High-Performance In-Memory XNOR Computing: A 65 nm 12T SRAM Architecture for Neural Network Acceleration

Doanh Bui Le Quoc
Department of Electronics, Ho Chi Minh City University of Technology, Vietnam | Vietnam National University Ho Chi Minh City, Vietnam
blqdoanh.sdh212@hcmut.edu.vn

Phuoc Luan Vo
Department of Electronics, Ho Chi Minh City University of Technology, Vietnam | Vietnam National University Ho Chi Minh City, Vietnam
luan.vovpluan100@hcmut.edu.vn

Phuc Nguyen Phan Thien
Department of Electronics, Ho Chi Minh City University of Technology, Vietnam | Vietnam National University Ho Chi Minh City, Vietnam
nptphuc.sdh232@hcmut.edu.vn

Linh Tran
Department of Electronics, Ho Chi Minh City University of Technology, Vietnam | Vietnam National University Ho Chi Minh City, Vietnam
linhtran@hcmut.edu.vn (corresponding author)

Received: 23 April 2025 | Revised: 13 May 2025 | Accepted: 1 June 2025

 DOI: https://doi.org/10.48084/etasr.11646

ABSTRACT

This paper presents a 64×16 XNOR-SRAM array in 65nm CMOS technology for In-Memory Computing (IMC), designed to accelerate deep neural networks with low latency and high-power efficiency. Using a 12-transistor bitcell, the architecture performs XNOR-and-Accumulate (XAC) operations within the SRAM, reducing data movement. Cadence Spectre simulations show a 342.67 ps delay and 901.133 µW power consumption at 1.2 V, with robust ternary and binary operation. A flash Analog-to-Digital Converter (ADC) and an analog multiplexer enhance precision, despite minor nonlinearities from transistor mismatches. Compared to prior designs, the proposed XNOR-SRAM offers competitive latency for edge AI applications.

Keywords-In-Memory Computing (IMC); SRAM; Deep Neural Network (DNN); low latency; power efficiency

I. INTRODUCTION

In the rapidly evolving landscape of computational technology, the demand for faster and more energy-efficient processing has driven significant innovations in computer architecture. Traditional architectures, such as the Harvard model, face limitations due to the memory bottleneck, where data transfer between memory and processing units incurs substantial time and energy costs. To address this challenge, In-Memory Computing (IMC) has emerged as a transformative approach, enabling computations directly within memory units to minimize data movement and enhance performance [1-4]. A notable advancement in this domain is the XNOR-SRAM, a specialized Static Random-Access Memory (SRAM) design that integrates computational capabilities, particularly well-suited for accelerating Deep Neural Network (DNN) operations [5-7].

IMC has emerged as a key solution to address the memory bottleneck in traditional von Neumann and Harvard architectures, particularly for data-intensive applications like machine learning and DNNs [8, 9]. SRAM-based IMC architectures have been extensively studied, each tackling challenges in power consumption, computational delay, and scalability. Authors in [10] developed a 65nm 256×64 6T XNOR-SRAM macro for binary/ternary DNNs, leveraging

analog bitline accumulation for efficient XNOR operations. However, its moderate delay and limited array size constrained throughput for large-scale computations. Similarly, the 65nm 256×256 6T In-Memory Multi-Bit Multiplication and Accumulation (IMAC) architecture developed by authors in [11], which was designed for multi-bit multiplication and accumulation, improved computational efficiency but introduced peripheral circuitry complexity, increasing power consumption for certain operations. Both designs rely on analog signal processing, which, while efficient, faces noise and nonlinearity issues, requiring robust components like sense amplifiers (SAs) or Analog-to-Digital Converters (ADCs).

In contrast, authors in [12] explored a digital IMC approach with a 28nm 64×64 6T SRAM array, focusing on bitline shifting for multi-bit operations. This design achieved an ultra-low power consumption but incurred a high delay, making it less suitable for high-throughput DNN tasks. The reliance on digital processing also reduced flexibility compared to analog-based computations, which are often more efficient for neural networks. Authors in [13] proposed a 65nm CONV-SRAM targeted for in-memory dot-product computations for Convolutional Neural Networks (CNNs). This approach prioritizes energy efficiency but lacks support for multi-bit precision, which limits its use in complex DNNs. Authors in [14] proposed a 65nm mixed-signal binarized CNN accelerator integrated dense weight storage to reduce data movement. However, the focus on binarized networks restricted generalizability. Authors in [15] proposed a 65nm 4Kb SRAM macro with algorithm-dependent IMC, which offered high throughput but required complex control logic. The compute SRAM in [16, 17] used a reconfigurable structure for bit-serial computations near peripheral circuits, enhancing analog stability but limiting throughput by activating only two rows per cycle. These studies highlight diverse IMC approaches with trade-offs in array size, process technology, power, delay, and precision.

This study introduces a novel 64×16 XNOR-SRAM architecture in 65nm CMOS technology, designed to advance IMC for DNN acceleration. Unlike prior works [10, 11, 14, 15], which connect additional transistors' drain/source terminals to SRAM bitlines, this design connects only gate terminals, significantly reducing interference during simultaneous activation of all rows. This enables full-row parallel XNOR-and-Accumulate (XAC) operations, overcoming the throughput limitations of designs like compute SRAM [16, 17], which are restricted to two-row activation. Additionally, while previous architectures [10, 11, 14, 15] employ SAs for analog-to-digital conversion, this work integrates a multi-bit flash ADC, enhancing signal conversion precision despite increased power demands. Using a 12T bitcell configuration, the proposed XNOR-SRAM optimizes computational efficiency and scalability for DNN workloads. Implemented at a 1.2 V supply, this study investigates and simulates the design, comparing its performance in power consumption and delay against existing IMC architectures to demonstrate its potential for energy-efficient, high-throughput neural network acceleration.

II. DESIGN OF THE PROPOSED ARCHITECTURE

In this work, we propose comparator designs derived from two architectures: a SA-based comparator and a conventional dynamic comparator utilizing a single tail transistor. These comparators are integrated into an SRAM-based Processing-in-Memory (PIM) architecture, originally introduced in [10] and shown in Figure 1. The proposed system consists of several key components, including a 64×16 bitcell array, a write bitline (BL) driver, a row decoder, an XNOR-mode wordline driver, and column peripheral circuits comprising a column decoder and a 16:1 analog multiplexer (MUX). Additionally, a 7-bit flash ADC is employed to digitize the analog computation results. The architecture supports dual-mode operation: memory mode and XNOR mode. In memory mode, the SRAM operates conventionally, storing and retrieving digital data. However, the innovation lies in the XNOR mode, where the system performs in-memory computation by executing bitwise XAC operations directly within the SRAM array, thereby significantly enhancing throughput and energy efficiency by minimizing data movement.

