A Fully Digital SRAM-Based Four-Layer In-Memory Computing Unit Achieving Multiplication Operations and Results Store

Zhiting Lin, *Senior Member, IEEE*, Shaoying Zhang, Qian Jin, Jianping Xia, Yunwei Liu, Kefeng Yu, Jian Zheng, Xiaoming Xu, Xing Fan, Ke Li, Zhongzhen Tong, Xiulong Wu, Wenjuan Lu, Chunyu Peng, *Member, IEEE*, and Qiang Zhao, *Member, IEEE*

***Abstract*— The separation of memory and arithmetic logic unit (ALU) in the von Neumann computing architecture hinders the development of big data and high-performance computing. In-memory computing (IMC) as a new computation method significantly reduces the latency and power consumption of data processing. In this study, we propose a fully digital static random access memory (SRAM)-based IMC architecture, which has the following advantages: 1) it simplifies multiplication to multicycle addition operations, reuses logic cells, and reduces hardware overhead; 2) by adding a pair of nMOS transistors to achieve internal write-back, the computational efficiency is improved, and at the same time, the final result of the multiplication can be stored locally, eliminating the need to read the computational result immediately; and 3) this scheme can be easily expanded to multiplication operations with different bit widths, which provides good scalability. A 4-kb SRAM-IMC macro chip is manufactured using the SMIC 55-nm technology to realize 4-bit multiplication, with an energy efficiency of 51.4 TOPS/W (0.9 V) and a throughput of 234.3 GOPS/mm^2. The proposed multiplication–accumulation architecture is applied to a neural network, which achieves 98.7% accuracy with the Mixed National Institute of Standards and Technology database (MNIST) dataset.**

***Index Terms*— In-memory computing (IMC), multiplication operation, static random access memory (SRAM), von Neumann bottleneck, write-back operation.**

I. Introduction

In THE conventional von Neumann computing architecture, the memory is separated from the arithmetic logic unit (ALU), as shown in Fig. 1(a), and large amounts of data are passed back and forth between the two when performing operations, resulting in excessive energy consumption and latency. This problem is referred to as the “memory wall.” The rapid development of artificial intelligence (AI), biological systems, neural networks, and other technologies in recent years has brought this problem to the forefront. Therefore, there is a need to develop new solutions to solve or mitigate this problem. An emerging in-memory computing (IMC) paradigm has attracted much attention in the scientific community. It overcomes the throughput and energy limitations of existing von Neumann architectures [1], as shown in Fig. 1(b), in which data operations and processing are combined in a memory array, reducing data transfer, lowering energy consumption, and improving computational efficiency. Currently, IMC is widely applied to machine learning algorithms [2], [3], sparse distributed memory [4], [5], sensor-rich platforms, self-driving cars, and the Internet of Things.

The emergence of IMC methods can be divided into analog IMC [6], [7], [8], [9], [10], [11], [12], [13], [14], [15] and digital IMC [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34]. Analog IMC may not be suitable for high-precision applications because it has the disadvantage of low conversion accuracy limited by the low-cost analog-to-digital converters (ADCs), while digital IMC has the advantage of high computational accuracy.

Currently, there are two types of approaches in digital IMC. One is to integrate Boolean arithmetic logic circuitry around the memory array to achieve multiple types of basic logic operations [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], such as NAND, AND, OR, and NOR, by distinguishing the discharge current or voltage formed by the simultaneous activation of multiple cells on the bitline. Further logic circuits are connected to achieve complex arithmetic logic. However, because the bitline discharge is susceptible to process, voltage, and temperature (PVT), the accuracy of logic operations is reduced, limiting the benefits of IMC.

The other approach is to closely combine logic cells with static random access memory (SRAM) cells [26], [27], [28], [29], [30], [31], [32], [33], [34]; however, this method can usually perform only single-bit operations. To achieve more complex operations, additional

Manuscript received 4 December 2022; revised 5 March 2023; accepted 6 April 2023. Date of publication 26 April 2023; date of current version 23 May 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62074001, in part by the State Key Program of the National Natural Science Foundation of China under Grant 61934005, in part by the National Key Research and Development Program of China under Grant 2018YFB2202602, and in part by the Joint Funds of the National Natural Science Foundation of China under Grant U19A2074. *(Corresponding author: Xiulong Wu.)*

Zhiting Lin and Xiulong Wu are with the School of Integrated Circuits, Anhui University (AHU), Hefei 230601, China, and also with the Anhui Provincial High-Performance Integrated Circuit Engineering Research Center, Hefei 230601, China (e-mail: xiulong@ahu.edu.cn).

Shaoying Zhang, Qian Jin, Jianping Xia, Yunwei Liu, Kefeng Yu, Jian Zheng, Xiaoming Xu, Xing Fan, Ke Li, Wenjuan Lu, Chunyu Peng, and Qiang Zhao are with the School of Integrated Circuits, Anhui University (AHU), Hefei 230601, China.

Zhongzhen Tong is with the School of Integrated Circuit Science and Engineering, Beihang University, Beijing 100191, China.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2023.3266651.

Digital Object Identifier 10.1109/TVLSI.2023.3266651

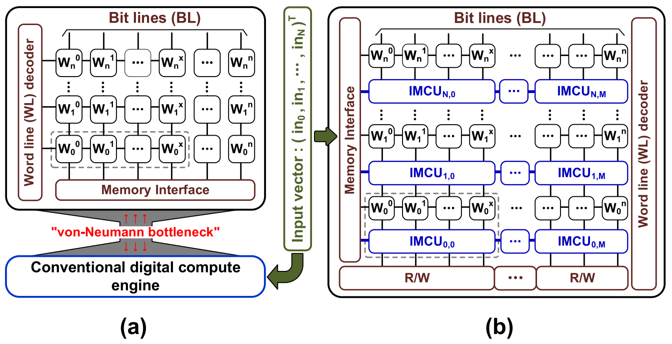


Fig. 1. (a) Conventional von Neumann architecture with distinct memory and processing units. In order to perform calculations, the matrix elements stored in the memory cells must be transferred to physically separate digital computing cells. This costs time and energy and is an inherent performance bottleneck. (b) In the IMC system, the processing unit is embedded in the memory array, the matrix elements stored in the SRAM cells remain unchanged, and data processing takes place in the IMCU.

arithmetic logic embedded in the array is required for real-time processing. In particular, the multiplication of multiple bits also requires a shift operation, which requires redundant cells to store the data in memory. After the multiplication result is obtained, the intermediate results of the calculation in the redundant cells are often useless and occupy storage space. Meanwhile, the write-back operation is implemented by the peripheral write circuit, which reduces the efficiency of IMC.

To reduce computational complexity and optimize the arithmetic operation while improving memory cell use at the same time, we propose a fully digital IMC that can reuse memory space. The advantages of the proposed structure over the existing IMC architecture are given as follows.

1) The multicycle input activation scheme reuses logic units, which reduces the overhead.
2) Internal write-back is implemented in the IMC unit (IMCU), which improves the efficiency. Multiplication results are stored locally so that there is no need to read the results immediately.
3) The idea provided by this scheme can be easily extended to multiplications with different bit widths, which allows good scalability.

The remainder of this article is organized as follows. Section II introduces the related work of SRAM IMC. Section III analyzes how to optimize the multiplication scheme, proposes the corresponding IMC macro circuit structure to realize scalable multiplication, and illustrates the working principle and process of the circuit with examples. Section IV presents the results. Section V describes how to configure the structure to realize the CNN algorithm. Finally, Section VI provides the conclusions.

II. Related Work

As mentioned earlier, to solve the problem of energy efficiency and latency, IMC has received a lot of attention. SRAM is widely used in IMC because of its faster write speed and lower write energy in memory. SRAM-IMC can be divided into analog IMC and digital IMC.

A. Analog IMC

In general, analog IMCs have relatively low computational accuracy and are suitable for applications that require less accuracy, such as deep neural network inference. Many analog IMCs are commonly used in fully connected networks (FNS) and convolutional neural networks (CNN) [8], [15], [36]. MAC computations include the multiplication of inputs, weights, and the accumulation of multiplication results. The weight data are generally stored in SRAM arrays, and external inputs activate word lines in different ways to control the SRAM cells. Depending on the amount of bitline discharge, different calculation results are obtained.

For example, in [7], the external input is converted into the corresponding number of pulses to activate the word line to control the SRAM discharge. However, when the input precision increases, the number of pulses increases, thus increasing the computation delay per MAC cycle. On the other hand, the input narrow pulses tend to distort with the word line transmission, which affects the SRAM cell discharge consistency. In [35], the external input is reflected by controlling the width of the word line pulses, and this scheme still increases the computation delay per MAC cycle. The reason is that as the input accuracy increases, the pulsewidth length increases accordingly and the computation period depends on the maximum pulsewidth. Although the computation period can be reduced by characterizing the external activation as word line voltage height [1] or bitline voltage height [8], the effective voltage value must be controlled between the threshold voltage V_{th} and the supply voltage V_{DD}, which will limit the input accuracy in practice.

The efficiency of the analog IMC lies in the multirow reading. However, if the bitline voltage drops to a certain level will cause the cell originally stored 1 to be rewritten to 0. This problem is called "read disturb," which can be solved by decoupling the read port by adding transistors to separate the read and write paths [14], [36]. However, this method requires changing the structure of the memory cell and increasing the number of transistors in the bit cell.

Achieving higher output accuracy in MAC computation requires high-quality ADC circuits. It requires more transistors and higher capacitance to support high-accuracy conversion. High-precision calculations with analog IMC require increased energy consumption and area. It will eventually diminish the advantages of efficient analog IMCs.

