Check for updates

Advanced Intelligent Systems

ADVANCED INTELLIGENT SYSTEMS Open Access

www.advintellsyst.com

RESEARCH ARTICLE OPEN ACCESS

Input Sparsity-Aware Computing-In-Memory with Bidirectional Conversion-Skippable Analog-to-Digital Converter

Choongseok Song[1] | Yonguk Sim[2] | Doo Seok Jeong[1,2]

[1]Division of Materials Science and Engineering, Hanyang University, Seoul, Republic of Korea | [2]Department of Semiconductor Engineering, Hanyang University, Seoul, Republic of Korea

Correspondence: Doo Seok Jeong (dooseokj@hanyang.ac.kr)

Received: 24 October 2025 | **Revised:** 16 March 2026 | **Accepted:** 27 March 2026

Keywords: bidirectional conversion-skippable ADC | computing-in-memory | deep neural network | input sparsity-aware computing

ABSTRACT

Input sparsity-aware multiply-accumulate computing is a promising solutionto reduce redundant operations and improve energy efficiency in deep neural network accelerators. However, analog computing-in-memory (CIM) architectures encounter challenges in exploiting sparsity due to the fixed resolution of analog-to-digital converters (ADCs), resulting in unnecessary latency and power consumption. This article presents an input sparsity-aware CIM macro with novel bidirectional conversion-skippable ADCs (BCS-ADCs), which enables both input block- and element-level conversion skipping. The proposed BCS-ADC can skip conversion not only on the most significant bit side of multiply-accumulate values based on input element-level sparsity but also on the least significant bit side (equivalent to rounding down). Block-level skipping is implemented using a population counter in the input activation driver. When designed in 45 nm CMOS at 100 MHz, the proposed CIM macro achieves up to 1.95× speedup and 2.44× energy efficiency compared to a nonskipping baseline. Compared to state-of-the-art ADCs, the BCS-ADC reduces power by up to 90.82% and improves conversion latency and energy efficiency by 14.82% and 21.02%, respectively. The BCS-ADC is designed based on the conventional successive approximation ADC with a few additional logic elements that occupy only 0.28% of the ADC area.

1 | Introduction

Deep neural networks (DNNs) have achieved remarkable success across various domains, including computer vision, natural language processing, and autonomous systems [1–11]. As DNNs continue to evolve, their capabilities have become increasingly impressive, often surpassing human-level performance in specific tasks. However, the improvements in model performance are accompanied by increased computational complexity, with state-of-the-art (SOTA) architectures often containing billions of parameters and requiring massive amounts of computation [12, 13]. This exponential growth in computational requirements poses significant challenges for real-world deployment, particularly in terms of energy consumption, hardware resources, and processing latency [14, 15].

Computing-in-Memory (CIM) presents a promising alternative to conventional computing platforms by addressing the limitations of CPUs, due to their limited computational resources, and GPUs, which suffer from substantial energy consumption. Conventional computing hardware, including CPUs and GPUs, is constrained by the physical separation between computational units and memory, requiring frequent data movement for computation, leading to high latency and energy consumption [16]. CIM mitigates these bottlenecks by enabling computation directly within the memory where parameters are stored [17–26]. Moreover, CIM is well-suited for DNN computation as it achieves high parallelism by utilizing individual bit-cells as computing units [27–32].

However, the inherent benefits of CIM architectures are not fully exploited for efficient DNN computation due to the substantial

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2026 The Author(s). *Advanced Intelligent Systems* published by Wiley-VCH GmbH.

area overhead, latency, and energy consumption associated with analog-to-digital converters (ADCs) [33–35]. In particular, the exponential growth in ADC cost (area and power) with increasing resolution imposes significant constraints on CIM efficiency [35, 36]. This limitation necessitates careful consideration of both ADC resolution and architecture selection, such as flash ADC or successive approximation (SAR) ADC implementations, to improve computing efficiency [34, 37].

To address these challenges, input sparsity-aware computing offers an efficient approach for reducing computational overhead for major workloads (multiply-accumulate (MAC) operations) [38–41]. By skipping computation involving zero operands, this computation method reduces latency and energy consumption. It is primarily implemented at the block-level such that MAC operations are skipped if all partial input activations (IAs) - processed simultaneously in a cycle—are zero. Given the efficiency gain at the cost of minimal hardware overhead, this approach is widely adopted across various CIM architectures [40, 41]. However, this method cannot address common cases (not all partial IAs are zero).

A group of partial IAs processed in parallel in a cycle includes a considerable number of zeros. Consequently, parallel MAC operations with a parallelism of 2^b often result in a MAC value (MACV) with a bit resolution less than b. Figure 1a shows the average bit resolution ($\overline{b}$) required for MACVs with parallelism 2^b, highlighting $\overline{b}<b$ given a number of zero operands. Nevertheless, CIM should be designed for b-bit MACVs, which otherwise results in accuracy degradation as shown in Figure 1b. In analog MAC-based CIM architectures utilizing ADCs, this accuracy degradation primarily stems from a global reduction in ADC bit resolution, which shrinks the sensing range and leads to the truncation of most significant bits (MSBs). Since MSBs carry the highest numerical weight, such truncation directly results in significant accuracy drops, as illustrated in Figure 1b.

Therefore, when processing highly sparse nonzero operands, the CIM ADC with a fixed b-bit resolution suffers from unnecessary latency and energy consumption. To cope with this inefficiency, it is desirable to dynamically adjust the ADC resolution for a given group of partial IAs processed in parallel in a cycle with regard to its element-level sparsity.

In this paper, we propose a novel ADC design with minimal modification to the well-established conventional SAR-ADC design to realize dynamic resolution adjustment based on element-level IA sparsity, which enables element-level conversion skipping. This novel ADC is referred to as a bidirectional conversion-skippable ADC (BCS-ADC) since it can skip conversion not only on the MSB side of MACVs but also on the least significant bit (LSB) side (equivalent to rounding down) to boost operational efficiency with trivial degradation of computational accuracy. A population counter is deployed in the IA driver in the CIM macro to evaluate the element-level sparsity. This also identifies block-level sparsity so that our CIM design realizes dual-level (block- and element-level) input sparsity-aware MAC computing. This approach reduces redundant computation while maintaining model accuracy, making it well-suited for real-world deep learning applications. Furthermore, the proposed method can be seamlessly integrated into various CIM architectures, providing a scalable solution for energy-efficient DNN acceleration. The main contributions of our work are summarized as follows:

- We propose a novel BCS-ADC design that dynamically adjusts its bit resolution by selectively skipping analog MACV conversions on the MSB-side, LSB-side, or both. This approach enhances operational efficiency with minimal additional area overhead (<1%) compared to the conventional SAR-ADC.
- We propose an input sparsity-aware CIM macro with BCS-ADCs, which realizes dual-level (block- and element-level) skipping using a population counter deployed in the IA driver.
- We systematically analyze the performance of the proposed CIM macro across multiple models (VGG16, GoogLeNet, ResNet18, ResNet50, and ViT) on CIFAR-10 and CIFAR-100.

