Proceedings of the 3rd International Conference on Software Engineering and Machine Learning
DOI: 10.54254/2755-2721/2025.29485

A Review of Optimizing SRAM-Based FPGA In-memory Computing

Wangchen Xu[1*], Gaoqi Dong[2], Hongbo Wen[3]

[1]*University of British Columbia, Vancouver, Canada*
[2]*The University of Manchester, Manchester, United Kingdom*
[3]*Zhixin High School, Guangzhou, China*
**Corresponding Author. Email:xuwangchen2021@163.com*

Abstract. When the demand for real-time data processing and energy keeps efficiency growing in fields like Advanced Driver-Assistance Systems (ADAS) for electric vehicles, in-memory computing (IMC) is becoming a key technology. The heart of effective IMC is Static Random Access Memory (SRAM). It is widely known for its fast access times and low power requirements. For these reasons, SRAM becomes an ideal choice for FPGA-based systems. This paper delves into optimizing SRAM for IMC by comparing the performance, power efficiency, and stability of three SRAM types: 6T, 8T, and 10 T. What's more, we introduce the innovative C3SRAM architecture. This technology leverages capacitive coupling to boost computational speed and energy efficiency significantly. Finally, we summarize the CONV-SRAM architecture, tailored for in-memory convolution operations in neural networks. Through these explorations, we provide practical insights into how SRAM can be optimized to meet the demands of high-performance, energy-efficient systems, focusing on applications like autonomous vehicles that require speed and power conservation.

Keywords: Static Random Access Memory (SRAM), Field-Programmable Gate Arrays (FPGAs), Advanced Driver Assistance Systems (ADAS)

1. Introduction

As the automotive industry continues to evolve, more and more technology is being invested in vehicle safety. These technologies, known as Advanced Driver Assistance Systems (ADAS), are controlled by complex real-time embedded systems [1]. Real-time data from sensors, such as cameras and radars, are processed to make quick and accurate driving decisions. To meet the technical requirements of low power consumption and high-speed data processing, in-memory computing (IMC) has emerged as a promising solution. By performing calculations directly within memory, IMC shifts away from traditional computing models, significantly reducing power usage and data transmission delays.

Field-Programmable Gate Arrays (FPGAs) are gaining popularity for implementing in-memory computing (IMC) due to their parallel processing capabilities and flexibility. A key component of this architecture is Static Random Access Memory (SRAM). Through in-memory computing,

DOI: 10.54254/2755-2721/2025.29485

SRAM banks can be repurposed as compute engines while performing bulk Boolean operations. Thus, SRAM plays a critical role in the performance and efficiency of IMC systems, and it is essential to optimize SRAM for FPGA-based IMC to meet the high-performance and low-power requirements of complex applications.

In this paper, we explore different strategies for optimizing FPGA-based in-memory computing by focusing on three key aspects of SRAM. First, we analyze the conventional 6T, 8T, and 10T SRAM architectures, examining their performance in terms of power efficiency, application suitability, and overall performance. After laying this groundwork, we introduce the C3SRAM design, a macro that leverages capacitive coupling to boost computational speed and energy efficiency, making it ideal for memory-heavy applications. We also examine the CONV-SRAM design, which is optimized for in-memory convolution operations and provides significant gains in processing speed and energy efficiency, particularly in deep neural network workloads.

This paper provides insights into optimizing SRAM techniques for FPGA-based in-memory computing by synthesizing different SRAM technologies. The findings highlight important design directions for creating high-performance, energy-efficient computing systems, particularly for applications like ADAS, where real-time processing and low power consumption are essential. This research not only pushes forward the technology used in electric vehicles but also lays the groundwork for further exploration of SRAM optimization across a variety of FPGA applications.