B. Digital IMC

In simple logic operations, usually, two rows of word lines are turned on simultaneously to obtain data stored in the same column as input, and then, a read-out circuit is used to obtain the output of the logic operation. These operations are based on bitline discharges and are susceptible to PVT. When complex operations need to be implemented, more peripheral digital logic units are required. For example, in [26], the addition and multiplication operations are realized through the cooperation of gate circuits and flip-flops. However, multiple external write-back operations require complex timing control and also increase the computational delay. In addition, shift

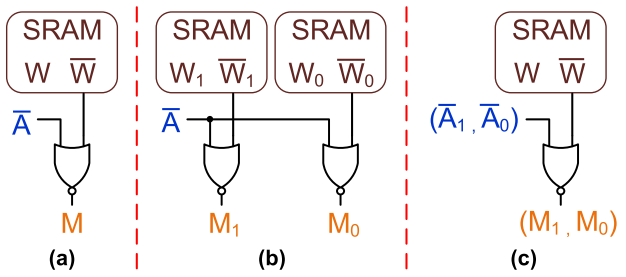


Fig. 2. (a) Dot-product operation with single-bit input and single-bit weights. (b) Dot-product operation with single-bit input and multibit weights. (c) Dot product of multibit input and single-bit weights.

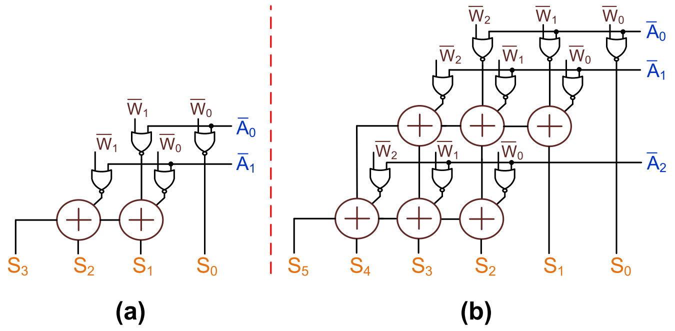


Fig. 3. Combinational logic circuit for (a) 2-bit and (b) extended to 3-bit multiplication.

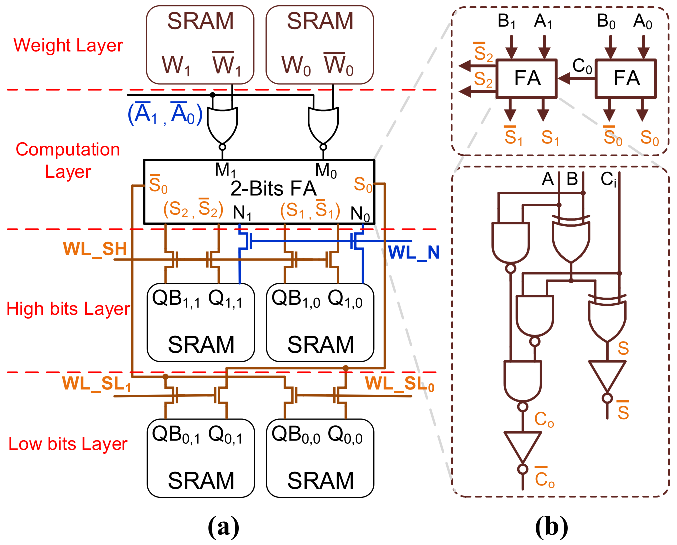


Fig. 4. (a) Circuit implementation of a multiplication scheme with integrated storage and computation. (b) Structure of adders in the computation layer.

operations are performed in arrays causing a waste of storage space.

Some fully digital architectures implement multiplication operations by embedding digital logic units in a memory array. The architecture in [34] uses a bit-serial input method and a parallel adder tree to implement massively parallel multiplication operations. The accumulation module around the array is used to accumulate the multiplication results. This architecture with both adder trees and accumulation modules has a large area overhead. The unique bit cell-based and bit-serial computation architecture in [33] has some advantages in area, but it needs to wait for the previous vertical ripple carry adder to get the correct accumulation result before it can guarantee the correct result of the horizontal ripple carry adder, which greatly increases the circuit latency. To trade off the computational accuracy, area overhead, and latency, we propose a fully digital IMCU to overcome the current challenges in analog and digital accelerators.

III. Implementation of Scalable Decomposition Computation

Implementing dot-product operations is the basis for multiplication operations, and AND logic gate circuits are the simplest form of implementing dot-product operations. When applied to a digital IMC, a NOR gate with four MOS transistors is a better solution because the SRAM cell can provide the reverse input. As shown in Fig. 2(a), one operand, $\bar{W}$, is input from the reverse side of the 6T SRAM cell, and the other operand, A, is input from the outside in the form of an inverse code; then, the output M of the NOR is the result of the $W \cdot A$ operation. On this basis, the multiplication result of single-bit input A and multibit weights W_1 and W_0 can be realized as M_1 and M_0, as shown in Fig. 2(b).

It is also possible to implement multiplication where the external input is multibit data. Let us take 2-bit input data A_1A_0 as an example. Input the low bit A_0 first; the NOR gets the dot-product result M_0. Next, input the high bit A_1; the NOR gets another result M_1 so that M_1M_0 is the final result of the operation, as shown in Fig. 2(c).

When both the internal weights and external inputs are multibit data, a shift is required. Let us take 2-bit data as an example. The operand in SRAM is W_1W_0 and the external input is A_1A_0. When the low bit A_0 is input, the operation result $M_{10}M_{00}$ is obtained, and when the high bit A_1 is input, the operation result $M_{11}M_{10}$ is obtained, and it is necessary to move one bit to the left to reflect the increase in weight. For the final result, the results of the two operations must be added. The above operations can be expressed by the combined logic circuit, as shown in Fig. 3(a). However, if the weighting and bit width of the external input continue to increase, then the sum of a single group of adders cannot provide the multiplication result. Fig. 3(b) shows a schematic of the combinational logic required to realize the 3-bit multiplication; it can be seen that the circuit overhead also increases accordingly. As the bit width continues to increase, the required circuit overhead also continues to increase. For this reason, we proposed a reusable multiplication scheme to improve the resource utilization of IMC.

A. Proposed Multiplication Circuit

To implement the above process inexpensively in memory, we proposed a reusable scheme. For clarity, Fig. 4 shows a 2-bit multiplication circuit. The entire circuit is divided into four layers, from top to bottom, the 2-bit weight layer, the 2-bit computation layer, the 2-bit high bit layer, and the 2-bit low bit layer. The inputs of the adder come from the NOR gate and the high bits layer, and the outputs of the adder are stored in the high and low bits layers. To ensure the reliability of the store operation, a double-ended write operation is used.

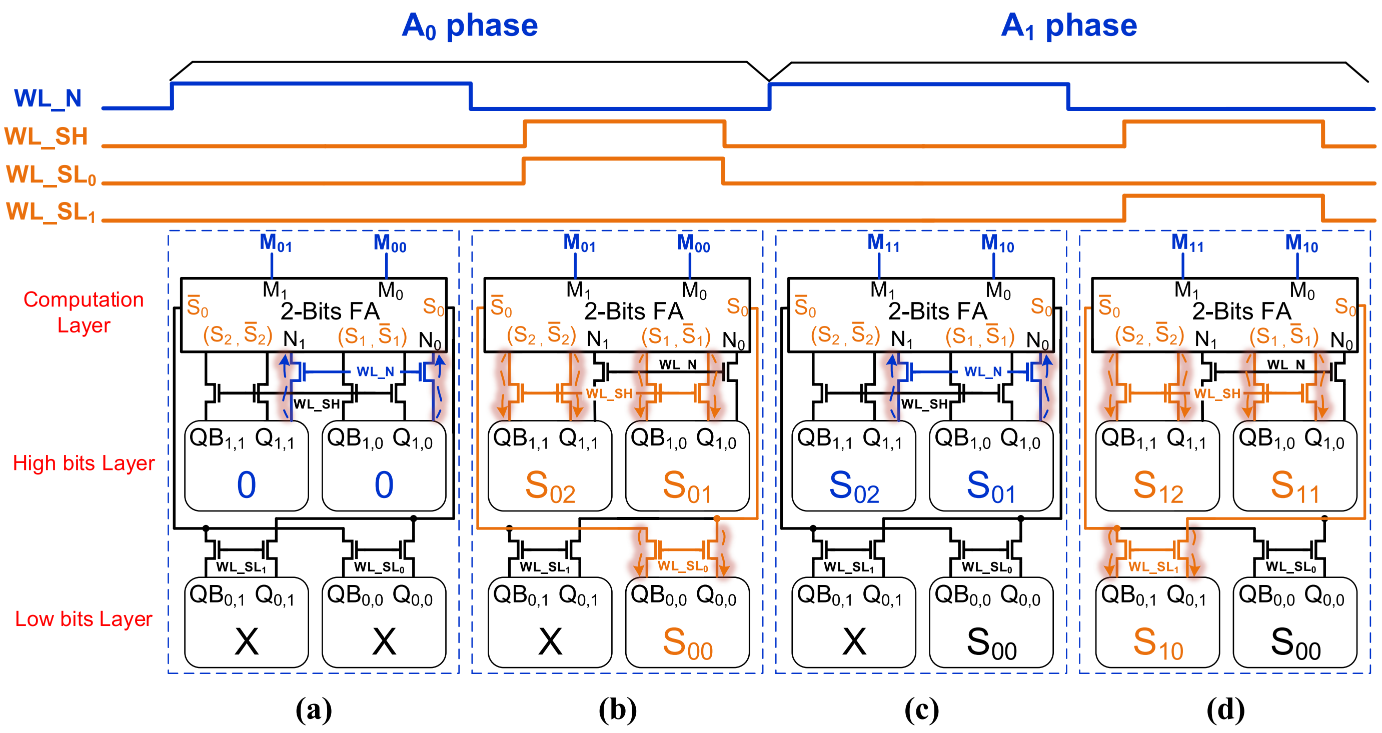


Fig. 5. (a) Calculation in A_0 phase: WL_N = 1, and the input of the adder is provided by the high bits layer. (b) Write-back in A_0 phase: signal WL_SH = 1, WL_SL$_0$ = 1, and the outputs of adder are written to the high and low bits layers. (c) Calculation in A_1 phase: signal WL_N = 1, data are provided by the high bits layer to the input of the adder. (d) Write-back in A_1 phase: signals WL_SH = 1 and WL_SL$_1$ = 1, and the outputs of adder are written to the high and low bits layers.