The rest of this paper is organized as follows. Section 2 briefly overviews recent works on input sparsity-aware MAC operations and dynamic resolution adjustment. Section 3 elaborates on the proposed CIM macro featuring BCS-ADCs, with a focus on the novel BCS-ADC design. Section 4 reports the performance of our

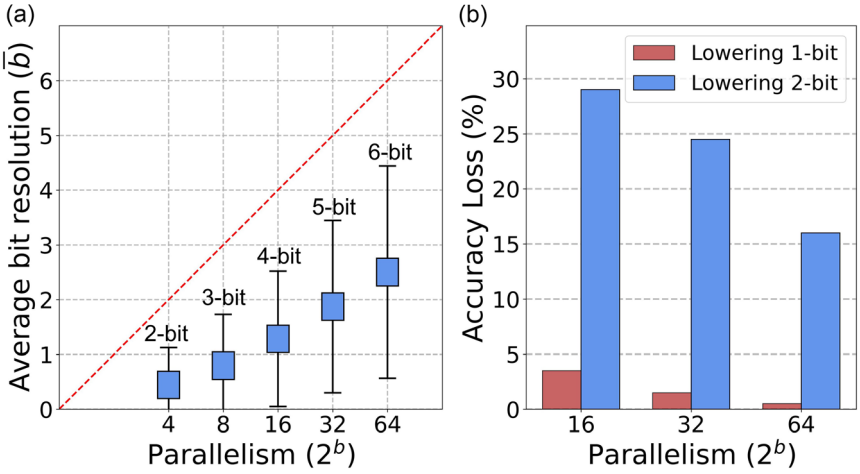

FIGURE 1 | (a) Average bit resolution $\overline{b}$ required for parallel MAC operations with respect to parallelism 2^b and corresponding ADC resolution b. (b) Accuracy loss caused by reducing the ADC resolution b by 1- and 2-bits but maintaining the parallelism 2^b. All data were acquired for ResNet50 on CIFAR-100.

26404567, 2026, 6, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aisy.202501336 by Hanyang University Library, Wiley Online Library on [19/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

CIM macro designed using 45 nm CMOS technology. The speedup and energy efficiency gains for multiple representative models are elaborated on in comparison with SOTA ADC designs. Further, we propose strategies to further improve operational efficiency with marginal accuracy degradation by leveraging the bidirectional conversion skipping capability of the BCS-ADC. Sec. 5 concludes our work.

2 | Related Works

In CIM, input sparsity-aware MAC computing frequently mitigates the cost overhead of ADCs through two approaches: block-level and element-level conversion skipping schemes. Block-level skipping schemes disable ADC operations when all operands within group of partial IAs (processed in parallel) are zero [41], whereas element-level skipping schemes leverage input sparsity at a finer-grained scale to improve the speedup and energy efficiency of CIM [38].

STICKER-IM [41] introduced a CIM architecture supporting a block-level sparsity-aware computing scheme. This scheme classifies operations into three distinct states based on the IA sparsity and a predefined threshold for sparse computation: (i) all-zeros state that skips the entire computation cycles, (ii) sparse state that doubles the parallelism to accelerate computation, and (iii) dense state that processes eight active rows. Although this approach improves throughput without introducing additional ADC overheads, it suffers from threshold-dependent accuracy degradation and requires additional offline training for the block-level row control.

Ali et al. [38]. proposed a dynamic ADC resolution scheme that reconfigures the ADC resolution in the range $2-6$b using two 2-bit and two 3-bit elementary SAR-ADCs interconnected by bridge capacitors and switches. This reconfigurable ADC's resolution is adjusted by a sparsity estimator that counts ones across 64 IA bit streams, leveraging input sparsity at a finer grained scale (element-level). Adjusting the ADC resolution optimally improves the ADC latency and power consumption. However, this reconfigurable ADC incurs significant area and energy overhead due to the extensive use of capacitors.

Si et al. [37]. introduced a low MAC-aware scheme that identifies low MAC cases (small analog MACVs) by using the predefined low MAC threshold (LMT). The LMT is acquired by offline analysis of DNN-level workloads with regard to the weight distribution. For given IAs, the resulting analog MACV is first compared with the LMT to identify low MAC cases. A low-precision conversion process is applied to the low MAC cases to reduce both conversion latency and energy consumption. Otherwise, standard 5-bit SAR-ADC operations are performed, generating 5-bit MACVs over five cycles. This scheme can accelerate the conversion of low MAC cases that account for over 60% of total MAC operations. However, this scheme relies on the predefined LMT, requiring prior analysis of network-level workloads.

Our CIM design featuring the BCS-ADC fully supports both coarse- and fine-grained scale sparsity-aware MAC operations without the use of predefined thresholds unlike the above-mentioned works. Furthermore, the BCS-ADC supports dynamic resolution adjustment with minimal modifications to the conventional SAR-ADC design unlike [38].

3 | Proposed Architecture

3.1 | CIM Architecture and Analog Computing

Figure 2a illustrates the proposed SRAM-based weight-stationary CIM architecture for bit-serial 8-bit IAs and 8-bit weights, implementing the proposed input sparsity-aware MAC computing with dual-level skipping. A group of 32 IAs is simultaneously applied to dedicated 32×8 bit-cells in a bit-serial manner through the IA driver. The IA driver includes a population counter that counts the number of ones across the 32 parallel IAs for each digit. The analog MACVs from the CIM array are converted into digital signals by eight BCS-ADCs in parallel and aligned based on the IA digit applied in a bit-serial manner. They are repeatedly accumulated by the shift-and-add unit (SnA).

We used an 8T-SRAM design that incorporates two additional pass transistors into a conventional 6T-SRAM for analog CIM, as shown in Figure 2b [42]. The 8T-SRAM cells, which share the RBL, establish a connection between the RBL and GND based on the bit stored in the 6T-SRAM cells as the weight and the IN signal from the IA driver. This mechanism functions as a bit-wise multiplier within the bit-cell, discharging the RBL, which is precharged to V_{ref}, only when both the input and the weight are '1'. When the two pass transistors operate in the saturation region, V_{RBL} can be expressed as follows

$$\Delta V_{\text{RBL}} \approx \sum_{i \in IA} (\text{IN}[i]\text{d}[i]) \cdot I_{\text{tr}} \cdot T_{\text{pulse}} / C_{\text{RBL}} \tag{1}$$

where IN$[i]$ and d$[i]$ denote the IN signal and bit-cell data of the i-th row, respectively; I_{tr} represents the saturation current through the two pass transistors; T_{pulse} denotes the pulse width of the IN signal; and C_{RBL} represents the RBL capacitance. A set of rows that are simultaneously activated is denoted by IA. The saturation region refers to the operating mode where each transistor maintains a gate-to-source voltage (V_{GS}) greater than its threshold voltage (V_{TH}), while the drain-to-source voltage (V_{DS}) remains smaller than V_{GS}. In this region, the transistor delivers a nearly constant current, enabling stable charge transfer in the circuit. For 5-bit ADC operation, 32 IN signals are activated and eight discharged $V_{\text{RBL}} (= V_{\text{ref}} - \Delta V_{\text{RBL}})$ values are simultaneously converted to digital outputs through eight BCS-ADCs.

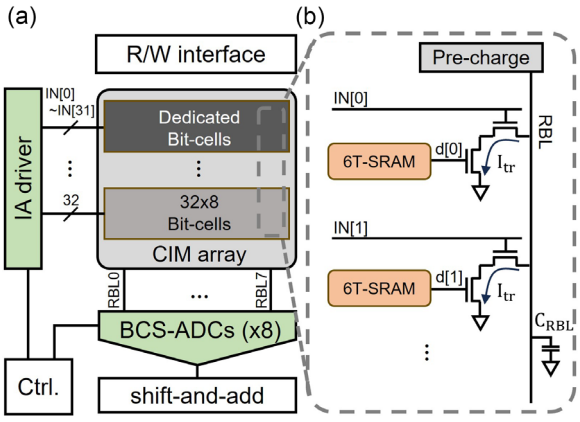

FIGURE 2 | Schematics of (a) the proposed CIM macro and (b) bit-cells for analog MAC operations.