2. Background

2.1. In-memory computing

In-memory computing (IMC) refers to a paradigm shift from traditional computing architectures, which aim to mitigate the limitations of the von Neumann bottleneck [2]. The separation of memory and processing units causes this bottleneck, which results in large data transmission overheads and energy inefficiencies. In a typical design, data must go back and forth between the CPU and the memory. This procedure adds to the latency and power consumption. To overcome this difficulty, IMC conducts computations directly inside the memory units. This minimizes the need to transfer data and greatly improves speed and energy efficiency.

Modern applications like Advanced Driver-Assistance Systems (ADAS) in electric vehicles need real-time data processing and minimal latency, which have made IMC particularly important. IMC is an appealing choice because these systems require enormous amounts of sensor data to be processed quickly and efficiently to make driving actions instantly. IMC lowers power consumption is important for battery-operated systems like electric vehicles, and speeds up processing by incorporating computational capabilities into memory.

2.2. Static Random Access Memory (SRAM)

Static Random Access Memory (SRAM) is a kind of semiconductor memory in which every bit is stored using bistable latching circuitry. SRAM offers faster access times and more dependability than Dynamic RAM (DRAM), which has to have its data refreshed regularly. Instead, SRAM stores data for as long as power is applied. These qualities make SRAM the perfect option for developing IMC designs and serving as processors' high-speed cache memory.

DOI: 10.54254/2755-2721/2025.29485

3. The performance of SRAM cells topologies analysis

To gain a satisfactory ADAS, properly considering what topologies of SRAM cells are the most suitable is a serious and crucial question. In this part, we will compare some common SRAM-cell topologies' performance and analyze their advantage and disadvantages to help researchers make decisions. In [3], the authors review the performance of various SRAM cell topologies from multiple aspects. The research team utilized Cadence Virtuoso IC6.1.5-64b with a 45nm generic process design kit for their experiments. They introduced several advanced technologies to enhance SRAM performance. However, this paper focuses specifically on the energy consumption, stability, delay, and area of 6T, 8T, and 10T SRAM cell topologies. The following Table 1 is the data from the original paper.

Table 1. Data

	6t	8t	10t
VDD	1	1	1
RSNM	210	245	390
WSNM	365	355	345
Read Power (μW)	6.18	4.65	6.18
Read current (μW)	32.76	33.11	32.56
Leakage Power (μW)	18.61	26.55	25.05
Leakage current (μA)	7.41	7.41	8.10
Write Power (nW)	178	138.2	312.5
Read access time (pS)	67.23	36.9	40.52
Write access time (pS)	75.79	62.62	189.1
Read energy (aJ)	13.56	89.23	127.23
Write energy (aJ)	22.82	8.41	59.09
Area(normalized w.r.t Conv.6T SRAM)	1	1.45	2.57

3.1. Power analysis

Power dissipation of SRAM includes two major components, one is the static power consumption that mostly comes from and another one is dynamic power consumption. Specifically, static power consumption is mostly rooted in leakage, especially gate leakage and drain leakage. These leakage currents still exist when the storage unit is in a static state (when no read or write operations are performed). The dynamic power consumption mainly comes from the charge and discharge of the load capacitor during the switching operation. Each time the state of the storage unit changes, the associated capacitors are charged or discharged, thus consuming energy. The analytical method used to calculate the power of a Conv. 6T and 10T SRAM cells are expressed through the following equations.

$$P_{\text{total}} = P_{\text{dynamic}} + P_{\text{static}} = C_{\text{load}} * V_{dd}^2 * f + I_{\text{leak}} * V_{dd} \quad (1)$$

