C ambient temperature and a 1.0 V supply voltage.

To calculate the throughput (TOPS) and energy efficiency (TOPS/W) of the proposed circuit, the average time and power consumption due to memory read and write operations were analyzed during the accumulation of 784 in-

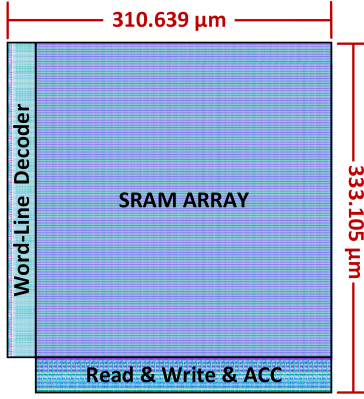


Fig. 11. Circuit layout of hybrid SRAM-IMC.

put*weight multiplications based on the MNIST dataset and a trained MLP model. Under these conditions, the average power consumption for a 2-bit memory read and counting operation at CLK1 = 500 MHz was measured at $82.39 \mu\text{W}$, while the average power consumption for a 2-bit memory write was $22.26 \mu\text{W}$. Additionally, since 86.155% of multiplications are skipped in the MLP model trained on the MNIST dataset [9], this results in an average of 104.5448 weight reads during 784 MAC operations. And during these MAC operations, an average of 97.113 bit-cell memory write operations were performed.

Because the data written to the memory during MAC operations must be read back, the memory read and counting operations need to be performed for a total of $104.5448 + 97.113$ bits. Therefore, the average time required for 784 MAC operations can be calculated as expressed in the equation below.

$$\begin{aligned} t_{\text{avg}} &= t_{\text{count}} + t_{\text{write}} \\ &= (104.5448 + 97.113) \times 2 \text{ ns} + 97.113 \times 2 \text{ ns} \\ &= 597.54 \text{ ns}. \end{aligned} \quad (4)$$

Power consumption is calculated similarly, as shown in the following equation. Notably, since the measured power corresponds to 2-bit memory operations, it has been normalized by dividing by 2.

$$P_{\text{avg}} = \frac{(P_{\text{count}} + P_{\text{write}}) \times \text{Number}_{\text{ACC}}}{\text{Number}_{\text{MAC}}} = 779.83 \mu\text{W}. \quad (5)$$

Accordingly, the TOPS for all 64 ACCs can be calculated as presented in the following equation, where TOPS/W is simply obtained by dividing the TOPS by power consumption.

$$\text{TOPS} = 2_{\text{operation}} \times \left(\frac{t_{\text{avg}}}{784} \right)^{-1} \times 64 \div 10^{12}$$

$$\begin{aligned} &= 2 \times \left(\frac{597.54 \times 10^{-9}}{784} \right)^{-1} \times 64 \div 10^{12} \\ &= 0.168. \end{aligned} \quad (6)$$

VI. COMPARISON WITH PRIOR WORKS

A comparison with prior works was conducted, as summarized in Table 1. The proposed hybrid SRAM-IMC achieves substantial gains in throughput, energy efficiency, and compute density over prior work. While its normalized performance is lower than the ISSCC'22 design [13], the task-level impact of bit-precision is modest. On ResNet-34/CIFAR-10, INT8/INT8 and 1-bit ToBNN

Table 1. Comparison with two architectures

	ISSCC'22 [13]	TCAS-II'23 [9]	This work
Memory type	6T-SRAM	STT-MRAM	6T-SRAM
Technology	28 nm		
Macro area [mm ²]	0.03	0.093	0.102
Array Size [kb]	32	1024	64
Precision [Input/weight]	Int-8b / Int-8b	Step-1b / Sign-1b	Step-1b / Sign-1b
Throughput [TOPS]	0.0055	0.017	0.168
Throughput [TBOPS]*	0.352		
Energy efficiency [TOPS/W]	27.38	58.69	215.36
Normalized energy efficiency [TBOPS/W]	1752.32		
Compute density [TOPS/mm ²]	0.183	0.183	1.631
Normalized compute density [TBOPS/mm ²]	11.712		
Need of external accumulator	O	X	X
Inference accuracy	ResNet-34 CIFAR-10: 92.05% [14]**	MLP MNIST: 92.12%	MLP MNIST: 96.58% Resnet-18 CIFAR-10: 82.85% Resnet-34 CIFAR-10: 88.64% Resnet-34 SVHN: 94.00%