26404567, 2026, 6, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aisy.202501336 by Hanyang University Library, Wiley Online Library on [19/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

3.2 | Bidirectional Conversion-Skippable ADC

Our BCS-ADC is designed based on the conventional SAR-ADC with a maximum resolution of 5-bit. The SAR logic of SAR-ADC is modified to rule out the MSB-side zero-bits of MACVs to reduce the ADC cycles and thus energy. The modified SAR logic is referred to as BCS logic. The upper bound of MACV at a cycle can be estimated by counting the number of ones in the input signal (IN_{sum}) regarding the following inequality

$$\sum_{i\in IA} IN[i]d[i] \le \sum_{i\in IA} IN[i] = IN_{sum} \tag{2}$$

where $IN[i] \in \{0,1\}$ and $d[i] \in \{0,1\}$. The MSB-side zero-bits of MACVs are identified from the upper bound of MACV (IN_{sum}). The BCS-ADC skips the conversion for these bits.

Table 1 summarizes the BCS logic output $\boldsymbol{D}$ and skip state with IN_{sum} in different ranges. Note that if $IN_{sum} = 0$, the output MACV is zero irrespective of weight values, and thus the block-level skip mode is activated. Otherwise, one of the element-level skip modes is chosen. The chosen skip mode restricts the range of convertible input voltage V_{RBL}. As shown in Figure 3, Mskip0 (no skipping) covers the whole range of V_{RBL} (V_{RBL}^{M0}) in need of five conversion cycles. In Mskip1, the convertible V_{RBL} range (V_{RBL}^{M1}) is restricted to $1/2V_{ref} - V_{ref}$, eliminating the need for the MSB conversion. Consequently, the BCS-ADC can reduce the number of conversion cycles depending on IN_{sum}, thereby reducing conversion latency. Additionally, V_{RBL} values below the lower bound of V_{RBL}^{Mi} are converted to the upper bound of MACVs for a given IN_{sum}, enabling V_{RBL} clipping. This likely improves the reliability of analog MAC operations by clipping V_{RBL} with uncontrollable variability.

The BCS logic also supports unconditionally skipping conversions on the LSB side of MACVs. The undetermined digits on the LSB sides are filled with zeros, and thus this skipping method is equivalent to rounding MACVs down. Lskip*i* denotes skipping conversions up to *i*th LSB, where Lskip0 denotes no LSB skipping. Conversion sequences in Lskip0 and Lskip1 are illustrated in Figure 4. In Lskip0, all MACVs from 0 to 31 are fully represented, while Lskip*i* ($0 < i \le 2$) allows only MACVs $2^i m$ in the range $[0, 31]$, where m is a nonnegative integer. The BCS logic

TABLE 1 | Output signals of the BCS logic and its control signals in Mskip and Lskip states.

IN_{sum}	D	skip state	Control signal
MSB skip			
[16, 32]	$D_4D_3D_2D_1D_0$	Mskip0	**MS** = 1111
[8, 15]	0 $D_3D_2D_1D_0$	Mskip1	**MS** = 0111
[4, 7]	0 0 $D_2D_1D_0$	Mskip2	**MS** = 0011
[2, 3]	0 0 0 D_1D_0	Mskip3	**MS** = 0001
1	0 0 0 0 D_0	Mskip4	**MS** = 0000
0	—	Block_skip	—
LSB skip			
—	$D_4D_3D_2D_1D_0$	Lskip0	**LS** = 11
—	$D_4D_3D_2D_1$ 0	Lskip1	**LS** = 10
—	$D_4D_3D_2$ 0 0	Lskip2	**LS** = 00

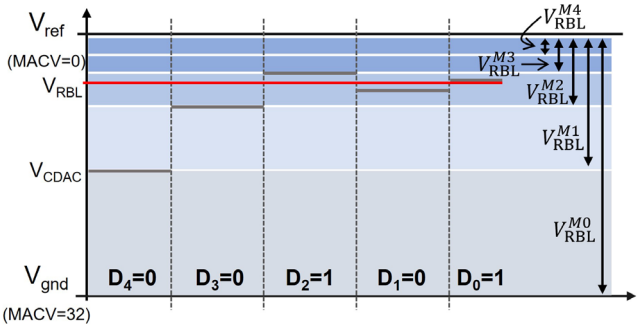

FIGURE 3 | Example of binary conversion in the BCS-ADC and range of V_{RBL} in each Mskip state.

signals for Lskip states are summarized in Table 1. Note that the BCS logic supports simultaneous Mskip and Lskip. The Lskip modes reduce both conversion latency and energy consumption for each analog MACV conversion, though it may cause accuracy degradation. In Section 4.3, we propose a strategy to minimize such accuracy degradations.

Figure 5a shows a schematic of the proposed BCS-ADC, highlighting the BCS logic in (b). The BCS logic is controlled by the Mskip state signal **MS** and Lskip state signal **LS**, which are summarized in Table 1. For Mskip only, Lskip is set to Lskip0 (**LS** = 11) while Mskip is set to Mskip0 (**MS** = 1111) for Lskip only. Mskip and Lskip can be chosen simultaneously. The BCS logic is a modification of the conventional SAR logic with transmission gate-based flip-flops (FFs) [43]. Three additional components (skip path and Mskip and Lskip controllers) are integrated into the conventional SAR logic, which are highlighted in Figure 5b. The conventional SAR logic consists of (i) a group of successive FFs (FF0-FF6) that send an output pulse to the adjacent FF once every clock cycle and (ii) another group of successive FFs (FF7-FF11) that output capacitive digital-to-analog converter (CDAC) control signals D_i in sync with the signal hopping through FF0-FF6.

3.2.1 | Skip path

The skip path bypasses Group (i) FFs that are paired with Group (ii) FFs outputting $D_i = 0$ because of the Mskip control signal **MS**. Additionally, it generates the end-of-signal (eos) signal to terminate further signal propagation and thus iterations when $LS[i] = 0$ for Lskip. The skip path consists of six multiplexers (MUXs).

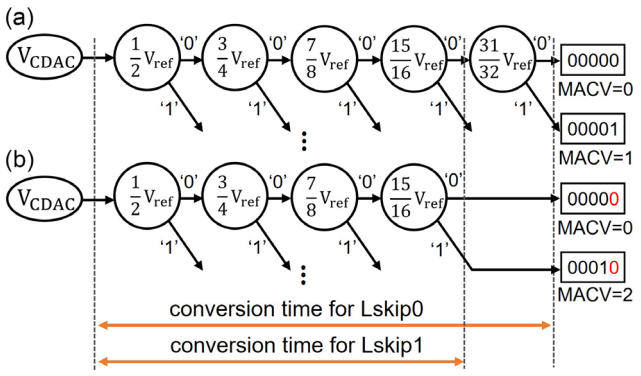

FIGURE 4 | Conversion cycles in (a) Lskip0 and (b) Lskip1.

26404567, 2026, 6, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aisy.202501336 by Hanyang University Library, Wiley Online Library on [19/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