Therefore, a pair of nMOS transistors are added to the memory cell in high and low bits layers, which are controlled by the local word line signals WL_SH, WL_SL$_1$, and WL_SL$_0$. The high bits layer provides the input signal of the adder and receives the output signal of the adder. When the data in the high bits layer are updated, the input signal of the adder is changed immediately, resulting in incorrect output data. Therefore, by adding an nMOS transistor controlled by the signal WL_N to the high bits layer, the calculation and storage are divided into two stages controlled by WL_N and WL_SH.

In the computation layer, the 2-bit adder consists of two full adders connected in series. Since the SRAM cells in the high and low bits layers are to be written double-ended, the output of the adder includes forward output and reverse output, and the full adder structure is shown in Fig. 4(b). It is worth noting that a low-overhead full adder can be used to achieve the function [34], [37], and the proposed architecture has no limitation on adders.

Fig. 5 shows the detailed process of the above circuit to perform a 2-bit multiplication operation. Before executing the operation, the 2-bit weight data W_1W_0 are stored in the weight layer, and the data "00" are prestored in the high bits layer. The external input A_1A_0 is input in two phases.

1) In the A_0 phase, it is divided into two steps, i.e., calculation and write-back. The calculation is shown in Fig. 5(a). WL_N = 1 and the output of the NOR gate is $M_{01}M_{00}$. The adder sums $M_{01}M_{00}$ and 00 to obtain the calculation result $S_{02}S_{01}S_0$. Next, the write-back of the intermediate calculation result is performed, as shown in Fig. 5(b). The output of the adder plays the role of a local bitline, and the write-back operation is just like the write operation of a 6T cell. Turn on the local word line, i.e., WL_SH = 1 and WL_SL$_0$ =1, while WL_N = 0; the data $S_{02}S_{01}$ are written back to the high bits layer and S_{00} is written back to the least significant bit (LSB) of the low bits layer.
2) In phase A_1, the procession is similar to that in phase A_0. First, the calculation is performed, as shown in Fig. 5(c). WL_N = 1, and the output of the NOR gate is $M_{11}M_{10}$. The adder sums $M_{11}M_{10}$ and $S_{02}S_{01}$ to obtain the calculation result $S_{12}S_{11}S_{10}$. Next, the write-back of the calculation result is performed, as shown in Fig. 5(d). When WL_N = 0, WL_SH = 1, and WL_SL$_1$ = 1; the data $S_{12}S_{11}$ are written back to the high bits layer and S_{10} is written to the most significant bit (MSB) of the low bits layer. Finally, the 4-bit results of multiplication operation $S_{12}S_{11}S_{10}S_{00}$ are stored in the high and low bits layers. These two layers can store intermediate results as well as the final calculation results. In summary, the proposed method can effectively reduce the hardware cost of the multiplication operation and improve the utilization of memory cells.

B. Multiplication Circuit Expanded to n-Bit

This scheme can be easily extended to an n-bit multiplication operation with acceptable cost. Fig. 6 shows the structure of an n-bit multiplication operation. Before the calculation starts, the n-bit weights, $W_{n-1}, \ldots, W_1W_0$, are written to the weight layer, and the high bits layer is prestored with

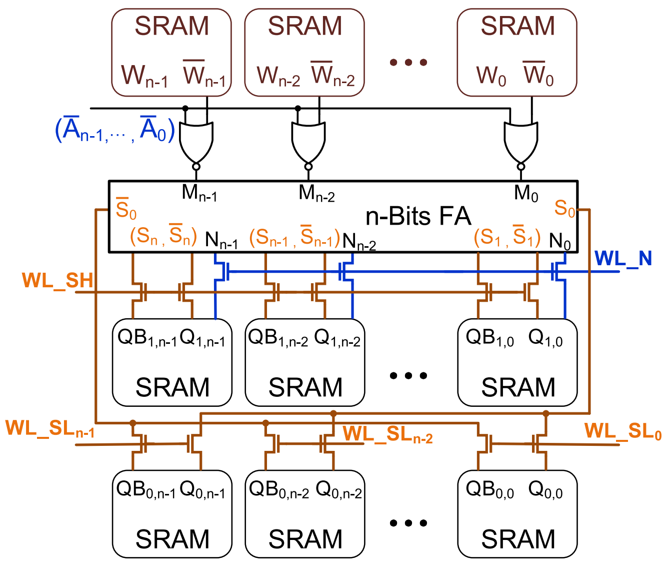

SRAM
W_{n-1} $\overline{W}_{n-1}$
W_{n-2} $\overline{W}_{n-2}$
W_0 $\overline{W}_0$
$(\overline{A}_{n-1},\cdots,\overline{A}_0)$
M_{n-1} M_{n-2} M_0
n-Bits FA
$\overline{S}_0$ S_0
$(S_n, \overline{S}_n)$ N_{n-1} $(S_{n-1}, \overline{S}_{n-1})$ N_{n-2} $(S_1, \overline{S}_1)$ N_0
WL_N
WL_SH
$QB_{1,n-1}$ $Q_{1,n-1}$ $QB_{1,n-2}$ $Q_{1,n-2}$ $QB_{1,0}$ $Q_{1,0}$
WL_SL_{n-1} WL_SL_{n-2} WL_SL_0
$QB_{0,n-1}$ $Q_{0,n-1}$ $QB_{0,n-2}$ $Q_{0,n-2}$ $QB_{0,0}$ $Q_{0,0}$

Fig. 6. Structure of the n-bit multiplication and storage solution.

"0, ..., 00." During the calculation, the external n-bit bit-width data $A_{n-1}, \ldots, A_1 A_0$ are input sequentially starting from the lowest bit A_0.

For each external bit input, two steps are performed: calculation and write-back. The timing of the operation is shown in Fig. 7(a). In the A_0 phase, the first calculation is performed only with the signal WL_N = 1. The output of the NOR gate is $M_{0,n-1}, \ldots, M_{01} M_{00}$. The adder sums $M_{0,n-1}, \ldots, M_{01} M_{00}$ and $0, \ldots, 00$ to obtain the calculation result $S_{0,n}, \ldots, S_{01} S_{00}$ and then stores the calculation result. Let WL_N = 0, WL_SH = 1, and $WL\_SL_0 = 1$. The data $S_{0,n}, \ldots, S_{02} S_{01}$ are written back to the high bits layer and S_{00} is written back to the LSB of the low bits layer. In phase A_1–A_{n-1}, the signals WL_N and WL_SH are the same as in phase A_0, as shown in Fig. 7(b) and (c). When writing back to the low bits layer, the signal WL_SL is changed with the phase, e.g., in phase A_1, $WL\_SL_1 = 1$, in phase A_2, $WL\_SL_2 = 1, \ldots,$ and so on, as shown in Fig. 7(e). Fig. 7(d) shows the data changes in the high and low bits layers throughout the procession.

C. Numerical Example

A digital example of 4-bit multiplication is given in the following to demonstrate the operation of the multiplication circuit. Before the operation starts, the weight layer stores [0, 1, 1, 0] and the high bits layer is "0000." Then, the external 4-bit data [1, 1, 0, 1] are input sequentially in four phases, starting with the lowest bit. The timing of the whole calculation process and the changes in the data in the high and low bits layers are shown in Fig. 8.

In the $A_0 = 1$ phase, when WL_N = 1, the dot product and summation are executed in the calculation layer, which can be expressed by the following equation:

$$[0, 1, 1, 0] \cdot 1 + [0, 0, 0, 0] = [0, 0, 1, 1, 0].$$

Then, the calculation result is written back so that the local word line WL_SH = 1 and $WL\_SL_0 = 1$, then the high bits layer data changes from [0, 0, 0, 0] to [0, 0, 1, 1] and the low bits layer changes to $[x, x, x, 0]$.

In the $A_1 = 0$ phase, the computation can be expressed as follows:

$$[0, 1, 1, 0] \cdot 0 + [0, 0, 1, 1] = [0, 0, 0, 1, 1].$$

Then, let the local word line WL_SH = 1 and $WL\_SL_1 = 1$, and thus, the high bits layer data changes from [0, 0, 1, 1] to [0, 0, 0, 1] and the low bits layer changes to $[x, x, 1, 0]$.

In the $A_2 = 1$ phase, the computation can be expressed as follows:

$$[0, 1, 1, 0] \cdot 1 + [0, 0, 0, 1] = [0, 0, 1, 1, 1].$$

Let the local word line WL_SH = 1 and $WL\_SL_2 = 1$, and thus, the high bits layer data [0, 0, 0, 1] become [0, 0, 1, 1] and the low bits layer data become $[x, 1, 1, 0]$.

In the $A_3 = 1$ phase, the computation is expressed as

$$[0, 1, 1, 0] \cdot 1 + [0, 0, 1, 1] = [0, 1, 0, 0, 1].$$

Let the local word line WL_SH = 1 and $WL\_SL_3 = 1$, and thus, the data of the high and low bits layers are updated to [0, 1, 0, 0] and [1, 1, 1, 0], respectively.

Finally, the stored data in the high and low bits layers are given as follows:

$$[S_7, S_6, S_5, S_4, S_3, S_2, S_1, S_0] = [0100, 1110].$$

D. System Architecture

In this study, a four-layer MCU is proposed. The high bits layer consists of 9T cells, and the low bits layer consists of 8T cells. Fig. 9 shows the overall architecture, including the IMCU array for multiplication, the decoder circuit, the switch circuit for data driving, the sensitive amplifier (SA) circuit for data reading, and the mode control circuit for switching different modes.