* Tera bit-operation per second = TOPS * Precision_{input} * Precision_{weight}

** The reported accuracy corresponds to a model quantized to the same precision in the cited work.

differ by only 3.41% in inference accuracy [14]. Consequently, the precision-induced performance gap between the two designs is relatively small. Specifically, compared with the SRAM-based IMC reported at ISSCC'22 [13], the proposed design delivers 7.87x higher energy efficiency, 8.91x higher compute density and 0.47x normalized throughput. Relative to the ToBNN-based design in TCASII'23 [9], it provides 3.67x higher energy efficiency and likewise an 8.91x increase in compute density.

The ISSCC'22 architecture is designed to accumulate up to 32 multiplication results within the IMC macro. When more than 32 multiplication results need to be accumulated, an external accumulator outside the IMC macro is required to serially accumulate these values. In the case of the TCAS-II'23 architecture, no external accumulator is needed. However, due to the limited bit-width of the counter, the accuracy significantly degrades when accumulating a large number of multiplication results. In contrast, the proposed hybrid SRAM-IMC does not require an external accumulator. Furthermore, because of the scalable EC, it can produce accurate MAC results regardless of the number of accumulated multiplications.

VII. CONCLUSION

In this work, we propose a digital-memory hybrid counter-based SRAM IMC that can provide accurate MAC results regardless of the number of multiplication results to be accumulated. Compared with state-of-the-art designs, the proposed architecture achieves higher energy efficiency and compute density. Although its normalized performance is lower than prior SRAM-IMC, the accuracy loss attributable to precision is modest, indicating that the effective performance gap is relatively small. Unlike prior 6T-SRAM-based IMC architecture, it eliminates the need for an external accumulator, which can introduce additional computing overhead. And when compared to prior MRAM-based ToBNN-IMC design, it demonstrates higher accuracy and is expected to achieve significantly higher accuracy as the number of MAC operations increases.

ACKNOWLEDGMENTS

This work was supported by the Post-Doctor Research Program (2021) through Incheon National University (INU), Incheon, Republic of Korea. The EDA tool was supported by the IC Design Education Center (IDEC),

South Korea.

REFERENCES

- [1] N. Srivastava, A. K. Rajput, M. Pattanaik, and G. Kaushal, "An energy-efficient and robust 10T SRAM-based in-memory computing architecture," in *Proc. of 2023 36th International Conference on VLSI Design and 2023 22nd International Conference on Embedded Systems (VLSID)*, Hyderabad, India, pp. 133-138, 2023.
- [2] S. Ananthanarayanan, B. S. Reniwal, and A. Upadhyay, "Design and analysis of multibit multiply and accumulate (MAC) unit: An analog in-memory computing approach," in *Proc. of 2023 36th International Conference on VLSI Design and 2023 22nd International Conference on Embedded Systems (VLSID)*, Hyderabad, India, pp. 109-114, 2023.
- [3] Z. Wang, C. Liu, and T. Nowatzki, "Infinity Stream: Enabling transparent and automated in-memory computing," *IEEE Computer Architecture Letters*, vol. 21, no. 2, pp. 85-88, Jul.-Dec. 2022.
- [4] A. Kneip and D. Bol, "Impact of analog non-idealities on the design space of 6T-SRAM current-domain dot-product operators for in-memory computing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 5, pp. 1931-1944, May 2021.
- [5] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-tile 2.4-Mb in-memory-computing CNN accelerator employing charge-domain compute," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 6, pp. 1789-1799, Jun. 2019.
- [6] D. Kim, Y. Jang, T. Kim, and J. Park, "BiMDiM: Area efficient bi-directional MRAM digital in-memory computing," in *Proc. of 2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, Incheon, Republic of Korea, pp. 74-77, 2022.
- [7] M. Gupta, S. Cosemans, P. Debacker, and W. Dehaene, "A 2Mbit digital in-memory computing matrix-vector multiplier for DNN inference supporting flexible bit precision and matrix size achieving 612 binary TOPS/W," in *Proc. of ESSCIRC 2023 - IEEE 49th European Solid State Circuits Conf. (ESSCIRC)*, Lisbon, Portugal, pp. 417-420, 2023.
- [8] H. Jiang, S. Huang, W. Li, and S. Yu, "ENNA: An efficient neural network accelerator design based on ADC-free compute-in-memory subarrays," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, no. 1, pp. 353-363, Jan. 2023.
- [9] T. Na, "Ternary output binary neural network with zero-skipping for MRAM-based digital in-memory computing," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 7, pp. 2655-2659, Jul. 2023.

