Design and Implementation of an 8T-SRAM-Based Computing-in-Memory (CiM) Architecture for Efficient Addition of 8-Bit Word

Yatharth Agarwal
Department of Electrical and Computer Engineering
Purdue University
West Lafayette, USA
agarw414@purdue.edu

Shalvi
Department of Electrical and Computer Engineering
Purdue University
West Lafayette, USA
sshalvi@purdue.edu

***Abstract*—The conventional Von Neumann architecture, marked by the separation of processing and memory units, grapples with the intrinsic challenge of data transfer bottlenecks between these components. In response, Computing-in-Memory (CiM) architectures emerge as a solution, conducting computations within the memory to alleviate data movement constraints and amplify energy efficiency. This project introduces an optimized 8T-SRAM-based CiM architecture tailored for efficient 8-bit word addition operations. Employing concurrent activation of multiple Read Word Lines (RWLs), the architecture facilitates parallel bitwise computations. Voltage sense amplifiers strategically positioned within the design enhance precision by amplifying subtle voltage differences, ensuring swift and accurate addition operations. The initiative encompasses the comprehensive design, implementation, and evaluation phases, covering memory write and read functionalities, 8-bit addition computations, layout efficiency metrics, and simulation outcomes using RC extracted values. The successful showcasing of the proposed CiM architecture validates its operational efficacy and charts a course for advancements in in-memory computing, specifically in 8-bit word addition, offering promising prospects for applications in high-throughput computing scenarios.**

***Index Terms*—8T-SRAM, Computing-in-Memory, Sense Amplifier, Compute Module.**

I. Introduction

The rapid evolution of technology has led to the emergence of transformative applications, most notably in the realm of artificial intelligence (AI). The proliferation of AI applications has propelled us into the era of 'Big Data,' where the storage and processing of vast amounts of data have become imperative. To meet the demands of this data-intensive era, the efficiency of hardware architectures is paramount.

A fundamental challenge in contemporary von Neumann architectures lies in the separation of the memory unit and processing unit. This segregation results in a high volume of processor-memory transactions, limiting performance and energy consumption. As the demand for computational power continues to surge, seeking innovative solutions that break through these bottlenecks becomes imperative.

Addressing this challenge, computing-in-memory (CiM) emerges as a promising paradigm, offering an elegant solution by integrating computation capabilities within the memory subsystem. This integration aims to minimize the need for frequent data transfers between the memory and processing units, thereby enhancing overall system performance and reducing energy consumption.

In this research project, our focus revolves around designing and implementing an 8T SRAM-based computing-in-memory architecture. The primary objective is to create a memory system beyond conventional storage functions, incorporating the ability to perform in-memory addition of two 8-bit numbers. By seamlessly integrating computation into the memory macro, we endeavor to contribute to developing more efficient and performant computing architectures capable of addressing the challenges posed by the era of 'Big Data.

Our report is organized as follows -

- We discuss the design of the SRAM Circuit in Section II, the Sense Amplifier in Section III, and the Compute Module in Section IV.
- Section V presents the Layout for various sub-blocks and outlines critical design considerations for the project.
- Evaluations and Results are presented in Section VI and the Methodology for Parallel Computing is in Section VII.
- Section VIII discusses possible Optimizations within our implementation.
- Lastly, our research concludes with a comparative analysis of In-Memory Computing with Von Neumann architecture and Near-Memory Computing in Section IX. Subsequently, Section X outlines the individual contributions of each team member, and Section XI expresses our gratitude through acknowledgments.

II. Design of SRAM array

A. *Writing data to the 8T SRAM*

The 8T SRAM employs voltage states on storage nodes Q and Q_B to store binary data. During the writing process, the WBL and WBLB signals are set to appropriate levels