The proposed circuit architecture supports two modes of operation, namely, SRAM mode and IMC mode. In the SRAM mode, the SRAM cells in the weight layer, high bit layer, and low bit layer of the IMCU can perform regular data access, consistent with the basic SRAM memory. As shown in Fig. 10(a), the timing module generates the read and write signal to activate the corresponding word lines achieving read and write operation. The advantage of IMC mode is the parallelism of operations, i.e., multiple rows can be operated at the same time. Before performing multiplication operations, it is necessary to write the weight data to the weight layer of IMCU in the SRAM mode and prestore the data "0" in the high bit layer. To improve the efficiency, it is possible to activate the rows that need to be prestored the data "0" simultaneously. The circuit is then switched to the IMCU mode by a mode control signal. As shown in Fig. 10(b), after receiving the mode control signal, the corresponding timing signals are generated and these signals will control the IMCU to operate, in which the external data are driven to the computation layer sequentially from low to high.

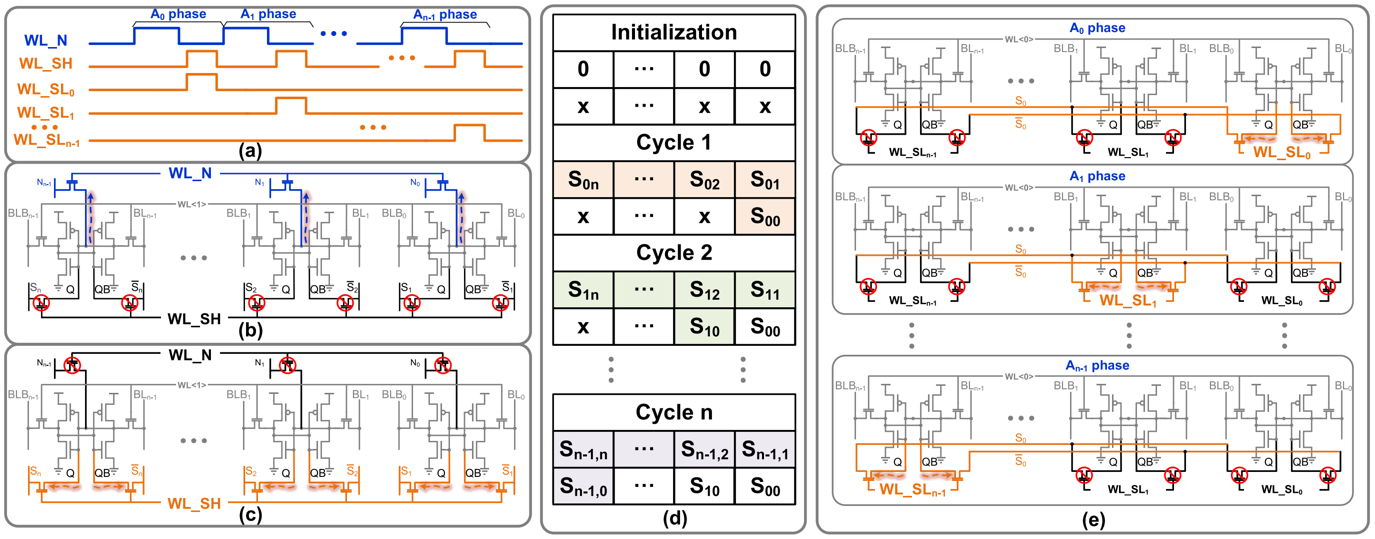


Fig. 7. Computation operation and write-back operation of n-bit multiplication: (a) operation timing diagram, (b) data flow of high bits layer during computation, (c) data flow of high bits layer during write-back, (d) data update process of high and low bits layers in each stage, and (e) data flow of low bits layer during write-back in each phase.

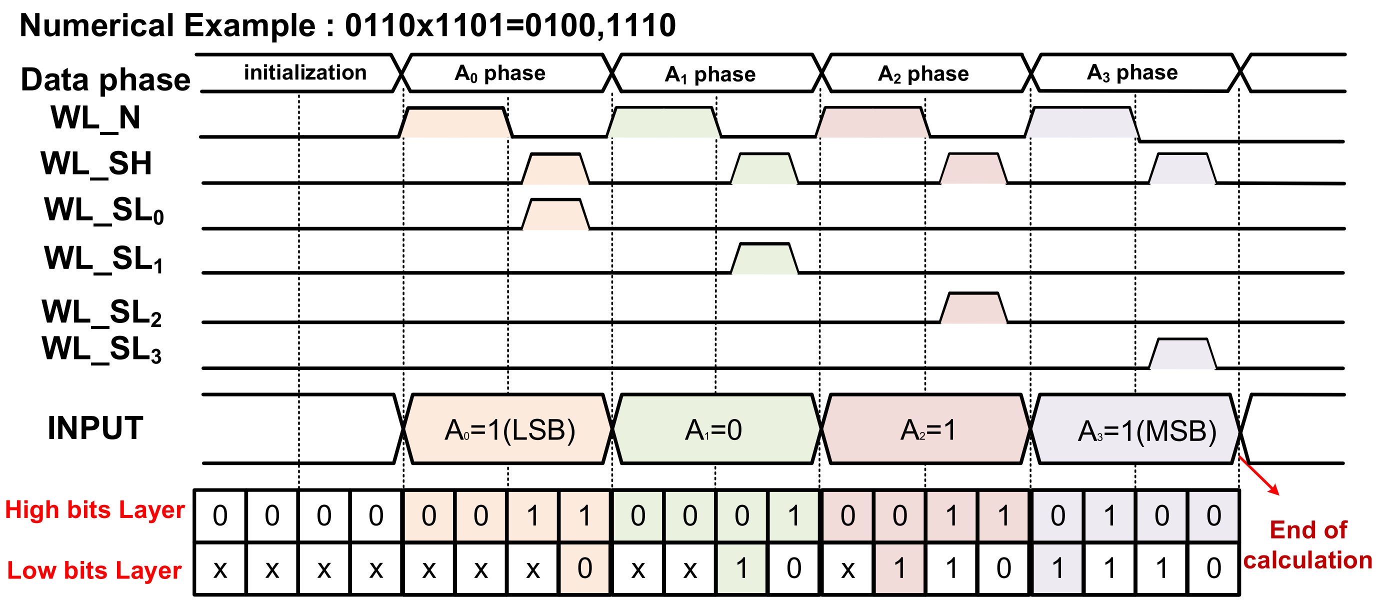


Fig. 8. Timing diagram of multiplication of internal weight 0110 and 4-bit input 1101 in IMCU. The same color represents the operation in the same cycle; different colors show the changes of the corresponding control signals and values in the memory cell in different cycles.

IV. Experimental Results

The proposed SRAM memory was simulated and fabricated using the SMIC 55-nm CMOS technology. The memory size was 64 × 64, i.e., a 4-kb array. The size of the array will not affect the function of the circuit, nor will it affect the performance of the circuit. Considering that 64 × 64 can meet the common filter sizes, thus, our prototype chip has chosen this array size. The parasitic parameters are extracted from the layout, and the postlayout simulation and chip testing are carried out.

A. Simulation Results

We simulated the read static noise margin (RSNM) and hold static noise margin (HSNM) of these three cell structures at different temperatures and process corners. Since the SNMs of the three structures are basically the same, only the RSNM and HSNM of the 8T cell are shown here. The 5k trails Monte Carlo simulations were performed using ADE XL in Cadence at V_{DD} = 1.2 V, 25 °C, and typical nmos and typical pmos (TT) process corner. The local and global variations are considered. Figs. 11 and 12 show the RSNM and HSNM simulation results, respectively. Both the RSNM and HSNM gradually decrease with increasing temperature. The 5k trails Monte Carlo simulation results of write operations are shown in Fig. 13. The average flip voltage and standard deviation of the three cell structures are 445.7 and 1.3 mV, respectively. This is because the 8T and 9T cells in this circuit do not change the basic coupling structure, so there is no effect on the noise tolerance during read, hold, and write operations.

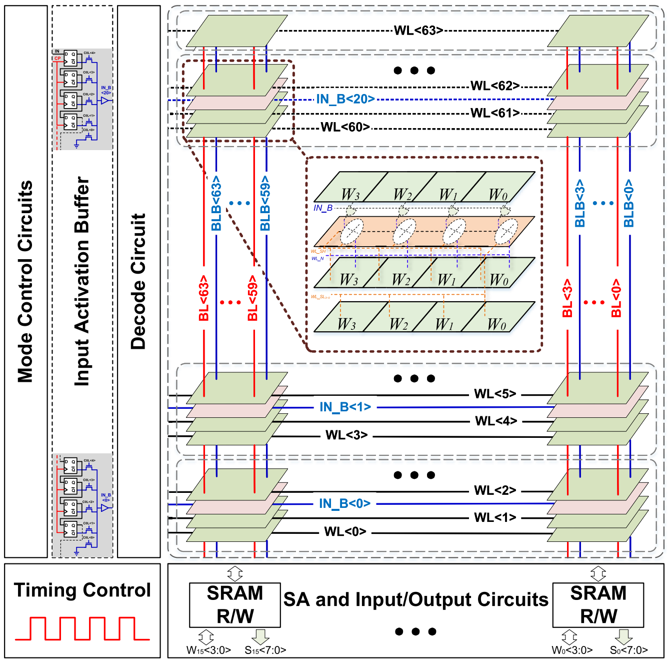


Fig. 9. Overall SRAM architecture and schematic representation of the reusable multiplication calculation macro.