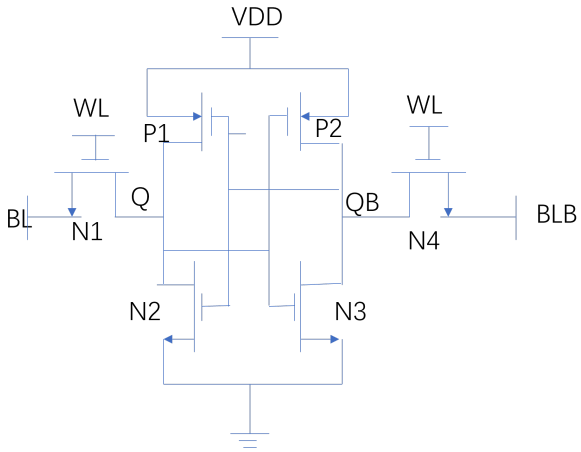
DOI: 10.54254/2755-2721/2025.29485

3.1.1. Read/write power

According to Table 1, we can find that the 10T and 6T SRAM have approximately the same read power but are much higher than the 8T SRAM. What's more, for the write power, the 10T SRAM is the highest, followed by 6T SRAM and 8T SRAM. These results are mostly because of the design aim of those SRAMs. First, the 6T (Figure 1) SRAM is the basic SRAM that has no other special design, but the 8T SRAM (Figure 2) uses two more transistors to separate the read and write path, specifically, the N4transistor create a new writing path and the N5N6 transistor form a reading path. In this way, less interruption by the noise, as well as less current, will be needed during one process, therefore, less power dissipation. To the 10T SRAM, four extra transistors often have other uses, which we will talk about later, so they consume more current.

3.1.2. Leakage

Leakage current in semiconductors and integrated circuits refers to the current that flows through a device even when no external driving signal (such as voltage or current) is applied. Leakage is always inevitable because of the manufacturing process and circuit design. Since 6T SRAM is the simplest SRAM with the least transistor, the leakage of 6T SRAM is the smallest. Additionally, 8T SRAM and 10T SRAM have approximately the same leakage power.



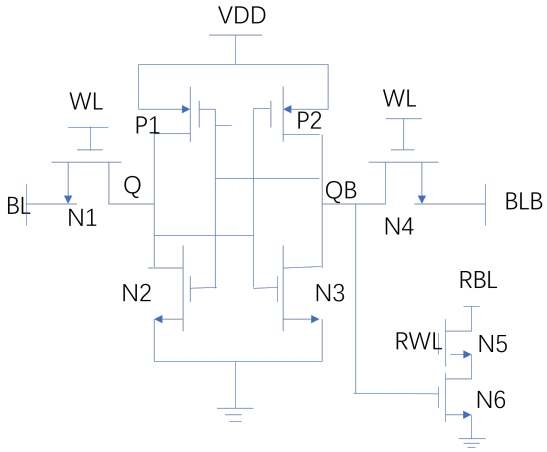


Figure 1. Conv.6T SRAMFigure 2: Conv.8T SRAM

3.2. Delay calculation

Delay is classified as two parts in the SRAM cell, which are read and write access time. The read access time is defined as the time from the time the read command is issued to the time the data stabilizes at the output, whereas the write access time is the time from the write command issued to the time the data is successfully written and kept stable. Specifically, read access time involves access time, read setup time, and read hold time as well as write access time involves write pulse width, write setup time, and write hold time.

3.2.1. Access time compare

From the origin paper, we can find that when the voltage is under 0.6V, both the read access time and write access time are significantly higher than 6T SRAM, especially the write access time.

DOI: 10.54254/2755-2721/2025.29485

However, when the voltage is bigger than 0.6V, those two times are nearly the same, especially the read access time. This is because of the characteristics of transistors at low voltages (such as reduced switching speed and increased leakage current), the resistance and capacitance effects of additional transistors in the design, and the effect of voltage on transistor performance. As the voltage becomes higher till the normal value, the delay time will decrease and have approximately the same read delay compared with 6T and 8T SRAM.