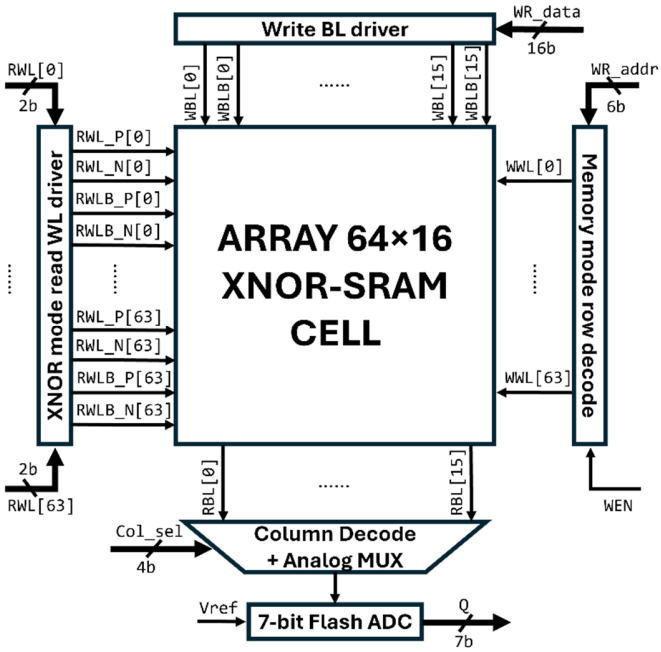

Fig. 1. Overview of the proposed architecture.

The XNOR mode is activated by a control signal following an initial precharge phase that prepares the bitlines. Once engaged, the architecture enables all 64 rows of the SRAM array to be activated simultaneously, leveraging peripheral support for parallel computation. This parallelism allows the system to perform 64 XAC operations in a single clock cycle. Bitwise XNOR operations are executed between the stored data and input vectors, with the results accumulated as an analog voltage on the read bitline (RBL). This accumulated voltage is transmitted through a transmission-gate-based analog MUX, chosen for its ability to preserve signal fidelity to the voltage ladder of the flash ADC. The ADC compares the input voltage against reference levels using a bank of comparators and

produces a thermometer code, which is then converted into a binary value via a thermometer-to-binary encoder. This binary output represents the result of the in-memory XAC computation.

III. IMPLEMENTATION OF THE PROPOSED ARCHITECTURE

A. XNOR-SRAM

In this work, we employ the 12-transistor (12T) bitcell, illustrated in Figure 2, as the foundational element for the XNOR-SRAM architecture shown in Figure 1, originally proposed in [10]. The bitcell comprises transistors T1 to T6, forming a conventional 6T SRAM cell, whereas transistors T7 to T10 function as additional pull-up and pull-down pairs that enable XNOR-mode operations. Transistors T11 and T12 serve as dynamic power gating elements, responsible for floating the RBL during idle states, thereby facilitating precharge of RBL when the read-enable (REN) signal is inactive.

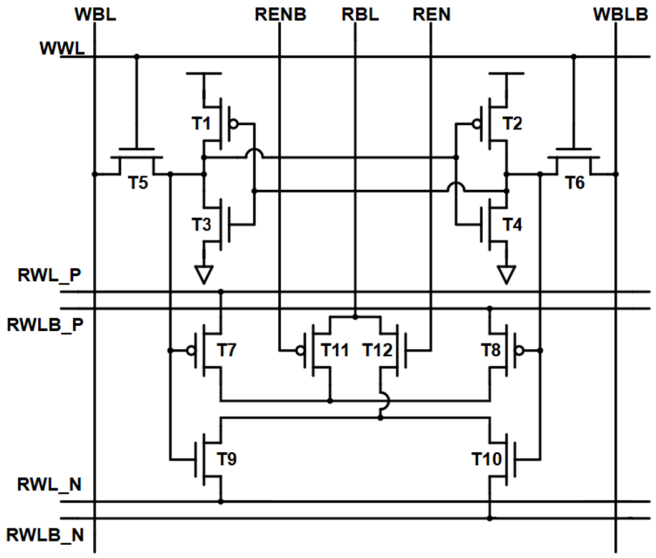

Fig. 2. Schematic of the XNOR-SRAM cell.

The input data are stored within the SRAM bitcells and subsequently processed through the XNOR driver. The XNOR driver is modulated by binary weights and a conversion signal to transform stored values into ternary representations of −1, 0, or +1. During read operations, the voltage levels across each column are accumulated and then digitized via a flash ADC, producing binary outputs that correspond to the cumulative ternary values.

In XNOR mode, the read wordline (RWL) signal activates four control signals – RWL_P, RWLB_P, RWL_N, and RWLB_N – depending on the binary or ternary input signal. For binary activations, a high XNOR output ("+1") is achieved using a strong PMOS pull-up and a weak NMOS pull-up, whereas a low output ("−1") is achieved using a strong NMOS pull-down and a weak PMOS pull-up. These operational cases are depicted in the first two rows of Figure 3. The collective PU and PD transistors in each column effectively create a resistive voltage divider between the supply voltage (V_{DD}) and ground (GND), with RBL serving as the output node. Ideally, when the PU and PD transistors are perfectly matched, the resulting RBL voltage (VRBL) exhibits a symmetric and linear relationship with the XAC value. However, fabrication-induced threshold voltage (V_{th}) variations often lead to deviations from this ideal behavior.

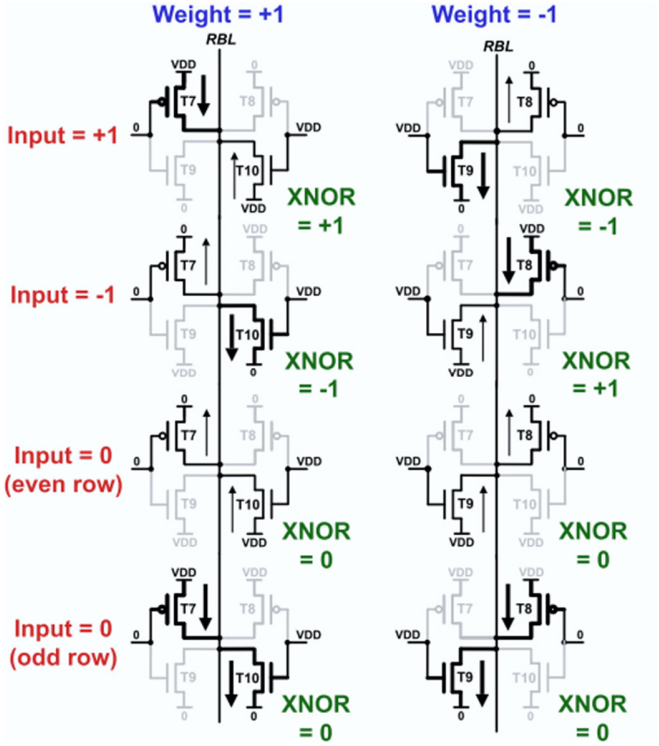

Fig. 3. XNOR-SRAM operating configuration.