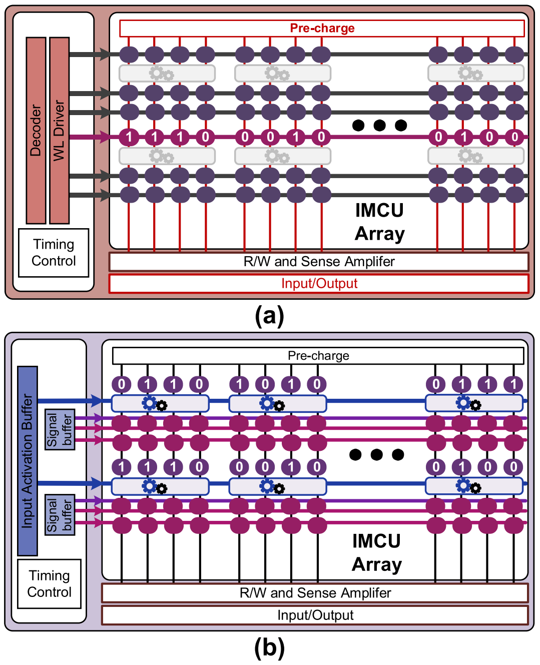


Fig. 10. Schematic of (a) SRAM mode and (b) IMC mode.

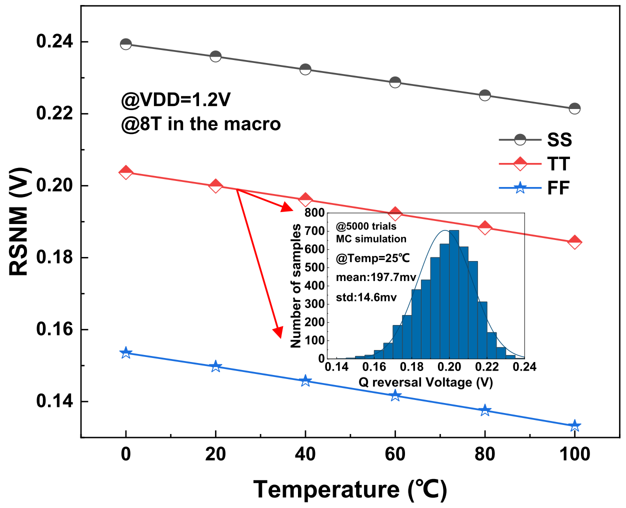


Fig. 11. RSNM of 8T with different temperatures and corners.

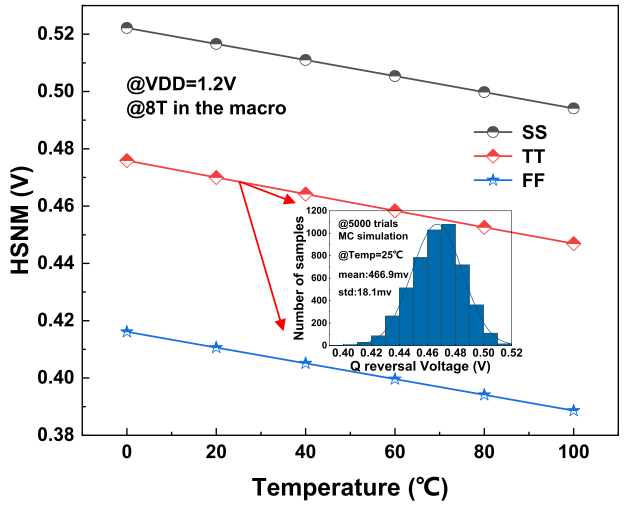


Fig. 12. HSNM of 8T with different temperatures and corners.

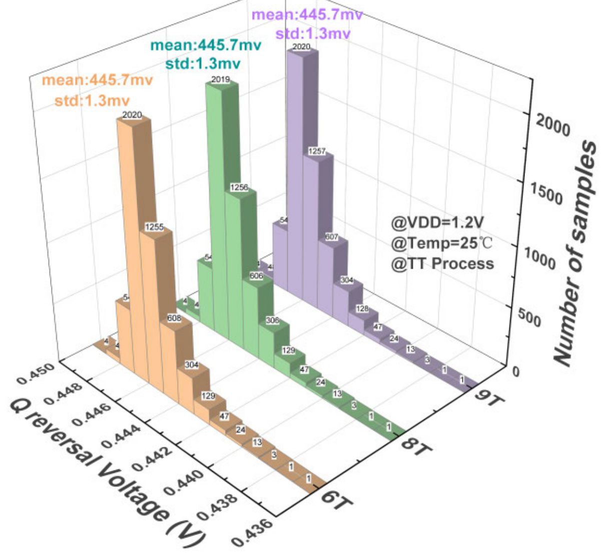


Fig. 13. 5k trails Monte Carlo simulation of 6T, 8T, and 9T write operation.

The write and read delays from 0 °C to 100 °C for different power supplies are shown in Fig. 14. In this study, the write delay is defined as the time from the rise of WL voltage to 50% V_{DD} until the storage node changes from low level to 90% V_{DD}, and the read delay of cell node voltage is defined as the time from the rise of WL voltage to 50% V_{DD} until the SA node changes from low level to 90% V_{DD}. Both the write delay and read delay in the figure decrease as the supply voltage increases. The write delay is about 490 ps and the read delay is about 965 ps at a supply voltage of 1.2 V.

B. Measured Results

Fig. 15 shows the chip test results for read/write and multiplication operation pass rates at different supply voltages

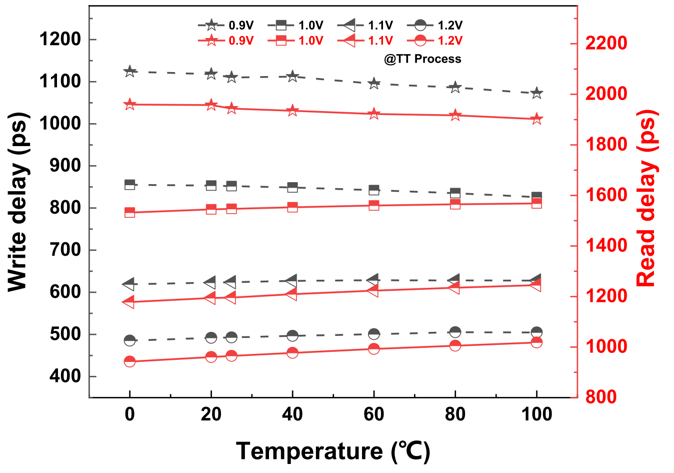


Fig. 14. Simulated write delay and read delay across temperature (0°–100°).

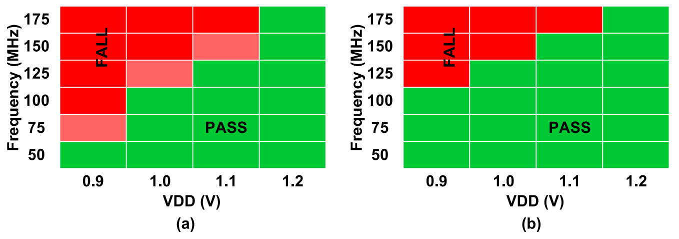


Fig. 15. Results of (a) read/write and (b) calculation operations at different supply voltages and frequencies.

and clock frequencies. Errors are marked in red, while green markers indicate correct results. Pink indicates that there is a frequency value in the interval. If it is higher than the frequency value, the calculation begins to have errors. The maximum frequency of the read/write operation [see Fig. 15(a)] and the multiplication operation [see Fig. 15(b)] increases as the supply voltage increases. When the supply voltage decreases or the frequency increases, the probability that the read/write or multiplication operation will fail increases. The read/write operation pass rate is lower when the supply voltage is below 1 V. This is because when the voltage drops below 1 V, the timing circuit cannot meet the timing constraints due to circuit delay, resulting in abnormal control signals and affecting the function of the circuit. If the timing circuit can be dynamically adjusted, a further reduction in the operating voltage can be expected. In addition, the whole calculation process is carried out in the digital domain, which ensures the accuracy of the calculation.

Fig. 16 shows the read/write and multiplication operations measured at 25 °C. The read/write and multiply operation frequency curves are similar. When the power supply voltage is 1.2 V, the maximum frequencies of 177.8 and 187.1 MHz are reached. Due to the influence of circuit parasitic parameters, the measured frequencies for the read/write and multiply operations are lower than the simulated results. If the system needs a faster operating frequency in the future, the array can be divided into several submodules. Then, in the layout, the timing circuit module can be placed in the center of the array to reduce parasitic parameters.

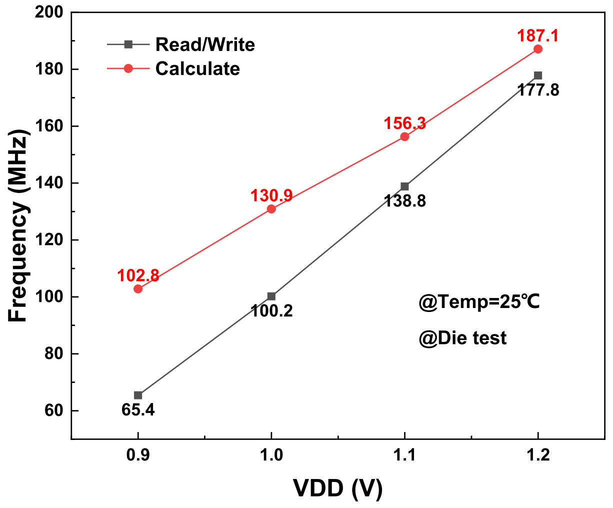


Fig. 16. Measured frequency of the read/write and calculation operations.

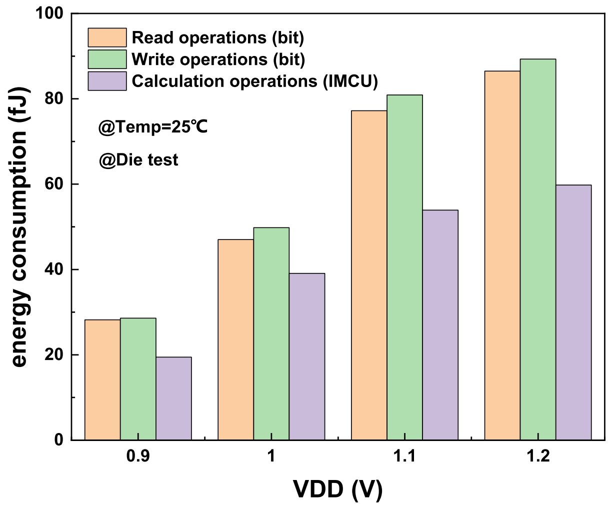


Fig. 17. Measured energy consumption results for read, write, and calculation operations.