3.2.2. Read/write energy

The 6T SRAM has the highest read energy while the 10T SRAM has the highest write energy. The caution could be the design, 6T SRAM is the simplest, so its structure has no surprising update and optimized method, thus the current runs through all transistors, which consumes much power. For the 10T SRAM, the basic aim was to have a better performance in reading and stability, to the tradeoff, the performance in writing is not satisfactory

3.3. Stability

Static Noise Margin (SNM) is defined as the maximum amount of noise (voltage disturbance) that an SRAM cell can tolerate while still retaining its correct data state (either 0 or 1). It represents the cell's tolerance to noise and disturbances without flipping its state. SNM is typically measured in terms of voltage. It is the minimum voltage that, if applied to the SRAM cell, will not cause it to change its state. It could be divided into the read static noise margin(RSNM) and write static noise margin(WSNM)

10T SRAM has the highest RSNM value, which is the advantage that 10T SRAM is supposed to have. The extra four transistors can provide additional support during the reading process, reduce the interference of the read operation to the memory unit, and thus improve the reliability of the read data. These transistors are usually located on the periphery of the storage unit, and their role is to enhance the current path when reading, ensuring the stability of the data read.

At the expense of high RSNM value, the WSNM of the 10T SRAM is the lowest. This is because most 10T SRAM is focusing on improving the performance in the reading process, which is the decision that designers make trade-offs between performance, stability, and power consumption to achieve optimal SRAM performance.

3.4. SRAM cells summary

6T: The first advantage will be its price, since it is the simplest SRAM with the least transistor, low design, and manufacturing costs, it is much cheaper than other SRAMs. On the contrary, this factor also causes it to have low SNM, which means that it has low stability.

8T: Two extra transistors optimized the circuit, helping it to have low power dissipation and low delay

10T: With the largest number of transistors, it has the highest RSNM and performance. More transistors also mean a higher bandwidth and become more malleable, which means that 10T SRAM can support bandwidth high-performance computing as well as adapt to complex application requirements.

DOI: 10.54254/2755-2721/2025.29485

4. C3SRAM: an in-memory-computing SRAM Macro based on robust capacitive coupling computing mechanism

The growing number of electric vehicles has likely strengthened the focus on several key features in the automotive world, which especially impact Advanced Driver-Assistance Systems (ADAS). Those systems include sensors that collect and process data, which creates a need for in-memory computing (IMC) with intelligent processing skills in order to cope with the demands of the task. This study [4] presents C3SRAM as an innovative development in the in-memory computing (IMC) space. The design mainly uses the capacitive coupling approach, which enables the effective creation of binary neural networks (BNNs), a significant development for applications that require fast and efficient data processing.

The basic structure of the C3SRAM architecture is a 256x64 storage unit array. In addition to the SRAM circuit, the array contains an input activation decoder and a percolumn flash analog-to-digital converter (ADC). The design allows for parallel processing of vector-matrix multiplication, enabling it to handle 256 binary inputs and 256×64 binary weights simultaneously. As we know, big data-enabled processing is an extremely essential component to boost binary neural network (BNN) performance because it increases the accuracy of such networks and cut down longer time computations, which has many benefits in many aspects. To better understand this new technology, the next section of this part will provide some details of the bMAC operation, as well as the capacitive coupling.

4.1. bMAC in C3SRAM

The main idea behind the C3SRAM design is to do calculations directly within the memory by utilizing a cunning cap trick. (Table 2) breaks down how different signals and voltages are handled to make this happen. The table shows how the different combinations of input values and weights stored on the capacitors have an obvious influence on the charge and discharge characteristics and therefore the bMAC operation.

Table 2. Table illustrating MWL (Memory Word Line) and MWLB (Memory Word Line Bar) voltages for different input and weight configurations. Red highlights indicate significant operations

Input	MWL	MWLB	Weight	
			+1 (Q)	-1 (QB)
+1	VDR	0	VDR	0
-1	0	VDR	0	VDR
0	VRST	VRST	VRST	VRST
Reset	VRST	VRST	VRST	VRST

+1, -1, 0, Reset: These represent the different binary input values that are fed into the memory cells for computation.

MWL, MWLB: are used to control the memory cells and influence the capacitor charging based on input and weight combinations.