The internal operation of the 6T SRAM bitcell is governed by transistors T1 to T6. During a read operation with the cell storing a logical '0', both bitlines are precharged to V_{DD}, and activation of the wordline (WL) enables T5 and T3, allowing discharge current through T5 and T3. This results in a slight voltage rise (ΔV) at the Q node. To prevent unintended bit flips, T3 must have a larger W/L ratio than T5, ensuring adequate voltage stability. A similar constraint applies to T4 and T6. In write mode, the bitlines are driven according to the intended logic value. For instance, when overwriting a stored '0' with a '1', the write bitlines are set to WBL = 1 and WBLB = 0. Upon WL activation, T2 and T6 conduct, creating a voltage divider between V_{DD} and WBLB. If T6 is sized larger than T2, the Qbar node is pulled low, enabling T1 to turn on and drive the Q node high, completing the write operation. Therefore, T6 must have a greater W/L ratio than T2, and similarly, T5 must be larger than T1. The detailed transistor sizing relationships required for correct read and write operation are summarized in Table I.

The additional circuitry supporting XNOR operations consists of transistors T7 to T12. Transistors T9 and T10 are sized sufficiently large to effectively pull down the RBL signal.

To maintain balance in the voltage divider network, T7 and T8 are designed to be twice the size of T9 and T10, respectively, compensating for the lower current drive of PMOS devices compared to NMOS of equal size. Furthermore, transistor T11 must be twice the size of T12 to maintain appropriate power gating behavior.

TABLE I. TRANSISTOR DIMENSIONS OF XNOR-SRAM

Device	Width (nm)	Length (nm)	Multiplier
T1, T2	180	60	1
T3, T4	300	60	1
T5, T6	250	60	1
T7, T8	600	60	1
T9, T10	300	60	1
T11	400	60	1
T12	200	60	1

B. *Flash Analog-to-Digital Converter*

1) *Voltage Ladder*

The voltage ladder block, as described in [18], comprises a series of equally spaced nodes connected through resistors of uniform value, forming a resistive voltage divider that partitions the reference voltage into 128 discrete levels. The reference voltage at each node:

$$V_{ref,i} = \frac{V_{ref}.(i+1).R}{128.R+R_i} \quad (1)$$

where $V_{ref,i}$ denotes the reference voltage at node i, R is the resistance of each segment in the ladder, and R_i represents the equivalent resistance seen from node i. The resulting reference levels exhibit a voltage step of approximately 9.3 mV between adjacent nodes, enabling fine-grained resolution across the reference range.

2) *Comparator*

As shown in Figure 4, the voltage comparator block in this work comprises a series of Operational Amplifiers (OPAMPs) designed to compare the input voltage V_{in} with a reference voltage V_{ref} at each node. It is assumed that when $V_{in} > V_{ref}$, the OPAMP output is logic '0', and when $V_{in} < V_{ref}$, the output is logic '1'. The proposed comparator, as illustrated in Figure 5, consists of two main components: a SA and a conventional one-tail dynamic comparator, following the designs presented in [19-22].

The operational principle of the SA is as follows: when the clock signal (CLK) is high ('1'), the complementary clock signal (CLKB) is also high, activating the precharge phase and pulling both output nodes, Q and QB, to the supply voltage V_{DD}. During this phase, the transmission gate transistors are turned off, preventing the precharge circuit from affecting the input nodes INN and INP. Simultaneously, the transistor connecting the source terminals of transistors T3 and T4 to ground is turned off, ensuring that neither Q nor QB is pulled to logic '0'. When CLK transitions to low ('0'), CLKB also goes low, disabling the precharge and leaving Q and QB floating. The transmission gates turn on at this point, linking INN and INP to Q and QB, respectively. Transistor T5 switches on, connecting the sources of T3 and T4 to VSS. If INP>INN, T3 activates earlier than T4, causing Q to be pulled down to logic '0'. Consequently, transistor T2 turns on, pulling QB up to V_{DD}. Due to the enabled transmission gates, the voltages at INN and INP propagate to Q and QB, respectively. The resulting voltages are latched by the inverter latch, assigning Q to V_{INN} and QB to V_{INP}.

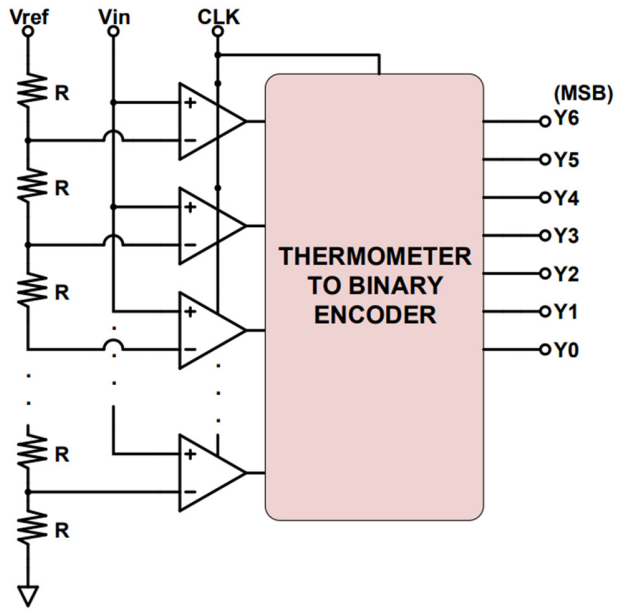

Fig. 4. Flash ADC architecture.

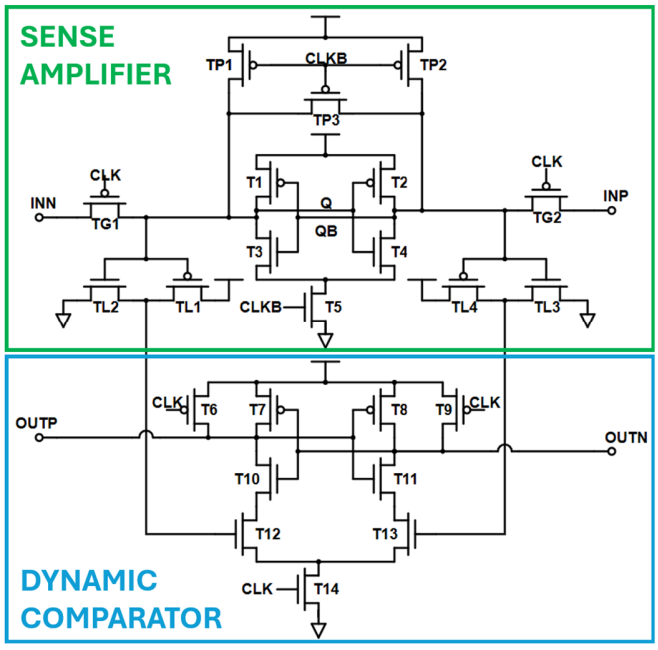

Fig. 5. Schematic of the comparator.