Fig. 17 shows the measured energy consumption for read/write operations and multiplication operations. The measured energy consumption includes the entire peripheral device and IMCU array. The energy consumption increases with increasing supply voltage at 25 °C. Compared with read and write operations, the power consumption for performing multiplication operations is the lowest. This is because there are no precharge circuits and SAs to be involved in the operation. At $V_{DD} = 0.9$ V, the energy consumption of the read/write and multiplication operation is 28.17 fJ/bit, 28.6 fJ/bit, and 19.47 fJ/IMCU. At $V_{DD} = 1.2$ V, the energy consumption of the read/write and multiplication operation is 86.5 fJ/bit, 89.3 fJ/bit, and 59.8 fJ/IMCU.

Fig. 18(a) shows the overall layout of the designed 4-kb chip, including the array and peripheral circuits. Fig. 18(b) shows the layout and size of the 4-bit IMCU. The chip specifications are shown in Table I. The overall area is 0.067 mm^2, and the area of the array is 167.9 × 266.5 μm, which accounts for 66.55% of the total area of the chip. Fig. 19 shows the test

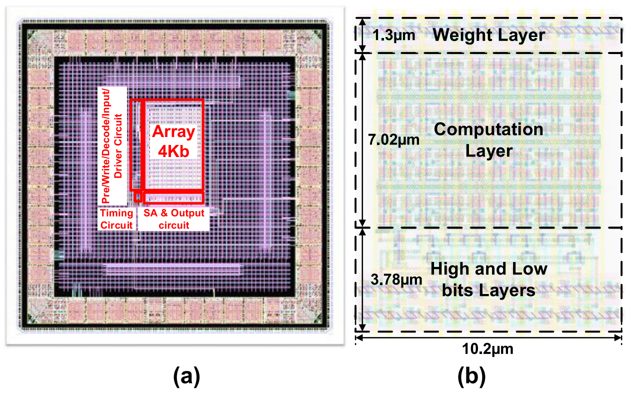


Fig. 18. (a) Chip architecture layout in a 55-nm CMOS process. (b) 4-bit IMCU layout size information.

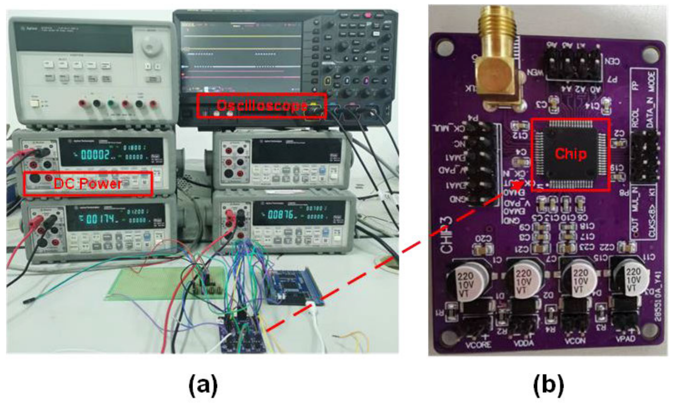


Fig. 19. (a) Test platform of the fabricated chip. (b) Chip on the PCB board.

TABLE I
CHIP SUMMARY

Chip summary	
Technology	Smic55ll_121825_1tm
Macro area	214.6μm x 313.3μm
Array area	167.9μm x 266.5μm
IMCU area	10.2μm x 12.1μm
Supply voltage	0.9-1.2 V
Capacity	4Kb

platform of the chip, including the SRAM IMC test chip and associated test tools.

Table II compares the proposed multiplication IMC scheme with previous work. Depending on the computational method, the IMC architecture can be divided into analog and digital solutions. In this comparison table, works [8], [10], and [14] use an analog calculation method, and the calculation results are affected by the ADC conversion accuracy. The methods in [26], [28], and [33] can perform high-precision arithmetic operations because they are combined with digital computing methods. However, compared with this work, they cannot provide higher energy efficiency. At the same time, this work can also provide good throughput per unit area. Compared with the conventional multiplication operation, this scheme reuses the logic unit to reduce the area overhead, and the calculation results are written back to the local unit, which is not available in other reference documents. In summary, this solution provides higher energy efficiency and good throughput per unit area compared with the existing IMC architecture using digital computing methods.

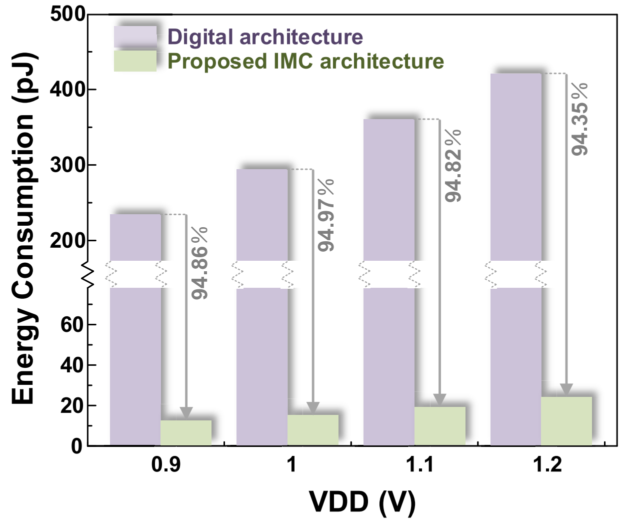


Fig. 20. Comparison of energy consumption between the conventional von Neumann architecture and the proposed IMC architecture.

V. APPLICATION AND DISCUSSION

In von Neumann's architecture, the energy required to access data from memory is much higher than the energy required for computing operations. To verify the effectiveness of the proposed circuit in terms of energy consumption, we compared it with the energy consumption of the conventional von Neumann digital strategy. For von Neumann's architecture, the main source of energy consumption is the repeated reading and writing of data. For the proposed IMC architecture, the main energy consumption is a four-cycle operation. As shown in Fig. 20, the energy efficiency of the circuit has increased by 94.35%–94.97%.

In CMOS circuits, there are two main power sources: dynamic power and static power. For the core array, the dynamic power consumption is mainly caused by read and write operations, and the static power consumption is mainly caused by the leakage current in the hold state. We simulated the proposed IMCU architecture and obtained dynamic power consumption (power consumption during data reading and writing) and static power consumption (power consumption during data retention). As shown in Fig. 21, the static power consumption is 0.69% of the read power consumption at a 1.2-V power supply voltage.

To better apply the IMCU to a neural network, we propose a multiplication–accumulation circuit architecture, as shown in Fig. 22. It consists of two parts: an IMCU-based multiplication module and a summation module based on the addition tree group. The weight layer of IMCU is used to store the convolution (CONV) core, and the multiplication results stored in the high and low bits layers are sent to the summation

TABLE II
COMPARISON OF THE PROPOSED STUDY WITH PREVIOUS IN-SRAM COMPUTING STUDIES

	This work	JSSC2019[8]	JSSC2021[10]	JSSC2022[14]	JSSC2020[26]	JSSC2020[28]	JSSC2021[33]
Technology	55nm	65nm	28nm	65nm	28nm	65nm	65nm
Supply Volyage	0.9-1.2V	0.8-1.2V	0.7-0.9V	0.9-1.2V	0.6-1.1V	0.85-1.2V	0.6-0.8V
Array size	4Kb	16Kb	64Kb	75Kb	128Kb	576Kb	16Kb
Bitcell area(μm^2)	3.01	-	4.1	4.56	0.78	-	10.53
Die area(mm^2)	0.067(macro)	0.063(array)	0.3234(macro)	1.04(macro)	1.2(array)	8.56	0.271(macro)
Input Precision	4 b	6 b	4,8 b	1 b	8 b	1-8 b	1-16 b
Weight Precision	4 b	1 b	4,8 b	1 b	8 b	1-8 b	1-16 b
Method of Computing	Digital In-Memory	Analog In-Memory	Analog In-Memory	Analog In-Memory	Digital In/Near-Memory	Analog+Digital In-Memory	Digital In-Memory
Frequency(MHz)	65.4-187.1	5	244	48	475	40-100	75.8-138
Energy Efficiency(TOPS/W)	51.4**	40.3-51.3	47.85-68.44(4b)	24.3*	21.1*	12-25*	22*
Performance per unit Area(GOPS/mm^2)	234.3 @55nm	63.5-127 @65nm	386.2(4b) @28nm	24.7* @65nm	109.2* @28nm	6.4*-16* @65nm	357.9* @65nm

*The input and weight are normalized to 4 b.
**Based on test data 0110 x 1101.
(Where the energy efficiency is average with nearly half of data flipping.)

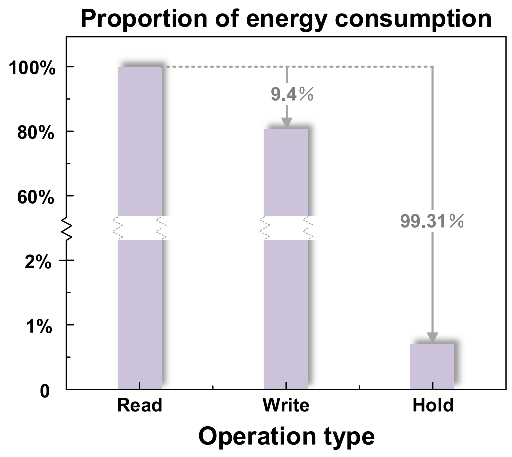


Fig. 21. Power consumption comparison of read, write, and hold operations.