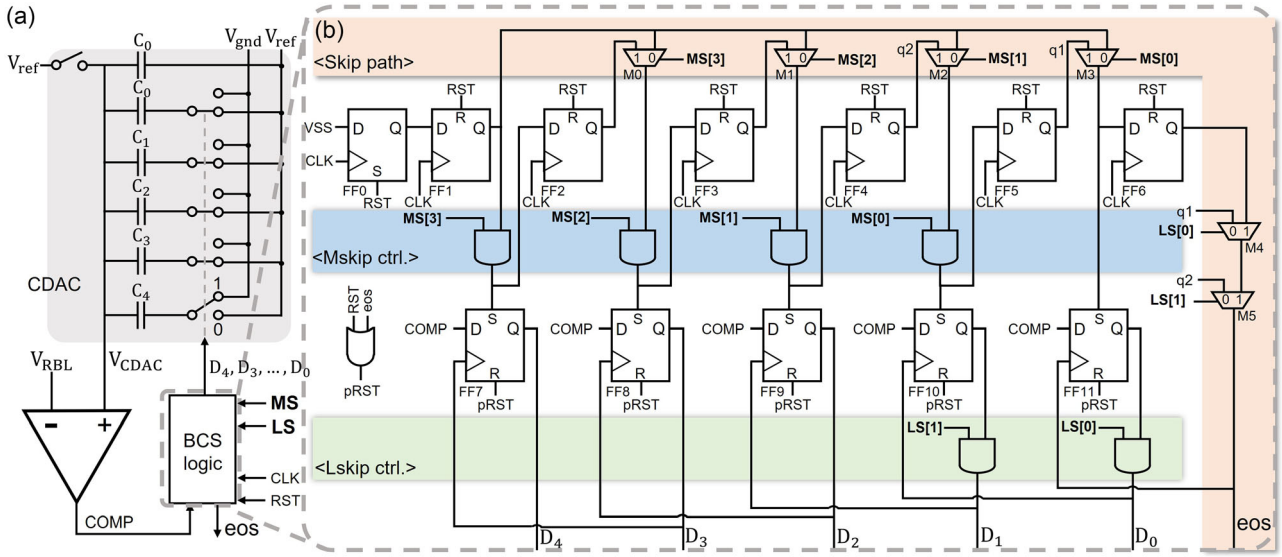

FIGURE 5 | Schematics of (a) BCS-ADC and (b) BCS logic.

3.2.2 | Mskip controller

The Mskip controller regulates the signal transmission to Group (ii) FFs upon a given Mskip control signal **MS** using four AND gates.

3.2.3 | Lskip controller

The Lskip controller regulates the BCS logic outputs D_0 and D_1 upon a given Lskip control signal **LS** using two AND gates.

For instance, when Mskip2 is chosen (i.e., $4 \le \mathrm{IN}_{\mathrm{sum}} \le 7$), the control signal **MS** = 0011 is generated, enabling the direct pulse transmission from FF1 to FF4 through M1 MUX in the Skip path (bypassing FF2 and FF3). The pulse then sequentially propagates through FF5 and FF6 before generating the eos signal if Lskip0 (**LS** = 11) is chosen. This process eliminates the MSB and MSB-1 conversion cycles in the BCS-ADC, thereby avoiding capacitive switching for C_4 and C_3, and thus saving a significant amount of power. If Lskip1 (**LS** = 10) is chosen instead of Lskip0, the pulse bypasses FF6 and directly triggers the eos signal, so that the LSB conversion cycle (and thus C_0 switching) is skipped.

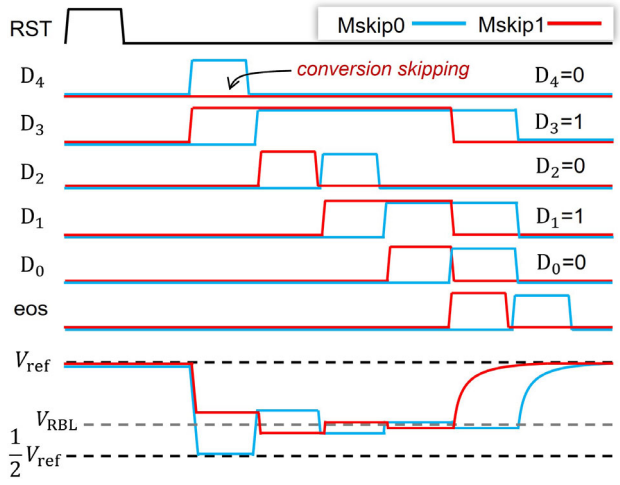

FIGURE 6 | BCS-ADC timing diagram for MACV = 10 in Mskip0 and Mskip1.

Figure 6 illustrates a timing diagram for BCS logic output signals **D** and V_{CDAC} for MACV = 10 in Mskip0 in comparison with Mskip1. After precharging both the top and bottom plates of the capacitor array to V_{ref}, the CDAC generates $V_{\mathrm{CDAC}} = 1/2V_{\mathrm{ref}}$ in the first (MSB) conversion cycle in the Mskip0, which is followed by the four SAR-ADC cycles and eventual eos signal generation. In contrast, Mskip1 initially generates $V_{\mathrm{CDAC}} = 3/4V_{\mathrm{ref}}$ by D_3 by skipping the MSB conversion, reducing conversion latency and power consumption. CDAC in the SAR-ADC consumes a considerable amount of power by capacitive switching, dominating the SAR-ADC (and also our BCS-ADC) power consumption. Among the $b+1$ capacitors in CDAC in a b-bit SAR-ADC, the largest capacitor C_{b-1} is responsible for the MSB conversion by dominantly consuming large power [44]. Since Mskip avoids switching the largest capacitor C_{b-1}, we expect a significant reduction in BCS-ADC power consumption.

3.3 | IA Driver with Population Counter

The IA driver in our CIM macro comprises (i) a register file, (ii) bit selector, (iii) row activator, and (iv) population counter as shown in Figure 7. The register file stores 32 8-bit IAs, and the bit selector chooses a particular digit of the 32 IAs for bit-wise MAC operations. The row activator simultaneously activates 32 rows of the total 256 to apply the 32 IAs in a bit-serial fashion while the other rows are grounded. The population counter computes the number of ones across 32 parallel IAs per cycle ($\mathrm{IN}_{\mathrm{sum}}$ in Equation (2)). $\mathrm{IN}_{\mathrm{sum}}$ determines whether to apply one of the Mskip modes (element-level skipping) or block-level skipping (if $\mathrm{IN}_{\mathrm{sum}} = 0$) as summarized in Table 1.

4 | Evaluation

4.1 | Experimental Setup

We designed our CIM macro with BCS-ADCs using Cadence GPDK045 (45 nm CMOS technology) with a 100 MHz clock frequency and a 1 V supply voltage. The analog components were designed and simulated using Cadence Virtuoso and Specter,

26404567, 2026, 6, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aisy.202501336 by Hanyang University Library, Wiley Online Library on [19/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

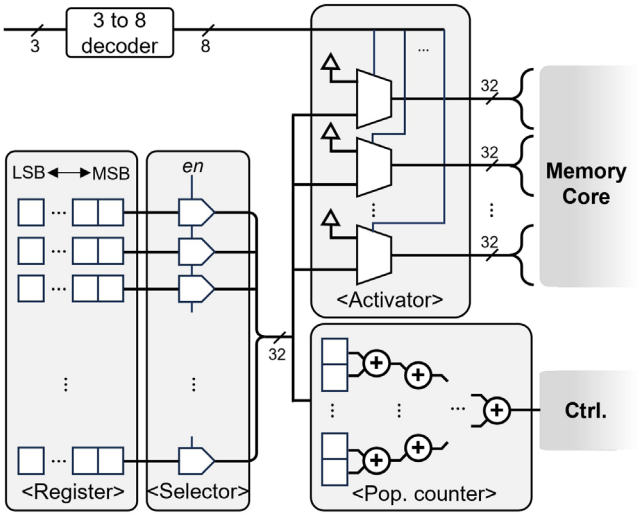

FIGURE 7 | Schematic of IA driver.

respectively, while the digital components were synthesized using Cadence GENUS. The performance of the CIM macro under DNN workloads was evaluated on five models (VGG16 [1], GoogLeNet [45], ResNet18/50 [2], and ViT [46]) on CIFAR-10 and CIFAR-100. Given that our CIM macro was designed for 8-bit IAs and weights, we quantized the weights and IAs in the FP32-trained models to INT8 using a uniform quantization method. Following the weight-stationary dataflow, 256 8-bit weights were distributed across the 256 rows in the CIM array, which consists of 256×8 bit-cells, that is, one weight per row. For convolution layers, we mapped weights along the C_{in} direction in a $C_{\text{out}} \times C_{\text{in}} \times k_H \times k_W$ kernel across the 256 rows in the CIM array. During inference, these 8-bit IAs are fed into the CIM array in a bit-streaming manner, being processed bit-by-bit from MSB to LSB.