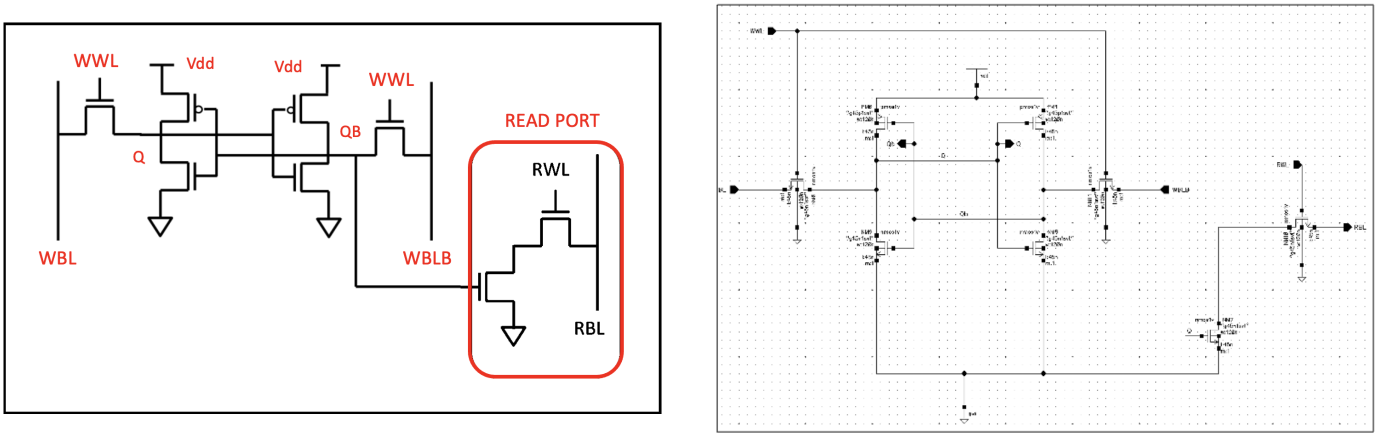


Fig. 1: The schematic illustrates the configuration of a typical 8T-SRAM bit-cell featuring separate ports for reading and writing operations.

(0 and V_{DD}, respectively, for writing Q='0'; and V_{DD} and 0, respectively, for writing Q='1'). Subsequently, WWL is activated, driven to V_{DD}, and the values of Q and Q_B are determined by the WBL and WBLB signals.

Fig. 1 depicts the operation of the 8T SRAM. A comprehensive overview of the voltage levels and their corresponding output values during the write operation is presented in Table I.

WBL	WBLB	WWL	Q	Q_B
0	V_{DD}	1	0	1
V_{DD}	0	1	1	0
0	V_{DD}	0	Q	Q_B
V_{DD}	0	0	Q	Q_B

TABLE I: Truth Table for the Write Operation

B. *Pre-charge Circuit*

In the Precharge Circuit, a PMOS transistor is employed, with its drain and source terminals connected to the supply voltage (V_{DD}) and the Read Bit line (RBL), respectively. The purpose of this circuit is to precharge the RBL to the V_{DD} level in preparation for a read operation. This pre-charging action is initiated by applying a low signal to the gate of the PMOS, lasting for a duration of 200 ps.

The PMOS transistor has been designed with a total width of 600 nm, a size determined to accommodate the substantial capacitance of the RBL. This sizing strategy serves a dual purpose: first, it enhances the overall drivability of the circuit, ensuring efficient operation, and second, it reduces the time required to precharge the RBL, thereby optimizing the speed of the pre-charging process.

C. *Reading data from the 8T SRAM*

For reading, the read port (Fig. 1) is utilized to sense the data on RBL by asserting RWL. If Q='0' (Q_B ='1'), V_{QB} is set to V_{DD}, and the read port conducts with both transistors in the ON state. This discharges RBL from its pre-charged value of V_{DD}, resulting in V_{RBL} = V_{DD}-Δ. Conversely, if Q='1' (Q_B ='0'), V_{QB} is set to 0, preventing the read port from conducting as the bottom transistor is OFF. This maintains RBL at its pre-charged value of V_{DD}, leading to V_{RBL} = V_{DD}. A voltage-based sense amplifier can interpret the sensed data discussed in the next section.

III. Design of the Sense Amplifier

The voltage-based sense amplifier utilizes p-type transistors (P3 and P4) (Fig. 2) configured with their source terminals directly connected to the Read Bit Line (RBL) and the Reference Voltage (V_{REF}), respectively. In this configuration, the gate terminals of P3 and P4 are interconnected to nodes X and $\overline{X}$, creating a differential setup. Initiated with the Sense Enable (SEN) signal at zero, P3 and P4 remain inactive initially. The voltages on nodes X and $\overline{X}$ are contingent upon the RBL voltage V_{DD} and the Reference Voltage (V_{DD} - Δ/2). This biased condition establishes a voltage difference between nodes X and $\overline{X}$, setting the stage for differential sensing.

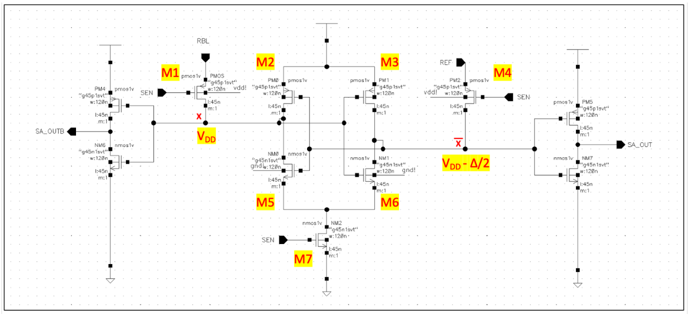


Fig. 2: Voltage-based Sense Amplifier

Upon the transition of the Sense Enable (SEN) signal from zero to V_{DD}, P3 and P4 become active. The transistor with the higher gate voltage, corresponding to the higher voltage node, dominates, discharging the bit line (X or $\overline{X}$) more rapidly. This discharge initiates a cascading effect, leading to one of the inverters dominating and driving the sense amplifier outputs, namely OR_OUT/AND_OUT and OR_OUTB/AND_OUTB. The resulting outputs represent complementary values based on the differential input and the race condition between the two transistors.