- [10] J. Chen, F. Mei, M. Liu, Y. Chen, and J. Wu, "A 32GS/s 7bit TI-SAR ADC in 28nm for 32Gb/s ADC-based SerDes receiver," in *Proc. of 2023 IEEE 15th International Conference on ASIC (ASICON)*, Nanjing, China, pp. 1-4, 2023.
- [11] D. Ahn, S. Ahn, and T. Na, "Area-optimized and reliable computing-in-memory platform based on STT-MRAM," *IEEE Journal of Semiconductor Technology and Science*, vol. 25, no. 1, pp. 56-65, Feb. 2025.
- [12] S. Lee, G. Lee, S. Ahn, and T. Na, "Analysis of low area digital up/down clipping counter for digital in-memory computing," *IEEE Access*, vol. 13, pp. 32808-32818, 2025.
- [13] B. Yan, J.-L. Hsu, P.-C. Yu, C.-C. Lee, Y. Zhang, *et al.*, "A 1.041-Mb/mm² 27.38-TOPS/W signed-INT8 dynamic-logic-based ADC-less SRAM compute-in-memory macro in 28 nm with reconfigurable bitwise operation for AI and embedded applications," in *Proc. of 2022 IEEE International Solid-State Circuits Conference (ISSCC)*, 2022.
- [14] A. Porsia, G. Perlo, A. Ruospo, and E. Sanchez, "On the resilience of INT8 quantized neural networks on low-power RISC-V devices," in *Proc. of 2025 55th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, Naples, Italy, pp. 119-122, 2025.
- [15] D. Ahn, S. Ahn, S. Lee, and T. Na, "Area-efficient/low-power MRAM-PIM based on crossbar array utilizing ternary output," *IEEE Transactions on Magnetics*, vol. 61, no. 12, Art. no. 3401909, Dec. 2025.



Sung Hun Jin received his Ph.D. degree in electrical engineering & computer science from Seoul National University, Seoul, Republic of Korea, in 2006. From 2006 to 2009, he was with Samsung Electronics Co., Ltd., Giheung, Republic of Korea. From 2009 to 2013, he was a postdoctoral research associate at University of Illinois at Urbana Champaign. From 2014 to 2025, he was a professor at Incheon National University, Incheon, Republic of Korea. Since 2025, he has been a professor at Kyung Hee University, Seoul, Republic of Korea.



Taehui Na received his B.S. and Ph.D. degrees in electrical & electronic engineering from Yonsei University, Seoul, Republic of Korea, in 2012 and 2017, respectively. From 2017 to 2019, he was with Samsung Electronics Co., Ltd., Hwasung, Republic of Korea, where he worked on phase-change random access memory (PRAM) and high-performance NAND (ZNAND) core circuit designs. Since 2019, he has been a professor at Incheon National University, Incheon, Republic of Korea. His current research interests are focused on process-voltage-temperature variation tolerant and low-power circuit designs for memory, microcontroller unit, and neuromorphic SoC.



Siyeol Lee received his B.S. degree in electronics engineering from Incheon National University, Incheon, Republic of Korea, in 2025. He is currently pursuing an M.S. degree in intelligent semiconductor engineering from Incheon National University, Republic of Korea. His current research interests include PVT variation tolerant and low-power circuit designs for memory, microcontroller unit, and neuromorphic SoC.



Dasom Ahn received her B.S. degree in electronics engineering from Incheon National University, Incheon, Republic of Korea, in 2024. She received the M.S. degree in intelligent semiconductor engineering from Incheon National University, Incheon, Republic of Korea, in 2026. Her current research interests include PVT variation tolerant and low-power circuit designs for memory, microcontroller unit, and neuromorphic SoC.