To evaluate the DNN workload-level performance, we integrated the CIM-specific MAC computation flow into an INT8-quantized, Python-based inference environment. The framework emulates the analog MAC process by performing bit-wise multiplications between bit-streaming inputs and weights, where 32 product results are accumulated and subsequently processed through a 5-bit ADC model. Based on this emulated flow, we collected profiling data to reflect the proposed BCS-ADC mechanisms: (1) input bit-stream profiling to determine Mskip states for calculating latency and energy efficiency and (2) ADC ouptput profiling to apply manually assigned Lskip states, where deterministic rounding down is performed. By propagating these outputs to subsequent layers, we evaluated the cumulative impact on classification accuracy, while also accounting for the additional speedup and energy gains provided by the Lskip mechanism.

4.2 | Power, Area, and Latency

The power and area overheads of the primary CIM components are summarized in Table 2, which were evaluated through prelayout simulation using Cadence GPDK045. The power consumption of the 256×8 CIM array varies in the range $6.4 - 136.0$ μW upon MACV that determines the degree to which C_{RBL} discharges. The population counter integrated into the IA driver accounts for only 4.64% of the IA driver's power overhead and 5.85% of its area overhead. The area breakdown of the BCS-ADC is also summarized in Table 2. As noted, the BCS-ADC introduces only minor modifications to the SAR logic of the SAR-ADC, incorporating the Skip path along with the Mskip and Lskip controllers. These additional components account for only 0.28% of the total area of the proposed BCS-ADC, confirming that the hardware cost for enabling the skipping functionality is marginal compared to a standard SAR-ADC.

TABLE 2 | Summary of power and area overheads of CIM components.

Module	IA driver (pop. count.)	CIM array	SnA
Power, μW	99.2 (4.6)	6.4–136.0	17.8
Area, μm^2	3188.5 (186.4)	5610.4	549.3
BCS-ADC area breakdown, μm^2			
CDAC	**BCS logic (additional logic)**	**Comparator**	
392.1	6.2 (1.1)	0.2	

Since the power of the BCS-ADC varies upon Mskip and Lskip modes, we separately address its power with different modes in Figure 8. The BCS-ADC operates with a total of seven conversion clock cycles in the nonskipping state (Mskip0 and Lskip0), consisting of two clock cycles for the precharge phase and five clock cycles for the binary conversion phase, while consuming 5.3 μW of power. In Mskip$i$ ($i \in \{1, 2, 3, 4\}$) with Lskip0, the conversion requires $7 - i$ clock cycles by skipping the switching of capacitors $(C_4, \ldots, C_{5-i})$ in the CDAC. This leads to reductions in power consumption from 5.3 μW in Mskip0 to 3.2 μW in Mskip4 as plotted in Figure 8. Mskip$i$ ($i \in \{0, 1, 2, 3\}$) with Lskip1 results in a further decrease by one clock cycle, leading to a required conversion cycle of $6 - i$. Note that Mskip4 with Lskip1 was ruled out because all conversions are skipped in this case. Similar to the previous case, the power consumption decreases from 5.6 μW in Mskip0 to 3.3 μW in Mskip3. To be more precise, the capacitor switching skip enhances energy efficiency rather than power efficiency. Mskip0 with Lskip1 consumes a larger power than Mskip0 with Lskip0 because the energy efficiency enhancement is normalized by the shorter conversion time (six clock cycles) for

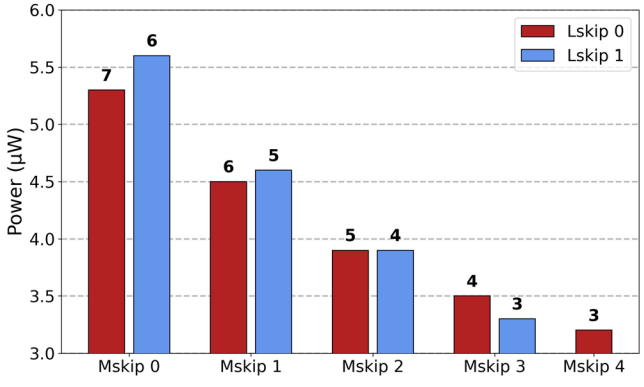

FIGURE 8 | Power consumption of BCS-ADC for each skip state. The bold numbers above the bars indicate the clock cycles required for conversion.

26404567, 2026, 6, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aisy.202501336 by Hanyang University Library, Wiley Online Library on [19/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Mskip0 with Lskip1. The same holds for Mskip1 with Lskip1, which consumes a larger power than Mskip1 with Lskip0.

Since the entire architecture operates within a synchronous execution flow at a fixed clock frequency, the reduction in conversion clock cycles directly translates into an overall reduction in operation latency. Unlike a conventional SAR-ADC that requires a fixed number of cycles, the BCS-ADC dynamically minimizes the total cycle count for each multi-cycle MAC operation through the Mskip and Lskip mechanisms.

Furthermore, the total area overhead for the entire sparsity-aware CIM architecture is approximately 1.58% compared to a conventional analog CIM baseline when accounting for both the IA driver with a population counter and the replacement of eight standard SAR-ADCs with the proposed BCS-ADCs. This result demonstrates that our proposed sparsity-aware mechanisms can be integrated with minimal area impact at both the component and architecture levels.

4.3 | DNN Workload-Level Performance

The proposed dual-level conversion skipping methods (Mskip and Block_skip) using the BCS-ADC cause no accuracy loss. We evaluated the speedup and energy efficiency of our CIM macro in four cases: (1) skip mode deactivated (baseline), which executes all MAC operations without skipping, (2) dual-level skip mode (Mskip and Block_skip) activated conditionally based on IN_{sum} following Table 1, (3) element-level skip mode (Mskip) only, and (4) block-level skip model (Block_skip) only. Figure 9 presents the analysis results for five models evaluated on CIFAR-10 and CIFAR-100. The dual-level skip mode (Mskip and Block_skip) demonstrates significant improvements in both speedup and energy efficiency, achieving $1.60-1.95\times$ and $1.97-2.44\times$ gains, respectively, for VGG16, GoogLeNet, ResNet18, and ResNet50. These models inherently exhibit high IA sparsity due to the use of ReLU activation functions. The speedup improvement achieved by the dual-level skip mode surpasses those of the element-level mode alone ($1.33-1.43\times$) and the block-level mode alone ($1.34-1.59\times$). The energy-efficiency improvement achieved by the dual-level skip mode also surpasses those of the element-level mode alone ($1.80-2.08\times$) and the block-level mode alone ($1.33-1.58\times$).

Unlike the ReLU-based models, ViT exhibits limited IA sparsity due to the use of GeLU activation functions. Consequently, the block-level skip mode alone barely yields improvements in speedup and energy efficiency. The dual-level skip mode achieves a speedup improvement of 6% on CIFAR-10 and 5% on CIFAR-100, along with energy efficiency gains of 15% and 12%, respectively, as shown in Figure 9. These improvements primarily arise from element-level conversion skipping.

Applying Lskip in conjunction with the dual-level skip mode (Mskip and Block_skip) can yield further improvements in both speedup and energy efficiency. However, unlike the dual-level skip mode alone, this approach may incur a trade-off in classification accuracy. We observed these additional performance gains and the associated accuracy degradation across GoogLeNet, ResNet18, and ResNet50 on the CIFAR-100 dataset. Given that our CIM macro processes 32 parallel IAs using a bit-serial scheme, we can selectively apply the Lskip mode at the level of individual IA digits. We considered the following scenarios

4.3.1 | Scenario 1

Lskip1 is applied to one of the eight digits across the 32 IAs, resulting in eight distinct cases.

4.3.2 | Scenario 2

Lskip1 is applied to a pair of consecutive digits across the 32 IAs, resulting in four distinct cases.