The dynamic comparator operates as follows: when CLK is low ('0'), transistor T14 is off, leaving the source nodes of transistors T12 and T13 floating. Although the gate voltages of

T12 and T13 are already latched, they do not influence the operation of transistors T10, T11, T8, and T7 during this phase. Transistors T6 and T9 remain on, pulling outputs OUTP and OUTN to V_{DD}. When CLK goes high ('1'), T14 turns on, grounding the source terminals of T12 and T13. Assuming $V_{INN} > V_{INP}$, T13 turns on before T12, which activates the source of T11 at logic '0'. As a result, T11 transitions from saturation to the linear region, pulling OUTN down to '0'. Since OUTN also controls the gate of T7, this causes T7 to turn on, driving OUTP to logic '1' and maintaining this state while CLK is high.

This comparator architecture offers the benefit of high sensitivity, capable of detecting voltage differences as small as 5 mV in simulations. Moreover, the fast comparison speed inherent to the one-tail dynamic comparator contributes to reduced overall delay. However, a drawback of incorporating the SA is the presence of kick-back current generated immediately after precharging, which can disturb the V_{ref}, leading to inaccurate comparisons. To mitigate this issue, additional capacitors are introduced to suppress the kick-back effect. The device dimensions corresponding to the design in Figure 5 are detailed in Table II.

TABLE II. TRANSISTOR DIMENSIONS OF COMPARATOR COMPONENTS

Device	Width (nm)	Length (nm)	Multiplier
TP1, TP2	300	60	1
TP3	200	60	1
TG1, TG2	2400	60	1
TL1, TL3	2000	60	1
TL2, TL4	1000	60	1
T5	4000	60	1
T6, T9	200	60	1
T7, T8	200	60	2
T10, T11	200	60	1
T12, T13	2000	60	1
T14	2000	60	3

3) Thermometer-to-Binary Encoder

In this work, a thermometer-to-binary encoder, as presented in [23, 24], is employed to convert the analog output values from the voltage comparison block in each column into a 7-bit digital representation. This encoded signal serves as the basis for determining the convolution value corresponding to that column. The encoder operates on the outputs of the comparator, which are expressed in thermometer code format. As illustrated in Figure 6, the encoder accepts 127 input bits, denoted as T[1:127], representing the thermometer-coded data. These inputs are processed in parallel through a series of logic gates, including NOT, AND, and OR gates, to generate a 7-bit Gray code G[0:6]. Subsequently, the Gray code is converted into a 7-bit binary code using XOR gates. This binary output represents the convolution result obtained from the corresponding SRAM column. Notably, each bit in the thermometer code influences only a single bit in the Gray code. This one-to-one mapping helps to suppress signal noise and enhances the accuracy of the final output.

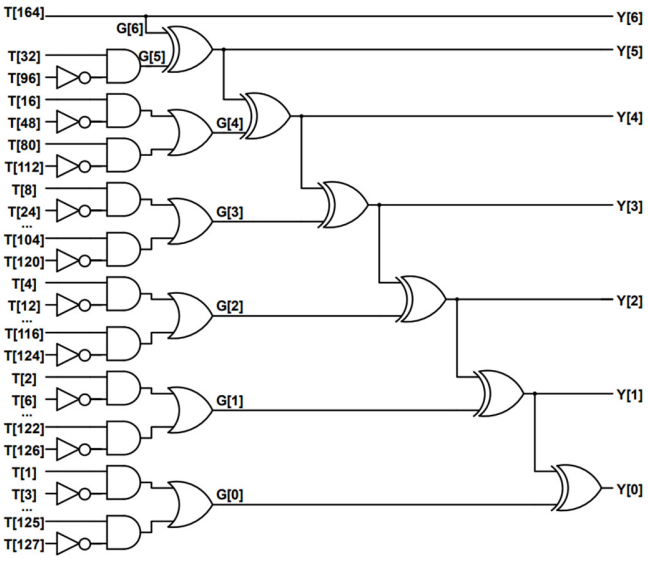

Fig. 6. Thermometer-to-binary encoder.

C. Peripherals

1) Write Bitline Driver

The write BL driver block is employed by each column within the SRAM array to control the complementary bitlines during the write operation. Prior to each write cycle, both the bitline (WBL) and its complement (WBLB) are precharged to the supply voltage. The primary function of the write driver is to pull one of the two bitlines down to ground level (logic '0') based on the input data, while the other bitline remains at V_{DD}, thereby establishing the differential signal required for a successful write operation. Various architectures for write drivers exist in the literature; however, to minimize the number of NMOS and PMOS transistors and reduce power consumption, the design adopted in this work is illustrated in Figure 7. The corresponding transistor dimensions are summarized in Table III.

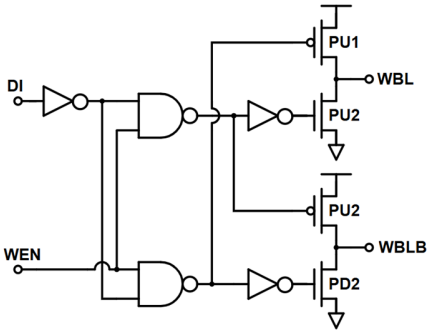

Fig. 7. Schematic of the write BL driver.

TABLE III. TRANSISTOR DIMENSIONS OF THE WRITE BL DRIVER

Device	Width (nm)	Length (nm)	Multiplier
PU1	500	60	1
PD1	2000	60	1
PU2	500	60	1
PD2	2000	60	1

The operation of the write driver is governed by the write enable signal (WEN). When WEN=0, the WBL and WBLB lines are in a high-impedance state and are connected to the precharge circuit, restoring both lines to V_{DD} in preparation for the subsequent write cycle. When WEN=1, the write driver actively drives WBL and WBLB either to logic '0' or V_{DD}, depending on the data input signal (DI). Given that each bitline is shared across 64 bitcells in a column, the associated capacitive (C) and resistive (R) loads are significant. As a result, the pull-down NMOS transistors must be sized sufficiently large to discharge the bitline effectively. Conversely, since the bitlines are precharged to V_{DD} during the idle state, the pull-up PMOS transistors do not require large drive strength and are only sized to maintain the voltage at V_{DD} during the write operation. Table IV summarizes the write driver operation.

TABLE IV. OPERATION OF THE WRITE DRIVER

WEN	DI	WBL	WBLB
0	X	Hi-Z	Hi-Z
1	0	0	V_{DD}
1	1	V_{DD}	0

In addition to the main circuit components, a precharge circuit is also employed, as shown in Figure 8. This circuit serves to precharge the voltages of the two bitlines (BL) and the offset bitline (BLB) to a high logic level (logic '1') prior to each write operation. Transistors P1 and P2 must be carefully designed to be highly symmetrical to ensure equal precharge voltages on the bitlines. Any mismatch in the bitline voltages may lead to incorrect read operations when the signals are processed by the SA. Moreover, TP1 and TP2 must be adequately sized to guarantee that the bitlines can be reliably pulled up to V_{DD} after a read or write cycle. Transistor P0 is incorporated to balance the voltage levels of the two bitlines; as it functions primarily to equalize any minor voltage discrepancies, it can be designed with a smaller size to optimize area utilization, as detailed in Table V.