+1 (Q) and -1 (QB): Represent the stored weights in the SRAM cells, determining the polarity of the operations performed.

V_{DR}, 0, V_{RST}: The table 2 shows these voltage levels applied to the MWL and MWLB lines, determining whether a capacitor is charged or discharged.

DOI: 10.54254/2755-2721/2025.29485

For instance, when input is +1, in the case weight = +1(Q), MWL is set to V_{DR} and MWLB is set to 0. Result: The capacitor is charged to V_{DR} (highlighted in red), indicating a positive multiplication result. By performing calculations directly within the memory cells, the architecture reduces power consumption compared to traditional memory-processing architectures. In addition, this method allows for simultaneous execution of multiple operations, reducing data movement and improving speed, which is crucial for applications that require high-speed and energy-efficient data processing.

4.2. Capacitive coupling based in-memory computation of bMAC

Capacitive coupling involves the passage of electrical energy through the dielectric substance (the insulator) located between two conducting entities (capacitors). In capacitive coupling, energy can be transmitted through an electric field that is formed between conductors. Consequently, signals can flow without a physical connection between them, or they can transmit energy. This principle is widely applied in many circuits to help achieve space-saving energy transfer.

The capacitive coupling has some critical technical advantages that lead to the ability to work in parallel, high speed, and low energy dissipation. Instead of direct voltage, it works with electric fields and uses little energy but more accurate computations. The execution of operations within memory cells not only reduces the latency theoretical limit factor but also avoids the memory and processor data transfer. These instances, in particular, determine that electric coupling is suitable for high performance applications like neural networks.

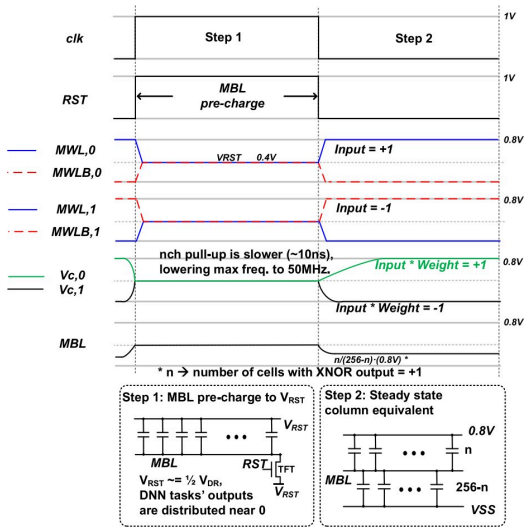


Figure 3. bMAC operation of C3SRAM [4]

Table 3. Table illustrating voltage changes

Ini	+1	-1	0
MWL	VDR	0	VRST
MWLB	0	VDR	VRST

The bMAC operation is depicted in (Figure 3) and involves two main steps, each completed within a half-cycle duration. Step1: Which is called Pre-Charging. In this step, the MBL of each column is pre-charged to $V_{RST} = 0.5 * V_{DR}$. This voltage is set near the expected bMAC output of 0, which helps minimize voltage swings during computation. Then comes the capacitor reset, which

DOI: 10.54254/2755-2721/2025.29485

means MWL and MWLB are both reset to V_{RST} , this is to ensure no initial voltage potential on the bitcell capacitors. This effectively arranges the capacitors in parallel, resetting them to a neutral state, as shown in Fig.4(left bottom).

Step2: Which can be called as Input Activation and Computation. Once the MBL is pre-charged, the input activations I_{ni} are applied to the MWL and MWLB in parallel. The voltage change can be found in (Table 3). The difference in voltages across MWL and MWLB induces a displacement current through the capacitor C_C, which performs the multiplication.

We can then calculate Displacement Current I_C and Charge Transfer Q_{Ci} by basic function. The final goal is to use these parameters to calculate the Shared MBL Voltage V_{MBL} by the following equation:

$$V_{MBL} = C_C V_{DR} \cdot \frac{\sum_{i=1}^{256} (XNOR_i)}{256C_C + C_p} \tag{2}$$

where:

• C_C is the coupling capacitance of each cell.