4.3.3 | Scenario 3

Lskip1 is applied to a group of four consecutive digits across the 32 IAs, resulting in two distinct cases.

For all scenarios, we avoided applying Lskip1 to MACVs in Mskip4 and Block_skip.

Figure 10 presents the results for the eight cases in Scenario 1, four cases in Scenario 2, and two cases in Scenario 3. In Scenarios 1 and 2, where Lskip1 is applied to the MSB or to both the MSB and MSB-1 of the IAs, the impact on speedup and energy efficiency is marginal. This is due to the inherently high sparsity of these digits, which already triggers Mskip4 or Block_skip in most cases as shown in Figure 11. As noted, Lskip was avoided

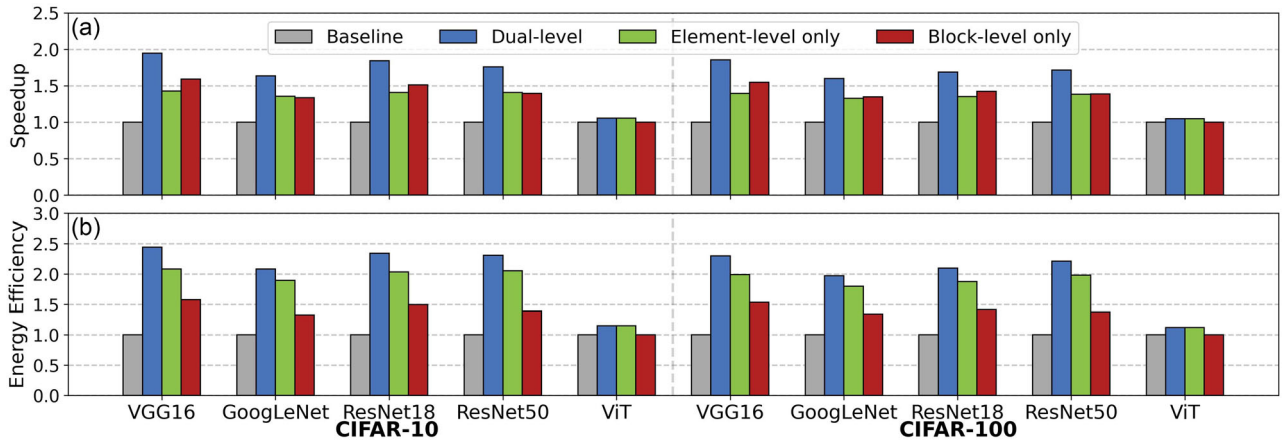

FIGURE 9 | (a) Speedup and (b) energy efficiency gains for dual-level skip (Mskip and Block_skip), element-level skip (Mskip) only, and block-level skip (Block_skip) only.

26404567, 2026, 6, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aisy.202501336 by Hanyang University Library, Wiley Online Library on [19/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

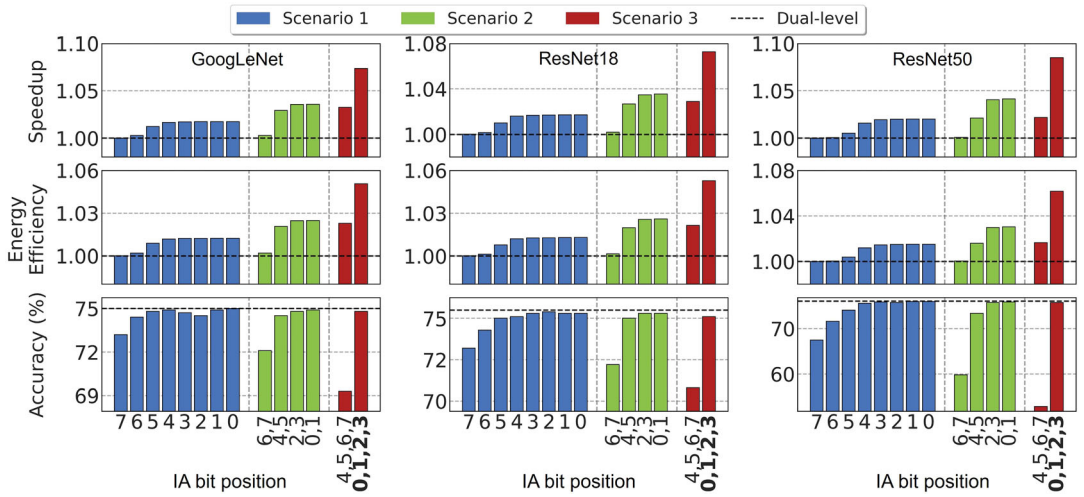

FIGURE 10 | Additional improvements in speedup and energy efficiency by unconditional Lskip1 (skipping the conversion of the MACV's LSB) in three different scenarios for GoogLeNet, ResNet18, and ResNet50 on CIFAR-100. The performance in the dual-level skip mode (Mskip and Block_skip) is taken as the baseline. Accuracy loss due to Lskip1 is also plotted for each case.

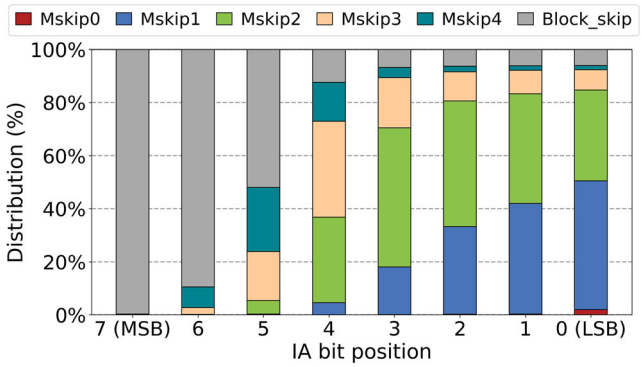

FIGURE 11 | Skip state distribution across the bit positions of 8-bit IAs for ResNet50 on CIFAR-100. For the MSB, the distribution consists of 0.25% Mskip3, 0.02% Mskip4, and 99.73% block_skip.

when Mskip4 or Block_skip is activated. However, Lskip in these cases results in significant accuracy degradation. As in Figure 11, the IA sparsity is relatively low near the LSB. Consequently, applying Lskip1 for the LSB of the IAs in Scenario 1 yields additional improvements in speedup by 1.74% – 2.02% and energy efficiency by 1.22% – 1.50% with only a marginal accuracy loss of 0.0% – 0.2%. Notably, applying Lskip1 for the lower four bits of the IAs in Scenario 3 improves speedup by 7.27% – 8.51% and energy efficiency by 5.08% – 6.19% with a marginal accuracy loss of 0.2% – 0.4%.

4.4 | Comparison with SOTA ADCs

We compare the performance of the BCS-ADC with two SOTA ADCs designed to enhance the CIM performance: (1) ADC with the low MAC-aware scheme [37] and (2) ADC with reconfigurable resolution upon input sparsity [38]. While these prior works aim to reduce conversion cycle, they involve distinct trade-offs. The low MAC-aware scheme [37] requires no additional peripheral circuitry for skipping conversion cycles. However, as a fixed threshold based method, it necessitates offline analysis to predefine the appropriate LMT, which limits its adaptability to real-time input variations. In the case of the reconfigurable resolution ADC [38], the sensing margin can be enhanced at lower resolutions by maintaining a fixed sensing range. Nevertheless, this approach requires additional analog components such as amplifiers, increasing circuit complexity and power consumption. This limits the potential for power reduction of ADC despite the reduced conversion cycles. The proposed BCS-ADC achieves dynamic skipping of both conversion cycles and their corresponding capacitor switching. This mechanism enables simultaneous gains in speed and energy efficiency without the need for offline analysis or additional analog amplifiers. However, the sensing margin does not inherently increase because the effective sensing range is adjusted to accommodate the skip states. These comparisons are summarized in Table 3.