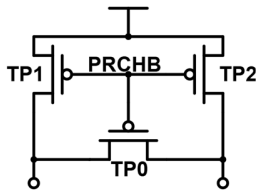

Fig. 8. Schematic of the precharge block.

TABLE V. TRANSISTOR DIMENSIONS OF THE PRECHARGE BLOCK

Device	Width (nm)	Length (nm)	Multiplier
TP0	300	60	1
TP1	600	60	1
TP2	600	60	1

During operation, when the precharge control signal (PRCHB) is active low (PRCHB=0), the bitlines BL and BLB are pulled up to V_{DD} through transistors TP1 and TP2, preparing the system for the subsequent cycle. Conversely, when PRCHB is deactivated (PRCHB=1), the bitlines are left floating, enabling read or write operations. Transistor P0 plays a critical role in maintaining voltage balance, compensating for any potential asymmetries between TP1 and TP2 that may arise due to process variations.

2) XNOR Mode Wordline Driver

The XNOR-mode read driver, illustrated in Figure 9, is employed by the rows of the SRAM array to control the RWL signal pairs during XNOR-mode operations. This driver selectively pulls one of the two RWL lines to logic '0', based on the data input to the RWL1 and RWL0 lines. Similar to the write driver circuit, each RWL line is connected to 16 bitcells along a single row, resulting in relatively large parasitic resistance and capacitance. Consequently, robust transistor sizing is required – specifically, a large NMOS transistor for effective pull-down and a correspondingly large PMOS transistor for reliable pull-up operation – to ensure the RWL lines can reach both logic '0' and V_{DD}, as summarized in Table VI.

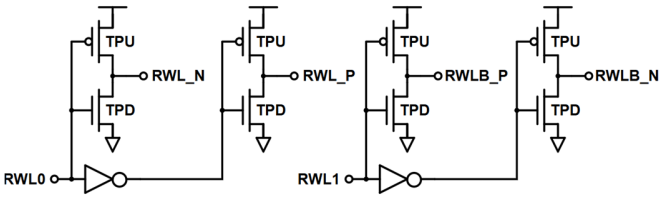

Fig. 9. Schematic of the XNOR-mode read wordline driver.

TABLE VI. TRANSISTOR DIMENSIONS OF THE XNOR-MODE READ WORDLINE DRIVER

Device	Width (nm)	Length (nm)	Multiplier
PU	2000	60	1
PD	1000	60	1

Since the RWL signal must maintain a valid logic level (either '0' or V_{DD}) throughout the entire clock cycle, the PMOS transistors are sized to be approximately twice as wide as their NMOS counterparts. This is done to compensate for the inherently slower switching speed of PMOS devices, as was discussed in a previous section. For instance, when the input data corresponds to a value of +1 (i.e., RWL[1:0] = 11), the RWL_N and RWLB_P signals are pulled down to logic '0', while the RWL_P and RWLB_N signals are pulled up to V_{DD}, thereby enabling the XNOR functionality of the circuit.

3) Row Decoder

The SRAM architecture comprises 64 rows, necessitating a 6-bit address for row selection. The row decoder block, illustrated in Figure 10, is composed of eight row decoder cells, with each cell responsible for controlling eight wordlines. This hierarchical design aims to reduce signal propagation delay, which is analyzed based on the RC delay model [25]. Each decoder cell interfaces with nine address lines. Within each cell, a decoding branch is governed by six address signals – A0, A0B, A1, A1B, A2, and A2B – which are used to select a specific wordline (WL). Additionally, three enable signals – E0, E1, and E2 – are employed to activate one of the eight

decoder cells. The corresponding WL control signal is generated through a NOR3 gate that processes the branch address signals in combination with the enable signals. When the appropriate address combination is asserted, the resulting control signal drives the WBL to logic '1', thereby enabling the WL line associated with the target bitcell row. The functional behavior of the decoder operation is detailed in Table VII.

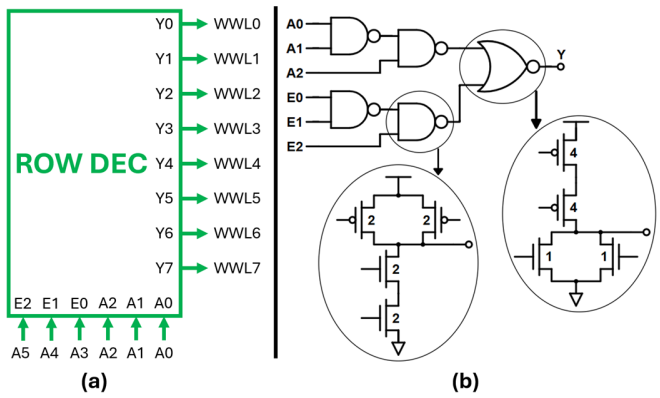

Fig. 10. Row decoder: (a) block diagram, (b) schematic.

To further mitigate signal degradation and improve timing performance, buffer stages are incorporated along the address lines, effectively enhancing signal integrity, and reducing transmission losses.

4) Column Decoder

The column decoder block, illustrated in Figure 11, is implemented using eight decoder cells, with each cell responsible for controlling eight wordlines. This hierarchical design approach is adopted to minimize signal transmission delay, which is estimated based on RC delay theory.

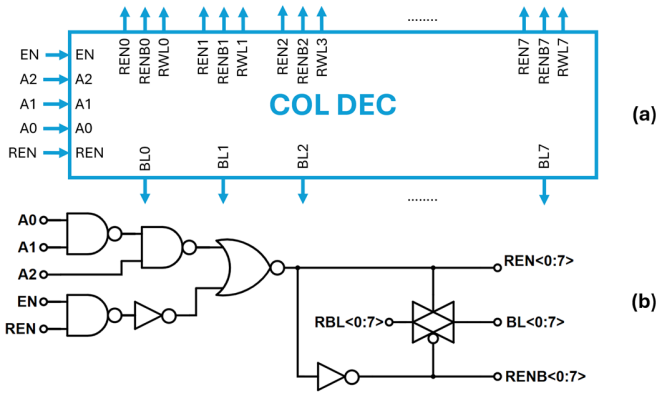

Fig. 11. Column decoder: (a) block diagram, (b) schematic.