• V_{DR} is the reference voltage.

• $XNOR_i$ is the result of the XNOR operation for the i-th cell, representing the binary multiply result.

• C_p is the parasitic capacitance of the MBL plus the input capacitance of the ADC.

The shared MBL voltage has all cells inside the MBL contributing and worth the total inefficiencies of the bMAC operations in the column. This is the pivotal advantage of capacitive coupling Free Running Processors (FPR) as they naturally integrate or synthesize the outputs from multiple bMAC (Bitwise Multiply and Accumulate) operations, which enables quick and energy-efficient data processing pertinent to memory arrays.

5. CONV-SRAM: an energy-efficient SRAM With in-memory dot-product computation for low-power convolutional neural networks

5.1. Application scenario

In the context of autonomous driving, there are strict requirements for the speed and stability of data transmission. The content captured by image sensors needs to be processed in the cloud using complex convolutional neural networks (CNNs). However, the raw image data consumes a large amount of network bandwidth. Edge computing at the local level can greatly streamline this process by quickly processing data locally and sending only the essential information to the cloud, which is much smaller in size compared to the original data. This approach not only reduces bandwidth consumption but also enhances the speed and stability of the response.

Edge computing is crucial for autonomous driving and the Internet of Things (IoT) because it allows devices to make decisions locally or process detected data directly, and therefor only send essential information to the cloud, reducing the bandwidth consumed by transmission and the resulting latency.

5.2. Algorithm of computation

CONV-SRAM [5] is an embedded static random access memory (SRAM) architecture. Its features include optimization for dot product operations, utilization of in-memory computing, and integration of a binary-weight convolutional neural network (CNNs), to achieve a balances between low power

DOI: 10.54254/2755-2721/2025.29485

consumption and efficient processing. In-memory computing (IMC) breaks through the limitations of the von Neumann architecture to further enhance the efficiency of convolution operations.

Complete convolutional neural network operations require substantial computational power, while the processing capabilities of onboard computers are quite limited. Therefore, in addition to optimizing the hardware architecture, the algorithms of the model also need to be optimized.

$$Y_{x,y,k} = \sum_{c=1}^{C} \sum_{j=1}^{R} \sum_{i=1}^{R} W_{i,j,c,k} \times X_{x+i,y+j,c} \tag{3}$$

$$\{1 \le (x+i), (y+j) \le H1 \le x, y \le \lfloor \tfrac{H-R}{S} \rfloor + 1 (= E) 1 \le k \le M \tag{4}$$

The Equation 3 and 4 described one of the fundamental operations in CNN computations—extracting local features from input data and generating corresponding feature maps to interpret and understand the content of images. This is achieved by performing a dot product between the 3-D input feature map and the filter weights. By flattening the three-dimensional tensors into one-dimensional vectors, the dot product computation can be directly performed. One-dimensional vectors stored in memory are stored contiguously, thus allowing for faster processing compared to three-dimensional tensors. Introducing binary filter weights (i.e., using weights represented by values occupying only one bit) can convert the original floating-point operations into integer operations, thereby reducing computational complexity. Simultaneously, the storage space required for the weight values is significantly reduced, which also decreases bandwidth usage.

$$Y_{OUT,K} = \tfrac{1}{N} \sum_{i=1}^{R \times R \times C} \omega_{k,i} \times X_{IN,i} \tag{5}$$

$$V_{Y\_AVG,k} = \tfrac{1}{N} \sum_{i=1}^{R \times R \times C} \omega_{k,i} \times V_{a,i} \tag{6}$$

After these simplifications, the equation used in the digital domain (Equation 3 and 4) can be obtained. The data processed through digital-to-analog converters (DACs) can then be fed into the analog domain equation equivalent to Equation 5. In practical use, the input data is first converted into analog voltages readable by Equation 6 through DACs, followed by computing the multiply-and-average (MAV). Finally, the analog average voltage is converted back into the digital domain using analog-to-digital converters (ADCs) for further processing. After these simplifications, the hardware requirements for computation are significantly reduced, allowing complex CNNs to be processed quickly using limited local resources.