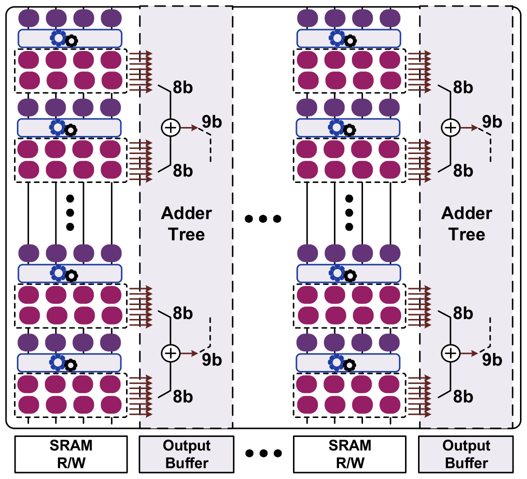


Fig. 22. Multiplication–accumulation circuit architecture based on IMCU.

module for accumulation. To enhance the parallelism of operation, the CONV kernel is stored in columns, the external image data are input in rows, and the multiperiod dot-product operation is performed by bit and weight layers. When the final multiplication result is stored in the high and low bits layers, the summation module accumulates the multiplication results of the same column and finally obtains the CONV result.

CNNs are usually composed of concatenated CONV layer and a fully connected (FC) layer. Fig. 23 shows the basic configuration diagram of the CONV/FC layer of CNN, where the size of the 3-D filter is $R \times R \times C$. R represents the width/height and C represents the number of input channels. The size of the input image is $H \times H \times C$, where H represents the width/height of the image. A 3-D filter is applied to a 3-D input characteristic map (IFMP) to obtain the size of a 3-D output characteristic map (OFMP), i.e., $E \times E \times C$, where E represents the width/height of the 3-D output characteristic map (OFMP).

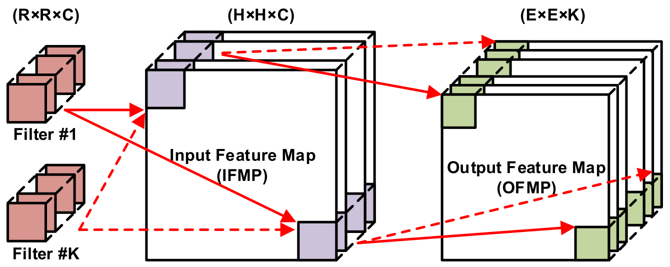


Fig. 23. Basic configuration diagram of CNN's CONV/FC layer.

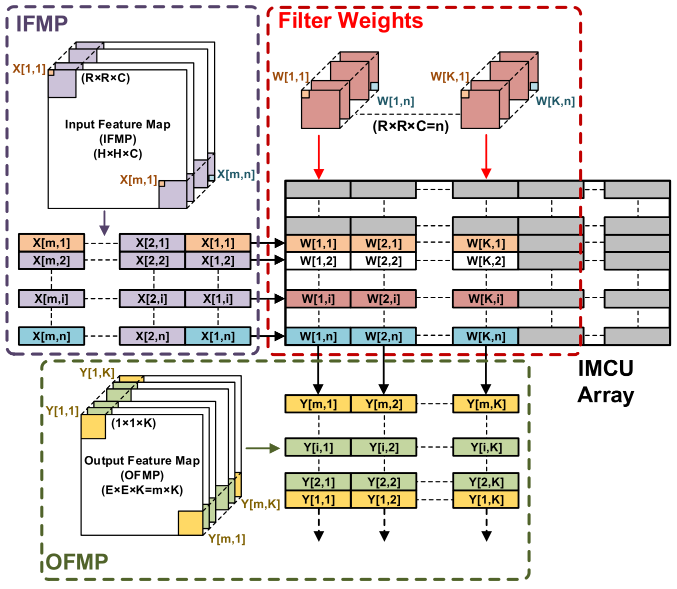


Fig. 24. Filter weight, input, and output characteristics are mapped to the multiplication–accumulation array of IMCU for MAC operation of a single CONV/FC layer.

Fig. 24 shows the CNN mapping strategy for the proposed 4-bit MAC array. For a single CONV/FC operation, the 3-D block located in the top-left corner of the IFMP is expanded into a set of vectors X [1, 1] – X [1, n], where the number of 4-bits represented by X will be input from the LSB successively through four cycles, and the number of vector inputs m is determined by the size and step size of the IFMP, considering the cycle of prestore the data "0"; hence, m 3-D blocks need 5 $\times$ m cycles. Filters are stored in the weight layer of IMCU by columns. The total K columns represent the number of filters, and the effective number of rows is n, representing the number of 4-bit weights of each filter. Thus, the $K \times n$ accumulation results can be obtained through four cycles. Vector Y [1, 1] – Y [m, 1] is obtained by m times calculating. Running Mixed National Institute of Standards and Technology database (MNIST) dataset on this architecture can achieve 98.7% accuracy.

VI. Conclusion

IMC is an effective method to overcome the von Neumann bottleneck. In this study, we proposed an SRAM IMC scheme using digital methods to perform high-precision operations. The proposed circuit is a fully digital four-layer IMCU. External data are sent to the computing layer through a multicycle serial input, which simplifies the multiplication operation of multibits into an addition operation, and the circuit that performs the addition operation is reused to reduce area overhead. At the same time, the final multiplication result is stored in the IMCU so that the calculation result does not need to be read immediately. In addition, this method is easily extensible to operations with different bit widths. The manufactured SRAM IMC macro can achieve a throughput of 234.3 GOPS/mm^2. At a 0.9-V power supply voltage, an energy efficiency of 51.4 TOPS/W is achieved. At the same time, in order to better apply IMCU to CNN, a multiplication–accumulation circuit architecture is proposed. Applying the MNIST dataset to this architecture can achieve 98.7% accuracy.

References

[1] J. Zhang, Z. Wang, and N. Verma, "A machine-learning classifier implemented in a standard 6T SRAM array," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2016, pp. 1–2, doi: 10.1109/VLSIC.2016.7573556.

[2] M. Kang, S. K. Gonugondla, M.-S. Keel, and N. R. Shanbhag, "An energy-efficient memory-based high-throughput VLSI architecture for convolutional networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2015, pp. 1037–1041.

[3] 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 *Proc. IEEE Symp. VLSI Circuits*, Honolulu, HI, USA, Jun. 2018, pp. 141–142.

[4] M. Kang, E. P. Kim, M.-S. Keel, and N. R. Shanbhag, "Energy-efficient and high throughput sparse distributed memory architecture," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2015, pp. 2505–2508.

[5] M. Kang and N. R. Shanbhag, "In-memory computing architectures for sparse distributed memory," *IEEE Trans. Biomed. Circuits Syst.*, vol. 10, no. 4, pp. 855–863, Aug. 2016.

[6] E. Lee et al., "A charge-domain scalable-weight in-memory computing macro with dual-SRAM architecture for precision-scalable DNN accelerators," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 8, pp. 3305–3316, Aug. 2021, doi: 10.1109/TCSI.2021.3080042.

[7] M. E. Sinangil et al., "A 7-nm compute-in-memory SRAM macro supporting multi-bit input, weight and output and achieving 351 TOPS/W and 372.4 GOPS," *IEEE J. Solid-State Circuits*, vol. 56, no. 1, pp. 188–198, Jan. 2021, doi: 10.1109/JSSC.2020.3031290.

[8] 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 J. Solid-State Circuits*, vol. 54, no. 1, pp. 217–230, Jan. 2019.

[9] S. Jain, L. Lin, and M. Alioto, "±CIM SRAM for signed in-memory broad-purpose computing from DSP to neural processing," *IEEE J. Solid-State Circuits*, vol. 56, no. 10, pp. 2981–2992, Oct. 2021, doi: 10.1109/JSSC.2021.3092759.

[10] 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 J. Solid-State Circuits*, vol. 56, no. 9, pp. 2817–2831, Sep. 2021, doi: 10.1109/JSSC.2021.3073254.

[11] J.-W. Su et al., "A 28 nm 64 Kb inference-training two-way transpose multibit 6T SRAM compute-in-memory macro for AI edge chips," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 240–242.

[12] 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 Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 8, pp. 2521–2531, Aug. 2020.

[13] R. Khaddam-Aljameh et al., "An SRAM-based multibit in-memory matrix-vector multiplier with a precision that scales linearly in area, time, and power," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 2, pp. 372–385, Feb. 2021.

[14] S. K. Bose et al., "A 389 TOPS/W, always ON region proposal integrated circuit using in-memory computing in 65 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 58, no. 2, pp. 554–568, Feb. 2023, doi: 10.1109/JSSC.2022.3194098.

[15] Y.-C. Chiu et al., "A 4-Kb 1-to-8-bit configurable 6T SRAM-based computation-in-memory unit-macro for CNN-based AI edge processors," *IEEE J. Solid-State Circuits*, vol. 55, no. 10, pp. 2790–2801, Oct. 2020, doi: 10.1109/Jssc.2020.3005754.

[16] S. Huang, H. Jiang, X. Peng, W. Li, and S. Yu, "XOR-CIM: Compute-in-memory SRAM architecture with embedded XOR encryption," in *Proc. 39th Int. Conf. Comput.-Aided Design*, Nov. 2020, pp. 1–6.

[17] A. K. Rajput and M. Pattanaik, "Implementation of Boolean and arithmetic functions with 8T SRAM cell for in-memory computation," in *Proc. Int. Conf. Emerg. Technol. (INCET)*, Jun. 2020, pp. 1–5.

[18] J. Chen, W. Zhao, and Y. Ha, "Area-efficient distributed arithmetic optimization via heuristic decomposition and in-memroy computing," in *Proc. IEEE 13th Int. Conf. ASIC (ASICON)*, Chongqing, China, Oct. 2019, pp. 1–4.