Each decoder cell interfaces with nine address lines. Specifically, one branch of the decoder is controlled by six address signals – A0, A0B, A1, A1B, A2, and A2B – which are employed to select a specific wordline (WL). Additionally, three enable signals – E0, E1, and E2 – are utilized to activate one of the eight decoder cells. The logic control is implemented such that a NOR2 gate processes the appropriate combination of these enable signals to generate a WBL signal. When activated, this WBL signal transitions to logic '1', thereby enabling the WL line associated with the targeted bitcell row. The detailed operation of this decoding mechanism is illustrated in Table VIII.

TABLE VII. OPERATION OF THE ROW DECODER BLOCK

E2	E1	E0	A2	A1	A0	WWL0	WWL1	WWL2	WWL3	WWL4	WWL5	WWL6	WWL7
0	X	X	X	X	X	0	0	0	0	0	0	0	0
X	0	X	X	X	X	0	0	0	0	0	0	0	0
X	X	0	X	X	X	0	0	0	0	0	0	0	0
1	1	1	0	0	0	1	0	0	0	0	0	0	0
1	1	1	0	0	1	0	1	0	0	0	0	0	0
1	1	1	0	1	0	0	0	1	0	0	0	0	0
1	1	1	0	1	1	0	0	0	1	0	0	0	0
1	1	1	1	0	0	0	0	0	0	1	0	0	0
1	1	1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	1	1	0	0	0	0	0	0	0	1

TABLE VIII. OPERATION OF THE COLUMN DECODER BLOCK

EN	A2	A1	A0	BL0	BL1	BL2	BL3	BL4	BL5	BL6	BL7
0	X	X	X	X	X	X	X	X	X	X	X
1	0	0	0	RBL0	X	X	X	X	X	X	X
1	0	0	1	X	RBL1	X	X	X	X	X	X
1	0	1	0	X	X	RBL2	X	X	X	X	X
1	0	1	1	X	X	X	RBL3	X	X	X	X
1	1	0	0	X	X	X	X	RBL4	X	X	X
1	1	0	1	X	X	X	X	X	RBL5	X	X
1	1	1	0	X	X	X	X	X	X	RBL6	X
1	1	1	1	X	X	X	X	X	X	X	RBL7

To ensure signal integrity and reduce transmission losses across the address lines, buffer clusters are strategically inserted. These buffers help to maintain signal strength and mitigate degradation due to parasitic resistance and capacitance.

5) Analog Multiplexer

During each read cycle, the SRAM reads all bits located within the same column and performs a voltage-based aggregation using an XOR operation, as previously described. The resulting analog voltage signal is then transmitted to a flash ADC for digitization. To ensure that the integrity of the transmitted voltage level is preserved, a 2:1 MUX utilizing transmission gate logic is employed in the design, as detailed in [24]. Additionally, a 16:1 MUX is used to facilitate the selection of the appropriate column output for ADC conversion. This MUX is controlled by four selection signals – S0, S1, S2, and S4 – originating from the column decoder block, thereby ensuring synchronization between data reading and its subsequent transmission to the flash ADC.

The MUX architecture is composed of transmission gates, which are a fundamental type of logic element implemented using complementary PMOS and NMOS transistors connected in parallel. These transistors are driven by complementary gate control signals, denoted as S and $\overline{S}$, as illustrated in Figure 12. The MUX functions through a select signal, typically labeled S, which determines the active data path. When the select signal S is logic '0', transmission gate 1 enables the connection between input D0 and the output, while transmission gate 2 disables the path from input D1. Conversely, when S is logic '1', transmission gate 2 enables the connection from D1 to the output, and transmission gate 1 is turned off, thereby isolating D0. This mechanism ensures reliable selection and transmission of analog signals within the MUX structure. The device dimensions are detailed in Table IX.

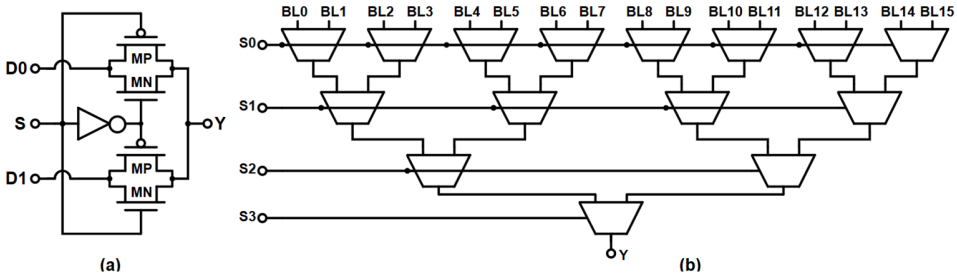

Fig. 12. Schematic of: (a) 2-to-1 MUX, and (b): 16-to-1 MUX.

TABLE IX. TRANSISTOR DIMENSIONS OF A 2-TO-1 MUX

Device	Width (nm)	Length (nm)	Multiplier
MN	3000	60	1
MP	3200	60	1

IV. RESULTS AND DISCUSSION

The proposed XNOR-SRAM architecture, designed for IMC applications, was rigorously simulated using Cadence Spectre with 65nm CMOS technology. The array comprises 64 rows and 16 columns, enabling simultaneous activation of all rows to perform 64 XNOR operations in a single clock cycle. The output signal is accumulated on the RBL line and subsequently converted into a digital format using a flash ADC.

A. XNOR-SRAM Functionality and Linearity

The correctness of the XNOR-SRAM operation was validated through waveform simulations under various ternary and binary input conditions. For instance, when the stored weight was +1, input activations of +1, 0, and -1 produced RBL voltages corresponding to logical +1, 0, and -1, respectively. The analog accumulation behavior on the RBL line demonstrated that voltage levels scaled with the XAC values between -64 and +64. Notably, the RBL voltage level showed asymmetry due to transistor mismatches and threshold voltage variation, resulting in nonlinear accumulation characteristics as shown in Figure 13. This effect is attributed primarily to process variations between PMOS and NMOS transistors in pull-up and pull-down roles.

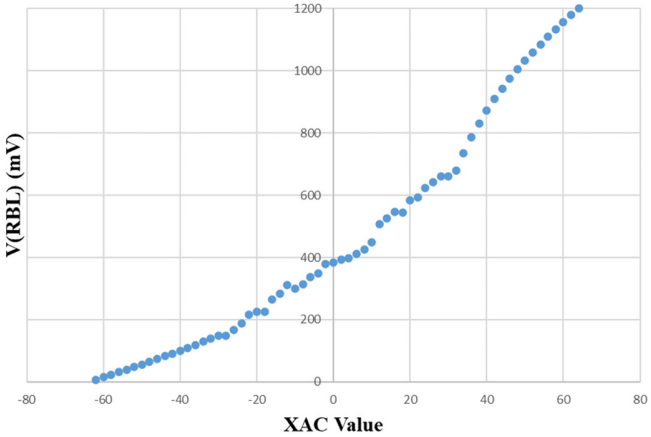

Fig. 13. Relationship between V(RBL) and XAC values.