5.3. Circuit structure

The small transistor sizes used in traditional SRAM cells result in their precision being highly dependent on voltage stability. In contrast, the CONV-SRAM architecture employs global Digital-to-Analog Converters (DACs) to directly deliver analog voltages to the bitlines of the computation units, thereby reducing the potential for voltage variation.

Furthermore, traditional 6T SRAM cells can experience low voltage issues when multiple bit lines are activated simultaneously, leading to a phenomenon known as “pseudo-write.” This limits the voltage range. In comparison, the 10T cells used in CONV-SRAM avoid this problem by separating the read and write ports, allowing for a wider voltage range in analog computations without the risk of pseudo-writes. Despite the additional transistors, the well-designed 10T cells do not occupy more area than the 6T cells.

DOI: 10.54254/2755-2721/2025.29485

Previous techniques like AdaBoost could achieve precise computational results but come at the cost of increased energy consumption. Similarly, the on-chip training model proposed by Gonugondla et al. requires retraining the network for each chip, which consumes more time. In comparison, the CONV-SRAM architecture is more efficient.

As a result, the CONV-SRAM architecture could deal with tasks locally without compromising on power consumption and area efficiency.

The CONV-SRAM architecture was tested on the MNIST handwritten digit recognition task using the LeNet-5 CNN, and the results demonstrated accuracy close to that of digital-domain computations, significantly outperforming previously known in-memory computing methods. This indicates that the CONV-SRAM architecture is suitable for low-power machine learning applications, such as advanced driver-assistance systems (ADAS), 'always-ON' sensing, and Internet of Things (IoT) devices.

Optimization of the internal structure is important and crucial. In the CONVSRAM architecture, targeted solutions are provided for issues related to DAC signal conversion and the structural design of multiplication operations.

The philosophy of CONV-SRAM is to simplify the complexity of expressions, utilize analog voltage operations, and compress the circuit structure, thereby enhancing efficiency while ensuring that the circuit size remains unchanged.

Initially, converting digital signals to analog required a 64-to-1 multiplexer. However, after optimization, only an 8-to-1 multiplexer is needed, reducing the space required. Additionally, the optimized architecture balances the delays across signals of varying lengths. This process is divided into two phases: in the first phase, the three Most Significant Bits (MSBs) of X_{in} are used to select the pulse width required for precharging in the first half, while the three Least Significant Bits (LSBs) are used for the second half. Based on this, a control signal (TD_{56}) is used to switch between the two parts, allowing an 8-to-1 mux to share timing signals between the two phases.

The second phase involves multiplication and averaging of the signals. In the previous step, the digital voltage signal was converted into an analog voltage signal, which is then multiplied by a 1-bit filter weight and subsequently averaged. The calculation method is as follows: the read word line (RWL) is turned on to select a row in the memory array, discharging the local bitlines (LBLT or LBLF). Afterwards, the voltage difference is obtained, representing the product of the analog voltage and the stored weight. Upon completion of this step, the local bitlines are shorted together to calculate the average value, and depending on the sign of the input signal, the voltage is transmitted to different voltage rails (VpAVG or VnAVG). The control signals ENP and ENN are used to manage the switching of these voltage rails, while the PCHG control switch is used to handle a wide range of voltages. Test results show that when the SRAM bit-cell is used for weight evaluation, the maximum discharge time (about 500ps) is much shorter than the total clock cycle (100ns), meaning that the voltage variation on the bit-cell will not affect the actual computation time.