This voltage-based sense amplifier topology offers advantages in simplicity and efficient handling. Its direct connection of source terminals to voltages eliminates the need for cross-coupled inverters, simplifying the overall sensing process. The configuration naturally handles pre-charging without explicit precharge steps, contributing to the efficiency of the voltage-based sense amplifier, particularly in scenarios where simplicity and efficient handling of specific memory configurations, such as 8T SRAM, are paramount.

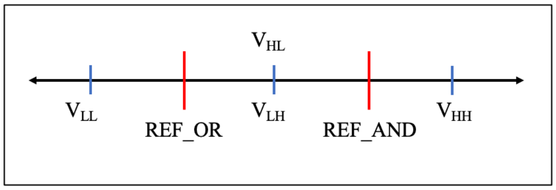


Fig. 3: Reference Voltages for the Sense Amplifier

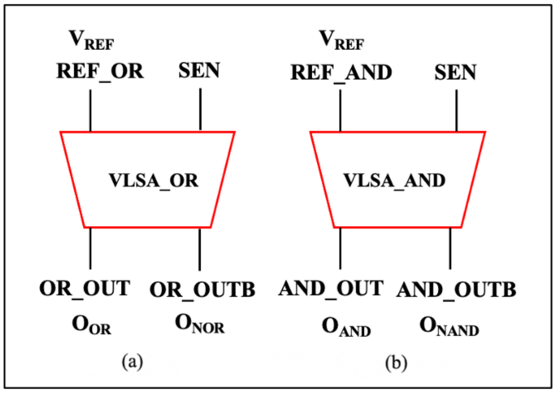


Fig. 4: (a) Bitwise OR Sensing Scheme. (b) Bitwise AND Sensing Scheme

In Figure 2, M2-M5 and M3-M6 create the inverters responsible for converting the differential voltage on the bit lines into a full swing at the output. The bit lines in this design facilitate the pre-charging of internal nodes. [1] A notable advantage of this configuration compared to the Current level sense amplifiers is its reduced transistor count, leading to quicker access times and a more compact footprint. [2]

V_{SL}	O_{OR}	O_{NOR}	O_{AND}	O_{NAND}
V_{LL}	0	1	0	1
V_{LH}	1	0	0	1
V_{HL}	1	0	0	1
V_{HH}	1	0	1	0

TABLE II: Truth Table for the VLSA

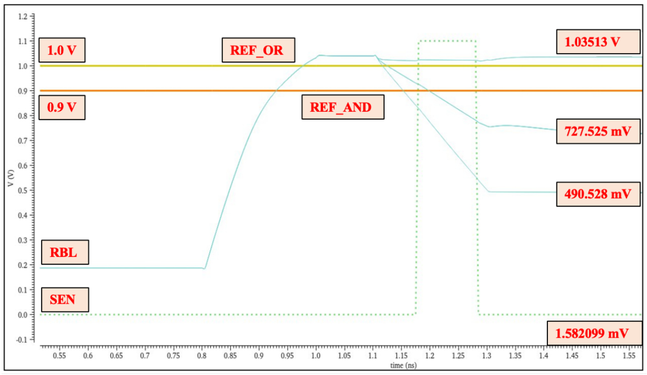


Fig. 5: Timing Diagram for the Sense Amplifier; The figure showcases the different levels of discharge on RBL based on inputs and the relative timing wrt reference and sense signals.