For these three ADCs, their ADC latency and energy consumption were evaluated for GoogLeNet, ResNet18, ResNet50, and ViT on CIFAR-100. The results are shown in Figure 12. Our

TABLE 3 | Comparison with state of the art ADCs.

	[37][a]	[38]	Ours
Tech. node (nm)	45	65	45
Frequency (MHz)	100	Async.	100
Supply voltage (V)	1.0	1.1	1.0
Resolution (bit)	5	2–6	5
Latency (ns)	40[b] 80[c]	3.5–20	30–70
Energy (pJ)	0.14[b] 0.43[c]	0.2–1.1	0.10–0.37
Sensing margin	Fix	Variable	Fix
Additional circuit	No	Pop. count Amplifier	Pop. count
Offline analysis	Yes	No	No
Sparsity-aware	No	Yes	Yes

[a]Technology scaling to 45 nm.
[b]Low MAC case.
[c]Non low MAC case.

26404567, 2026, 6, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aisy.202501336 by Hanyang University Library, Wiley Online Library on [19/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

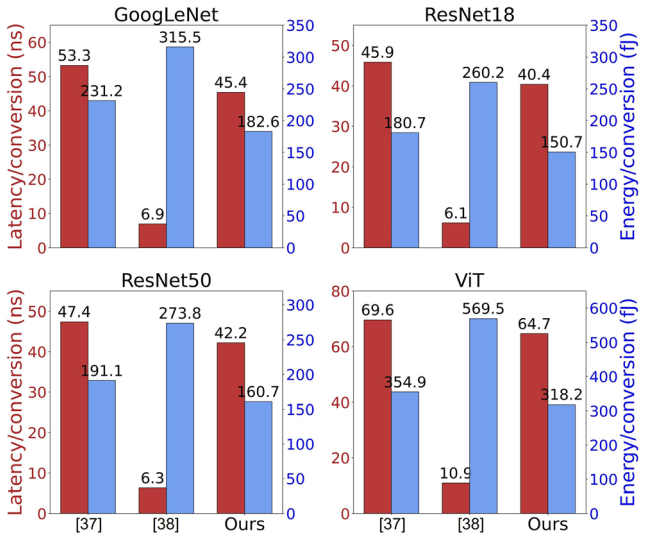

FIGURE 12 | Performance comparison of the BCS-ADC with two SOTA ADCs for GoogLeNet, ResNet18, ResNet50, and ViT on CIFAR-100.

BCS-ADC and the ADC in [37] perform synchronous iterative conversion at a clock frequency of 100 MHz while the ADC in [38] is designed to perform asynchronous conversion that is completed within 20 ns at the cost of high energy consumption. The BCS-ADC reduces energy consumption by 41.31% – 44.13% compared to the ADC in [38], leading to a 90.93% – 91.26% reduction in power consumption at the cost of a 6.16 – 6.70× increase in latency. Our BCS-ADC surpasses the ADC with the low-MAC-aware scheme in [37] in both speedup (7.04% – 14.82% reduction in latency) and energy consumption (10.34% – 21.02% reduction), resulting in a 3.53% – 7.37% reduction in power consumption, across all models.

4.5 | Robustness to Variability in Analog MAC Operations

The BCS-ADC offers the upper bound of MACVs conforming to the inequality in Equation (2), restricting the level to which V_{RBL} discharges (V_{RBL}^{Mi} for Mskip*i*) based on the IA sparsity IN_{sum} as in Figure 3. Therefore, V_{RBL} values below the lower bound of V_{RBL}^{Mi} are converted to the upper bound of MACVs, enabling V_{RBL} clipping. This issue may stem from the uncontrollable and inherent variability in analog MAC operations and/or short-circuited memory bitcells in the worst case. For instance, in Mskip1 (MACV upper bound of 15), any variability-induced V_{RBL} values below $V_{ref} - V_{RBL}^{M1}$ are converted to MACV = 15. This V_{RBL} clipping effect likely improves the robustness to variability in analog MAC operations.

5 | Conclusion

The proposed input sparsity-aware CIM macro featuring the BCS-ADC supports both coarse (block-level) and fine (element-level) grained scale sparsity-aware computing. The BCS-ADC, designed for bidirectional element-level skipping (Mskip and Lskip), incurs only a 0.28% area overhead, while achieving 1.60 – 1.95× speedup and 1.97 – 2.44× energy efficiency improvement across DNN workloads (VGG16, GoogLeNet, ResNet18, and ResNet50) compared to the baseline (nonskipping), primarily due to the MSB-side conversion skipping (Mskip). Furthermore, additional LSB-side conversion skipping (Lskip) achieves additional improvements in speedup by 7.27% – 8.51% and energy efficiency by 5.08% – 6.19% at the cost of only a 0.2% – 0.4% accuracy loss. Compared to SOTA ADCs, the BCS-ADC achieves power savings. It reduces power consumption by 90.93% – 91.26% compared to the reconfigurable precision ADC [38], and by 3.53% – 7.37% compared to the low-MAC-aware ADC scheme [37].

Author Contributions

Choongseok Song: conceptualization (equal); data curation (lead); validation (lead); writing – original draft (equal) **Yonguk Sim**: conceptualization (supporting); data curation (supporting) **Doo Seok Jeong**: conceptualization (lead); project administration (lead); supervision (lead); writing – original draft (equal).

Funding

This research was supported by National R&D Program through the National Research Foundation of Korea (NRF) funded by Ministry of Science and ICT (RS-2025–02654040). This work was partly supported by IITP under the artificial intelligence semiconductor support program to nurture the best talents (RS-2023–00253914) grant funded by the Korea government (MSIT). The EDA tool was supported by the IC Design Education Center (IDEC), Republic of Korea.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

1. K. Simonyan and A. Zisserman, arXiv preprint arXiv:1409.1556, (2014).

2. K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016): 770–778.

3. A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep Learning for Computer Vision: A Brief Review," *Computational Intelligence and Neuroscience* 2018, no. 1 (2018): 7068349.

4. M.-H. Guo, T.-X. Xu, J.-J. Liu, et al., "Attention Mechanisms in Computer Vision: A Survey," *Computational Visual Media* 8 (2022): 3–331.

5. J. Hirschberg and C. D. Manning, "Advances in Natural Language Processing," *Science* 349, no. 6245 (2015): 6261.

6. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies*, (Association for Computational Linguistics (ACL), 2019), 4171–4186, 1. (long and short papers).

7. Y. Liu, M. Ott, N. Goyal, et al., arXiv preprint arXiv:1907.11692 (2019).

8. K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, arXiv preprint arXiv:2003.10555, (2020).

9. M. Saleem, A. Khadim, M. Fatima, M. A. Khan, H. K. Nair, and M. Asif, *2022 International Conference on Cyber Resilience (ICCR)*, (IEEE, 2022), 1–6.

26404567, 2026, 6, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aisy.202501336 by Hanyang University Library, Wiley Online Library on [19/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

10. B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Machine-learning-enabled Cooperative Perception for Connected Autonomous Vehicles: Challenges and Opportunities," *IEEE Transactions on Neural Networks and Learning Systems* 29, no. 6 (2017): 2042.

11. Q. Yang, S. Fu, H. Wang, and H. Fang, "[Back inside Cover]," *IEEE Network* 35, no. 3 (2021): 96.

12. A. Radford, J. Wu, R. Child, et al., "Language Models are Unsupervised Multitask Learners," *OpenAI Blog* 1, no. 8 (2019): 9.

13. T. Brown, B. Mann, N. Ryder, et al., "Language Models are Few-shot Learners," *Advances in Neural Information Processing Systems* 33 (2020): 1877.

14. S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark Analysis of Representative Deep Neural Network Architectures," *IEEE Access* 6 (2018): 64270.