B. Performance Evaluation

According to Figure 14 and 15, the worst-case delay – measured from the rising edge of the REN signal to the 80% rising point of the flash ADC output—was 342.67 ps. Delay varied as a function of the supply voltage, reaching 1076.31 ps at 0.6 V and reducing steadily with increasing V_{DD}, stabilizing

at nominal levels near 1.2 V. This highlights the trade-off between power supply and switching speed. Besides, power analysis revealed that the maximum average power consumption occurred when the XAC value was -64, resulting in 901.133 µW drawn from a 1.2 V supply. This peak consumption was due to full pull-down activation across 64 rows, requiring a full swing from 0 V to V_{DD} for the RBL precharge and subsequent signal propagation.

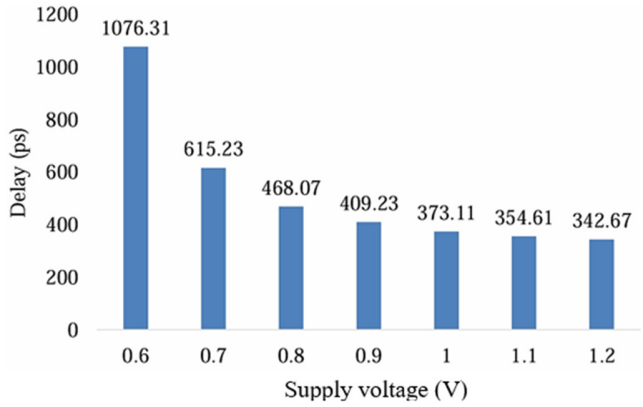

Fig. 14. Delay versus supply voltage.

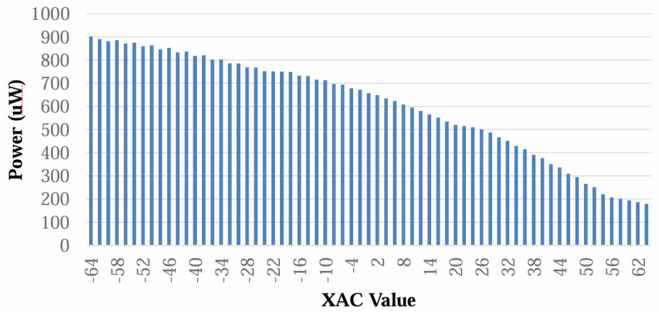

Fig. 15. Power consumption versus XAC values.

As demonstrated in Table X, a comparative analysis with existing designs underscores the strengths and trade-offs of this implementation. The XNOR-SRAM design in [10] used a 256×64 array with 6T cells and achieved a delay of 847.031 ps and 600 µW power consumption. The IMAC design [11], also based on 6T SRAM, outperformed with 311.34 ps delay and 379.01 µW power but required a larger 256×256 array and more complex periphery. A fully digital in-memory multiplier in [12] demonstrated ultra-low power (14.88 µW) but incurred a high delay of 2629.31 ps, making it less viable for high-throughput neural workloads.

TABLE X. COMPARISON OF PROPOSED AND EXISTING DESIGNS

Design	Technology	SRAM	Signal	Array	Latency (ps)	Power (µW)
This work	65nm	12T	Analog	64×16	342.67	901.133
[10]	65nm	6T	Analog	256×64	847.031	600
[11]	65nm	6T	Analog	256×256	311.34	379.01
[12]	28nm	6T	Digital	64×64	2629.31	14.88

In contrast, the 12T analog-based XNOR-SRAM architecture proposed in this study achieves competitive delay with enhanced robustness via full-row activation and flash ADC conversion. While power consumption is moderately higher, it enables higher precision and integration with analog processing pipelines. Additionally, simulations of the flash ADC and comparator circuits revealed reliable voltage-to-digital conversion across the input range. However, high-speed operation induced kickback noise that affected the reference voltage, particularly at higher RBL voltages. These disturbances led to minor errors in the thermometer-to-binary encoder stage. To mitigate such issues, future iterations may benefit from incorporating capacitive isolation or more advanced comparator topologies with reduced transient sensitivity.

V. CONCLUSION

This work proposes and simulates a 64×16 XNOR-SRAM array using 65nm CMOS technology for In-Memory Computing (IMC). The design performs 64 XNOR operations per cycle with analog accumulation and flash analog-to-digital converter (ADC)-based digital conversion. It achieves a delay of 342.67 ps and power consumption of 901.133 µW, offering high parallelism and precision. The results demonstrate the architecture's potential for efficient deep neural network acceleration.

ACKNOWLEDGMENT

We acknowledge Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for supporting this study.

REFERENCES

[1] P. Mannocci *et al.*, "In-memory computing with emerging memory devices: Status and outlook," *APL Machine Learning*, vol. 1, no. 1, Feb. 2023, Art. no. 010902, https://doi.org/10.1063/5.0136403.

[2] H. Bao *et al.*, "Toward memristive in-memory computing: principles and applications," *Frontiers of Optoelectronics*, vol. 15, no. 1, May 2022, Art. no. 23, https://doi.org/10.1007/s12200-022-00025-4.

[3] G. Pedretti and D. Ielmini, "In-Memory Computing with Resistive Memory Circuits: Status and Outlook," *Electronics*, vol. 10, no. 9, May 2021, Art. no. 1063, https://doi.org/10.3390/electronics10091063.

[4] X. Si *et al.*, "A Local Computing Cell and 6T SRAM-Based Computing-in-Memory Macro With 8-b MAC Operation for Edge AI Chips," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 9, pp. 2817–2831, Sep. 2021, https://doi.org/10.1109/JSSC.2021.3073254.

[5] N. Alnatsheh, Y. Kim, J. Cho, and K. K. Choi, "A Novel 8T XNOR-SRAM: Computing-in-Memory Design for Binary/Ternary Deep Neural Networks," *Electronics*, vol. 12, no. 4, Feb. 2023, Art. no. 877, https://doi.org/10.3390/electronics12040877.

[6] A. Gundrapally, N. Alnatsheh, and K. K. Choi, "Novel Low-Power Computing-In-Memory (CIM) Design for Binary and Ternary Deep Neural Networks by Using 8T XNOR SRAM," *Electronics*, vol. 13, no. 23, Dec. 2024, Art. no. 4828, https://doi.org/10.3390/electronics13234828.

[7] H. Tagata, T. Sato, and H. Awano, "Double MAC on a Cell: A 22-nm 8T-SRAM-Based Analog In-Memory Accelerator for Binary/Ternary

Neural Networks Featuring Split Wordline," *IEEE Open Journal of Circuits and Systems*, vol. 5, pp. 328–340, 2024, https://doi.org/10.1109/OJCAS.2024.3482469.

[8] S. Mittal, G. Verma, B. Kaushik, and F. A. Khanday, "A survey of SRAM-based in-memory computing techniques and applications," *Journal of Systems Architecture*, vol. 119, Oct. 2021, Art. no. 102276, https://doi.org/10.1016/j.sysarc.2021.102276.