DOI: 10.54254/2755-2721/2025.29485

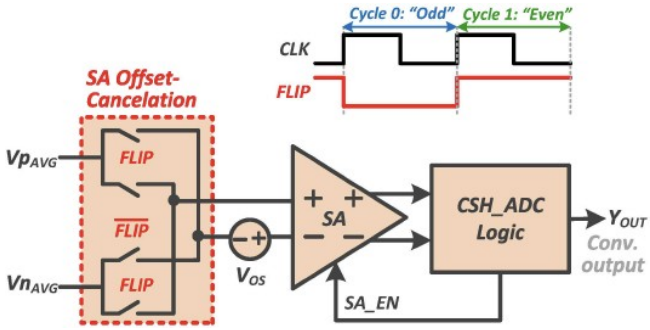


Figure 4. Circuit for the two-cycle OC technique

The third phase is converting the computed analog signal back into a digital signal.

This is done by simultaneously outputting the corresponding results of 16 filters through 16 parallel local arrays. The ADC used in the CONV-SRAM architecture employs a serial architecture, which consumes less power than a flash ADC, occupies a smaller area, and can achieve results close to flash ADC efficiency within a few clock cycles. Moreover, to mitigate the impact of bias voltage on computational accuracy, CONVSRAM also employs a two-circle offset-cancellation (OC) method. Figure 4 illustrates the OC structure. During the first and even-numbered cycles, FLIP = ”0”, the positive and negative average voltages are input to the positive and negative terminals, and the output voltage $Y_{out,0} = ADC(V_{yAV\ G,0} - V_{OS})$ is obtained, which is equivalent to the case without flipping. In odd-numbered cycles, FLIP = ”1”, and thus the calculation result is flipped. The results of each odd cycle and its corresponding even cycle are summed to represent the local convolution operation result. This method does not consume additional time or power, making it suitable for most CNNs.

6. Conclusion

In conclusion, this paper has explored the optimization strategies for SRAM-based inmemory computing within FPGA systems. This is especially relevant when considering demanding applications like Advanced Driver-Assistance Systems (ADAS) in electric cars. By analyzing our data, we determined important trade-offs in power economy, performance, and stability that affect the applicability of several SRAM typologies—6T, 8T, and 10T—for in-memory computing activities.

With the advent of cutting-edge designs like C3SRAM and CONV-SRAM, computational performance and energy efficiency have significantly increased thanks to novel techniques like capacitive coupling and in-memory dot-product calculation. These developments make processing quicker and more effective, which is necessary for real-time applications.

The study’s conclusions not only show how optimized SRAM technologies may improve the performance of FPGA-based systems, but they also open up new avenues for further investigation into how to make these strategies even better. With the increasing need for high-performance, low-power computing, especially in the automotive and Internet of Things domains, the information presented here will play a crucial role in steering the creation of more effective and efficient in-memory computing systems.

DOI: 10.54254/2755-2721/2025.29485

References

[1] Sangve, S. M. , Kulkarni, M. , Chawade, K. , Marathe, G. , & Pawar, P. . Advanced Driver Assistance System(ADAS). 2024 International Conference on Emerging Smart Computing and Informatics (ESCI). IEEE.
[2] Bansla, N. , & Rajneesh. (2021). Future erp: in-memory computing (imc) technology infusion.
[3] Harekrishna Kumar and VK Tomar. “A review on performance evaluation of different low power SRAM cells in nano-scale era”. In: Wireless Personal Communications 117.3 (2021), pp. 1959–1984.
[4] Zhewei Jiang et al. “C3SRAM: An in-memory-computing SRAM macro based on robust capacitive coupling computing mechanism”. In: IEEE Journal of Solid-State Circuits 55.7 (2020), pp. 1888–1897.
[5] Avishek Biswas and Anantha P. Chandrakasan. “CONV-SRAM: An Energy Efficient SRAM With In-Memory Dot-Product Computation for Low-Power Convolutional Neural Networks”. In: IEEE Journal of Solid-State Circuits 54.1 (2019), pp. 217–230. doi: 10.1109/JSSC.2018.2880918.