[19] H. Kim, Q. Chen, and B. Kim, "A 16K SRAM-based mixed-signal in-memory computing macro featuring voltage-mode accumulator and row-by-row ADC," in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, Macao, China, Nov. 2019, pp. 35–36.

[20] A. Agrawal et al., "X-SRAM: Enabling in-memory Boolean computations in CMOS static random access memories," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 12, pp. 4219–4232, Dec. 2018, doi: 10.1109/TCSI.2018.2848999.

[21] W. A. Simon, Y. M. Qureshi, M. Rios, A. Levisse, M. Zapater, and D. Atienza, "BLADE: An in-cache computing architecture for edge devices," *IEEE Trans. Comput.*, vol. 69, no. 9, pp. 1349–1363, Sep. 2020, doi: 10.1109/tcsi.2018.2848999.

[22] W. Simon, J. Galicia, A. Levisse, M. Zapater, and D. Atienza, "A fast, reliable and wide-voltage-range in-memory computing architecture," in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, p. 83.

[23] N. Surana, M. Lavania, A. Barma, and J. Mekie, "Robust and high-performance 12-T interlocked SRAM for in-memory computing," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Grenoble, France, Mar. 2020, pp. 1323–1326.

[24] Z. Lin et al., "In-memory computing with double word lines and three read ports for four operands," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 5, pp. 1316–1320, May 2020, doi: 10.1109/TVLSI.2020.2976099.

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

[26] J. Wang et al., "A 28-nm compute SRAM with bit-serial logic/arithmetic operations for programmable in-memory vector computing," *IEEE J. Solid-State Circuits*, vol. 55, no. 1, pp. 76–86, Jan. 2020, doi: 10.1109/JSSC.2019.2939682.

[27] B. Moons and M. Verhelst, "An energy-efficient precision-scalable ConvNet processor in 40-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 903–914, Apr. 2017.

[28] H. Jia, H. Valavi, Y. Tang, J. Zhang, and N. Verma, "A programmable heterogeneous microprocessor based on bit-scalable in-memory computing," *IEEE J. Solid-State Circuits*, vol. 55, no. 9, pp. 2609–2621, Sep. 2020.

[29] Y. Zhang et al., "Time-domain computing in memory using spintronics for energy-efficient convolutional neural network," *IEEE Trans. Circuit Syst. I, Reg. Papers*, vol. 68, no. 3, pp. 1193–1205, Mar. 2021.

[30] Z. Wang et al., "A 148 nW general-purpose event-driven intelligent wake-up chip for AIoT devices using asynchronous spike-based feature extractor and convolutional neural network," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2021, pp. 436–438.

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

[32] L. Deng et al., "Tianjic: A unified and scalable chip bridging spike-based and continuous neural computation," *IEEE J. Solid-State Circuits*, vol. 55, no. 8, pp. 2228–2246, Aug. 2020.

[33] H. Kim, T. Yoo, T. T.-H. Kim, and B. Kim, "Colonnade: A reconfigurable SRAM-based digital bit-serial compute-in-memory macro for processing neural networks," *IEEE J. Solid-State Circuits*, vol. 56, no. 7, pp. 2221–2233, Jul. 2021, doi: 10.1109/JSSC.2021.3061508.

[34] Y.-D. Chih et al., "An 89 TOPS/W and 16.3 TOPS/mm^2 all-digital SRAM-based full-precision compute-in memory macro in 22 nm for machine-learning edge applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2021, pp. 252–254.

[35] S. K. Gonugondla, M. Kang, and N. Shanbhag, "A 42 pJ/decision 3.12 TOPS/W robust in-memory machine learning classifier with on-chip training," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 490–492.

[36] H. Jiang, X. Peng, S. Huang, and S. Yu, "CIMAT: A transpose SRAM-based compute-in-memory architecture for deep neural network on-chip training," in *Proc. Int. Symp. Memory Syst.* Portland, OR, USA: Columbia, 2019, pp. 490–496.

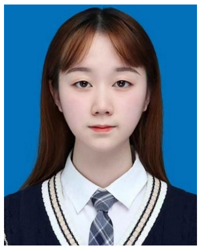
[37] S. Anand and S. Indu, "A low power and high speed 8-bit ALU design using 17T full adder," in *Proc. 7th Int. Conf. Signal Process. Integr. Netw.*, Apr. 2020, pp. 514–519, doi: 10.1109/SPIN48934.2020.9070844.

Zhiting Lin (Senior Member, IEEE) received the B.S. and Ph.D. degrees in electronics and information engineering from the University of Science and Technology of China (USTC), Hefei, Anhui, China, in 2004 and 2009, respectively.

From 2015 to 2016, he was a Visiting Scholar with the Department of Engineering and Computer Science, Baylor University, Waco, TX, USA. In 2011, he joined the School of Integrated Circuits, Anhui University, Hefei, where he is currently a Professor, and also with Anhui Provincial High-Performance Integrated Circuit Engineering Research Center, Hefei. He has published about 50 articles and holds over 20 Chinese patents. His research interests include pipeline analog-to-digital converters and high-performance static random access memory.

Shaoying Zhang received the B.S. degree in electrical engineering and automation from Zhejiang University of Water Resources and Electric Power, Hangzhou, China, in 2020. He is currently working toward the M.S. degree in integrated circuit engineering at Anhui University, Hefei, China.

His research interests include static random access memory and in-memory computing circuits.



Qian Jin received the B.S. degree from Xi'an University of Technology, Xi'an, China, in 2021. She is currently working toward the M.S. degree in integrated circuit engineering at Anhui University, Hefei, China.

Her current research interests include high-performance static random access memory and in-memory computing based on static random access memory (SRAM).

Jianping Xia received the B.S. degree in electronic information engineering from Huaiyin Normal University, Huai'an, China, in 2019. She is currently working toward the M.S. degree in integrated circuit engineering at Anhui University, Hefei, China.

Her current research interests include static random access memory and in-memory computing circuits.

Yunwei Liu received the B.S. degree in electronic science and technology from Anhui University, Hefei, China, in 2020, where he is currently working toward the M.S. degree in circuit and system.

His current research interests include high-performance static random access memory and in-memory computing.

Kefeng Yu received the B.S. degree in electronic science and technology from Anhui Science and Technology University, Bengbu, China, in 2020. He is currently working toward the M.S. degree in circuit and system at Anhui University, Hefei, China.

His current research interests include static random access memory and in-memory computing.

Jian Zheng received the B.S. degree in electronic information engineering from Huaibei Normal University, Huaibei, China, in 2020. He is currently working toward the M.S. degree in integrated circuit engineering at Anhui University, Hefei, China.

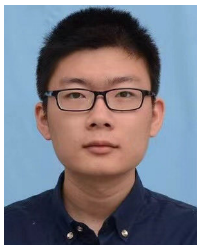
His research interests include static random access memory and in-memory computing circuits.

Xiaoming Xu received the B.S. degree in electronic science and technology from Anhui University, Hefei, China, in 2020, where he is currently working toward the M.S. degree in circuit and system.

His current research interests include high-performance static random access memory and in-memory computing.

Xing Fan received the B.S. degree in microelectronics science and engineering from Anhui University, Hefei, China, in 2019, where he is currently working toward the M.S. degree in circuit and system.

His current research interests include high-performance static random access memory and in-memory computing.



Ke Li received the B.S. degree in electronic science and technology from Jiangxi University of Science and Technology, Ganzhou, China, in 2019. He is currently working toward the M.S. degree in circuits and systems at Anhui University, Hefei, China.

His current research interests include static random access memory and nonvolatile static random access memory.

Zhongzhen Tong received the B.S. degree in electronic science and technology from Tianjin University of Technology and Education, Tianjin, China, in 2019, and the M.S. degree in circuits and systems from Anhui University, Hefei, China, in 2022. He is currently working toward the Ph.D. degree at the School of Integrated Circuit Science and Engineering, Beihang University, Beijing, China.

His current research interests include hybrid carbon nanotube field effect transistor (CNTFET)/spintronic integrated circuits design and in-memory computing based on static random access memory (SRAM).

Xiulong Wu received the B.S. degree in computer science from the University of Science and Technology of China (USTC), Hefei, China, in 2001, and the M.S. and Ph.D. degrees in electronic engineering from Anhui University, Hefei, in 2005 and 2008, respectively.

From 2013 to 2014, he was a Visiting Scholar with the Department of Engineering, The University of Texas at Dallas, Richardson, TX, USA. He is currently a Professor with Anhui University, and also with Anhui Provincial High-Performance Integrated Circuit Engineering Research Center, Hefei. He has published about 60 articles and holds over ten Chinese patents. His research interests include high-performance static random access memory and mixed-signal ICs.

Wenjuan Lu received the M.S. degree in circuits and systems and the Ph.D. degree in microelectronics and solid-state electronics from Anhui University, Hefei, China, in 2014 and 2017, respectively.

She is currently a Lecturer with Anhui University. Her current research interests include next-generation nonvolatile memory, and the circuit design of low-power, high-performance, and high-reliability nonvolatile memory devices.

Chunyu Peng (Member, IEEE) received the B.S. degree in communications engineering and the M.S. degree in circuits and systems from Anhui University, Hefei, China, in 2010 and 2013, respectively, where he is currently working toward the Ph.D. degree in microelectronics and solid-state electronics.

He is also an Associate Professor of Microelectronics and Solid-State Electronics with Anhui University. His research interests include signal processing, analog IC design, and high-performance reliable memory technology.

Qiang Zhao (Member, IEEE) received the M.S. and Ph.D. degrees in microelectronics and solid-state electronics from Anhui University, Hefei, China, in 2013 and 2020, respectively.

He is currently a Lecturer of Microelectronics and Solid-State Electronics with Anhui University. His current research interests include radiation hardening in VLSI designs and high performance and reliability in memories.