15. N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, arXiv preprint arXiv:2007.05558 (2020).

16. C. Song, C. Ye, Y. Sim, and D. S. Jeong, "Hardware for Deep Learning Acceleration," *Advanced Intelligent Systems* 6 (2024): 2300762.

17. H. You, W. Li, D. Shang, Y. Zhou, and S. Qiao, "A 1-8b Reconfigurable Digital Sram Compute-in-memory Macro for Processing Neural Networks," *IEEE Transactions on Circuits and Systems I: Regular Papers* 71, no. 4 (2024): 1602.

18. Y. Ma, Y. Qiu, W. Zhao, et al., "DCIM-GCN: Digital Computing-in-memory Accelerator for Graph Convolutional Network," *IEEE Transactions on Circuits and Systems I: Regular Papers* 71 (2024): 2735–2748.

19. X. Qiao, Q. Guo, X. Tang, et al., "A 16.38 TOPS and 4.55 POPS/W SRAM Computing-in-memory Macro for Signed Operands Computation and Batch Normalization Implementation," *IEEE Transactions on Circuits and Systems I: Regular Papers* 71, no. 4 (2024): 1706.

20. V. Sharma, X. Zhang, N. S. Dhakad, and T. T.-H. Kim, "FlexDCIM: A 400 MHz 249.1 TOPS/W 64 Kb Flexible Digital Compute-in-memory SRAM Macro for CNN Acceleration," *IEEE Transactions on Circuits and Systems I: Regular Papers* 72 (2025): 5696–5707.

21. S. Kim, S. Um, W. Jo, et al., "Scaling-CIM: EDRAM in-memory-computing Accelerator with Dynamic-scaling ADC and Adaptive Analog Operation," *IEEE Journal of Solid-State Circuits* 59 (2024): 2694–2705.

22. S. Yan, J. Yue, C. He, et al., "A 28-nm Floating-point Computing-in-memory Processor using Intensive-CIM Sparse-digital Architecture," *IEEE Journal of Solid-State Circuits* 59 (2024): 2630–2643.

23. J. Yang, T. Li, W. Romaszkan, P. Gupta, and S. Pamarti, "A 65-nm Digital Stochastic Compute-in-Memory CNN Processor With 8-bit Precisio," *IEEE Journal of Solid-State Circuits* 60 (2025): 3749–3761.

24. A. Guo, C. Xi, F. Dong, et al., "A 28-nm 64-kb 31.6-TFLOPS/W Digital-domain Floating-point-computing-unit and Double-bit 6T-SRAM Computing-in-memory Macro for Floating-point CNNs," *IEEE Journal of Solid-State Circuits* 59, no 9 (2024): 3032.

25. H. Diao, Y. He, X. Li, et al., "A Multiply-less Approximate SRAM Compute-in-memory Macro for Neural-network Inference," *IEEE Journal of Solid-State Circuits* 60 (2024): 695–706.

26. E.-J. Chang, C.-X. Xue, C. Deshpande, et al., *2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits* (IEEE, 2023), 1–2.

27. H. Zhang, S. He, X. Lu, et al., "SSM-CIM: An Efficient CIM Macro Featuring Single-step Multi-bit MAC Computation for CNN Edge Inference," *IEEE Transactions on Circuits and Systems I: Regular Papers* 70, no. 11 (2023): 4357.

28. K. Zhang, D. Zhang, M. Song, et al., "A Novel 9T1C-SRAM Compute-in-memory Macro with Count-less P0ulse-width Modulation Input and ADC-less Charge-integration-count Output," *IEEE Transactions on Circuits and Systems I: Regular Papers* 70, no. 10 (2023): 3944.

29. K. Yoshioka, "A 818-4094 TOPS/W Capacitor-reconfigured Analog CIM for Unified Acceleration of CNNs and Transformers," *IEEE Journal of Solid-State Circuits* 60 (2024): 1844–1855.

30. G. Jedhe, C. Deshpande, S. Kumar, et al., *2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits* (IEEE, 2023), 1–2.

31. K. Lee, J. Kim, and J. Park, *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design* (Institute of Electrical and Electronics Engineers (IEEE), 2022), 1–8.

32. Y.-D. Chih, P.-H. Lee, H. Fujiwara, et al., *2021 IEEE International Solid-State Circuits Conference (ISSCC)* (IEEE, 2021), 64, 252–254,

33. Y. Fu, J. Li, W.-H. Yu, et al., "Clut-cim: A Capacitance Lookup Table-based Analog Compute-in-memory Macro with Signed-channel Training and Weight Updating for Nonuniform Quantization," *IEEE Transactions on Circuits and Systems I: Regular Papers* 71 (2024): 4996–5004.

34. Q. Dong, M. E. Sinangil, B. Erbagci, et al., *2020 IEEE International Solid-State Circuits Conference-(ISSCC)* (IEEE, 2020), 242–244.

35. C. Song, J. Kim, and D. S. Jeong, "Optimal Weight-Splitting in Resistive Random Access Memory-Based Computing-in-Memory Macros," *Advanced Intelligent Systems* 5, no. 1 (2023): 2200289.

36. M. Saberi, R. Lotfi, K. Mafinezhad, and W. A. Serdijn, "Analysis of Power Consumption and Linearity in Capacitive Digital-to-analog Converters used in Successive Approximation ADCs," *IEEE Transactions on Circuits and Systems I: Regular Papers.* 58 no. 8 (2011): 1736.

37. X. Si, Y.-N. Tu, W.-H. Huang, et al., *2020 IEEE international solid-state circuits conference-(ISSCC)* (IEEE, 2020), 246–248.

38. M. Ali, I. Chakraborty, U. Saxena, A. Agrawal, A. Ankit, and K. Roy, "A 35.5-127.2 TOPS/W Dynamic Sparsity-Aware Reconfigurable-Precision Compute-in-Memory SRAM Macro for Machine Learning," *IEEE Solid-State Circuits Letters* 4 (2021): 129.

39. S. Liu, P. Li, J. Zhang, et al., *2023 IEEE International Solid-State Circuits Conference (ISSCC)* (IEEE, 2023), 250–252.

40. Y.-C. Lo, C.-C. Yeh, J.-S. Wu, et al., *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design* (Institute of Electrical and Electronics Engineers (IEEE), 2022), 1–9.

41. J. Yue, Y. Liu, Z. Yuan, et al., "STICKER-IM: A 65 nm Computing-in-Memory NN Processor Using Block-Wise Sparsity Optimization and Inter/Intra-Macro Data Reuse," *IEEE Journal of Solid-State Circuits* 57, no. 8 (2022): 2560.

42. J.-H. Kim, J. Lee, J. Lee, J. Heo, and J.-Y. Kim, "Z-PIM: A Sparsity-aware Processing-in-memory Architecture with Fully Variable Weight Bit-precision for Energy-efficient Deep Neural Networks," *IEEE Journal of Solid-State Circuits* 56, no. 4 (2021): 1093.

43. S. T. Oskuii, "*Comparative Study on Low-Power High-Performance Flip-Flops*," 2004.

44. H.-C. Hong and G.-M. Lee, "A 65-fJ/Conversion-Step 0.9-V 200-kS/s Rail-to-Rail 8-Bit Successive Approximation ADC," *IEEE Journal of Solid-State Circuits* 42 (2007): 102161.

45. C. Szegedy, W. Liu, Y. Jia, et al., *Proceedings of the IEEE conference on computer vision and pattern recognition*, (2015), 1–9.

46. A. Dosovitskiy, L. Beyer, A. Kolesnikov, et al., arXiv preprint arXiv:2010.11929, (2020).

26404567, 2026, 6, Downloaded from https://advanced.onlinelibrary.wiley.com/doi/10.1002/aisy.202501336 by Hanyang University Library, Wiley Online Library on [19/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License