[9] G. Desoli *et al.*, "16.7 A 40-310TOPS/W SRAM-Based All-Digital Up to 4b In-Memory Computing Multi-Tiled NN Accelerator in FD-SOI 18nm for Deep-Learning Edge Applications," in *2023 IEEE International Solid-State Circuits Conference*, San Francisco, CA, USA, 2023, pp. 260–262, https://doi.org/10.1109/ISSCC42615.2023.10067422.

[10] S. Yin, Z. Jiang, J.-S. Seo, and M. Seok, "XNOR-SRAM: In-Memory Computing SRAM Macro for Binary/Ternary Deep Neural Networks," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 6, pp. 1733–1743, Jun. 2020, https://doi.org/10.1109/JSSC.2019.2963616.

[11] M. Ali, A. Jaiswal, S. Kodge, A. Agrawal, I. Chakraborty, and K. Roy, "IMAC: In-Memory Multi-Bit Multiplication and ACcumulation in 6T SRAM Array," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 8, pp. 2521–2531, Aug. 2020, https://doi.org/10.1109/TCSI.2020.2981901.

[12] J. Zhang *et al.*, "In-Memory Multibit Multiplication Based on Bitline Shifting," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 2, pp. 354–358, Feb. 2022, https://doi.org/10.1109/TCSII.2021.3099798.

[13] A. Biswas and A. P. Chandrakasan, "CONV-SRAM: An Energy-Efficient SRAM With In-Memory Dot-Product Computation for Low-Power Convolutional Neural Networks," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 217–230, Jan. 2019, https://doi.org/10.1109/JSSC.2018.2880918.

[14] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A Mixed-Signal Binarized Convolutional-Neural-Network Accelerator Integrating Dense Weight Storage and Multiplication for Reduced Data Movement," in *2018 IEEE Symposium on VLSI Circuits*, Honolulu, HI, USA, 2018, pp. 141–142, https://doi.org/10.1109/VLSIC.2018.8502421.

[15] W.-S. Khwa *et al.*, "A 65nm 4Kb algorithm-dependent computing-in-memory SRAM unit-macro with 2.3ns and 55.8TOPS/W fully parallel product-sum operation for binary DNN edge processors," in *2018 IEEE International Solid-State Circuits Conference*, San Francisco, CA, USA, 2018, pp. 496–498, https://doi.org/10.1109/ISSCC.2018.8310401.

[16] J. Wang *et al.*, "14.2 A Compute SRAM with Bit-Serial Integer/Floating-Point Operations for Programmable In-Memory Vector Acceleration," in *2019 IEEE International Solid-State Circuits Conference*, San Francisco, CA, USA, 2019, pp. 224–226, https://doi.org/10.1109/ISSCC.2019.8662419.

[17] J. Seo *et al.*, "A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *2011 IEEE Custom Integrated Circuits Conference*, San Jose, CA, USA, 2011, pp. 1–4, https://doi.org/10.1109/CICC.2011.6055293.

[18] Z. Wei, R. Qin, and F. You, "A 6-bit 2GS/s Flash ADC with Low Power Consumption," in *2022 International Conference on Microwave and Millimeter Wave Technology*, Harbin, China, 2022, pp. 1–3, https://doi.org/10.1109/ICMMT55580.2022.10023386.

[19] J. R. Rusli, S. Shafie, R. M. Sidek, H. A. Majid, W. Z. W. Hassan, and M. A. Mustafa, "Optimized low voltage low power dynamic comparator robust to process, voltage and temperature variation," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 17, no. 2, pp. 783–792, Feb. 2020, https://doi.org/10.11591/ijeecs.v17.i2.pp783-792.

[20] B. Goll and H. Zimmermann, "A Comparator With Reduced Delay Time in 65-nm CMOS for Supply Voltages Down to 0.65 V," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 56, no. 11, pp. 810–814, Nov. 2009, https://doi.org/10.1109/TCSII.2009.2030357.

[21] B. Wicht, T. Nirschl, and D. Schmitt-Landsiedel, "Yield and speed optimization of a latch-type voltage sense amplifier," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 7, pp. 1148–1158, Jul. 2004, https://doi.org/10.1109/JSSC.2004.829399.

[22] M. Kavitha, S. Akhila, and A. Kannan, "Design and Implementation of a Second Order Continuous-Time ΣΔ Modulator for ECG Signal Acquisition," *Engineering, Technology & Applied Science Research*, vol. 13, no. 1, pp. 10128–10133, Feb. 2023, https://doi.org/10.48084/etasr.5567.

[23] S. Kumar, M. K. Suman, and K. L. Baishnab, "A novel approach to thermometer-to-binary encoder of flash ADCs-bubble error correction circuit," in *2014 2nd International Conference on Devices, Circuits and Systems*, Coimbatore, India, 2014, pp. 1–6, https://doi.org/10.1109/ICDCSyst.2014.6926213.

[24] Y. Gupta and S. Saini, "Thermometer to Gray Encoders," in *Performance Optimization Techniques in Analog, Mixed-Signal, and Radio-Frequency Circuit Design*, M. Fakhfakh, E. Tlelo-Cuautle, M. E. Fino, and S. Saini, Eds. Hershey, PA, USA: IGI Global Scientific Publishing, 2015, pp. 323–335, https://doi.org/10.4018/978-1-4666-6627-6.ch013.

[25] B. L. Dokic, "A Review on Energy Efficient CMOS Digital Logic," *Engineering, Technology & Applied Science Research*, vol. 3, no. 6, pp. 552–561, Dec. 2013, https://doi.org/10.48084/etasr.389.

AUTHORS PROFILE

Doanh Bui is a master student who also received his B.S. degree in Electronics and Telecommunications Engineering from Ho Chi Minh City University of Technology (HCMUT), VNU-HCM, in 2021. His research interests include CMOS digital circuit design, physical design in IC design.

Phuoc-Luan Vo is currently pursuing a B.S. degree in electrical and electronics engineering. His research interests include CMOS digital circuit design, physical design in IC design.

Phuc T. Nguyen-Phan is a master student who also received his B.S. degree in Electronics and Telecommunications Engineering from Ho Chi Minh City University of Technology (HCMUT), VNU-HCM, in 2023. His research interests include digital circuit design, memristive devices, non-volatile memory, and neuromorphic computing.

Linh Tran received the B.S. degree in Electrical and Computer Engineering from the University of Illinois, Urbana – Champaign (2005), M.S. and PhD. in Computer Engineering from Portland State University (2006, 2015). Currently, he is working as a lecturer at the Faculty of Electrical-Electronics Engineering, Ho Chi Minh City University of Technology – VNU HCM. His research interests include quantum/reversible logic synthesis, computer architecture, hardware-software co-design, efficient algorithms and hardware design targeting FPGAs, and deep learning.