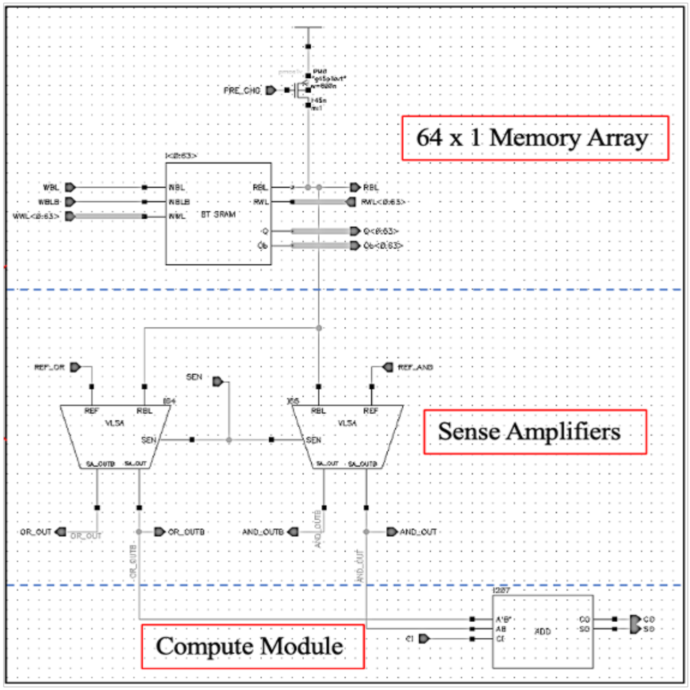
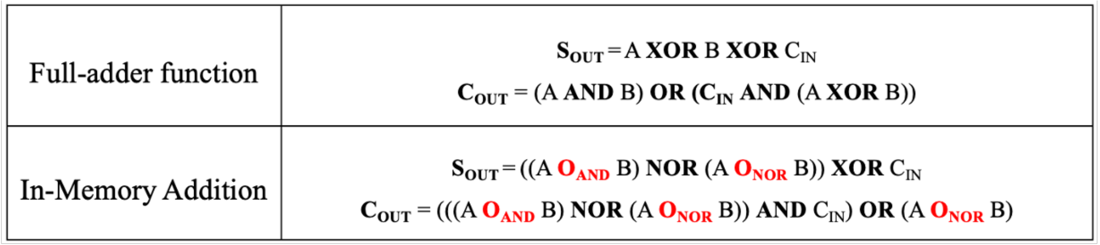


Fig. 6: 8T 64x1 SRAM Operation

IV. In-memory Computing

CiM's computational outputs seamlessly intertwine with near-array or peripheral Compute Modules (CMs). This integration allows for the derivation of sum and carry-out bits for each column, facilitating a coherent and efficient data flow within the memory hierarchy.

The Compute Module manages the propagation of carry-bits from the least significant bits (LSB) to the most important bits (MSB), intricately imitating the well-established procedure in standard adders. This efficient convey spread instrument guarantees the exact calculation of two numbers' amounts, keeping consistent with traditional number-crunching tasks.



Full-adder function	S_{OUT} = A **XOR** B **XOR** C_{IN} C_{OUT} = (A **AND** B) **OR** (C_{IN} **AND** (A **XOR** B))
In-Memory Addition	S_{OUT} = ((A O_{AND} B) **NOR** (A O_{NOR} B)) **XOR** C_{IN} C_{OUT} = (((A O_{AND} B) **NOR** (A O_{NOR} B)) **AND** C_{IN}) **OR** (A O_{NOR} B)

Fig. 7: Computing an In-memory addition

A. *Single-Cycle Computations vs. Conventional Approaches*

CiM's distinctive capability for executing computations within a single cycle starkly contrasts conventional methodologies, where the standard approach typically demands two cycles for reading and subsequent bitwise operations. This efficiency gain positions CiM as a promising paradigm for accelerated data processing.

V. Layout

A. *8T SRAM - Thin cell*

The 8T SRAM cell was designed using the thin cell layout and is presented in Figure 8.

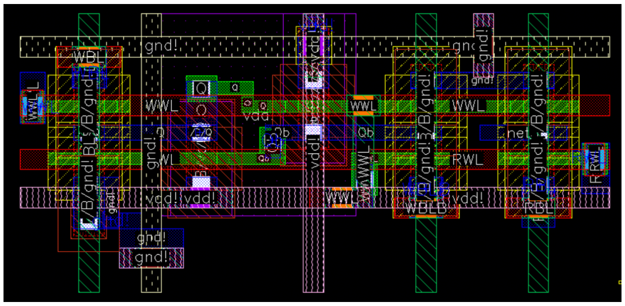


Fig. 8: Layout of the 8T SRAM cell

B. *Sense amplifier and Compute module*

CMOS design methodology was used for the VLSA and 1-bit full adder. VLSA and Adder layouts are presented in Figure 9 and Figure 10, respectively.

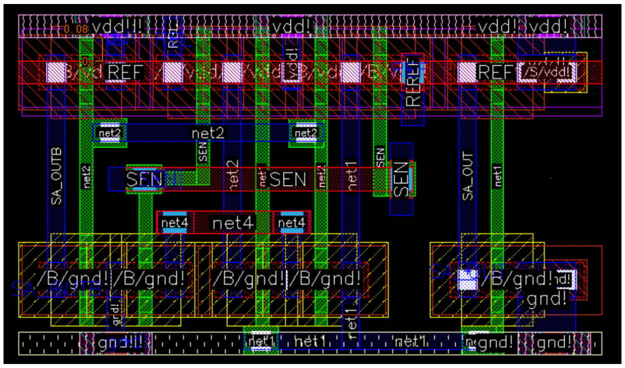


Fig. 9: Layout of the Voltage-based Sense Amplifier

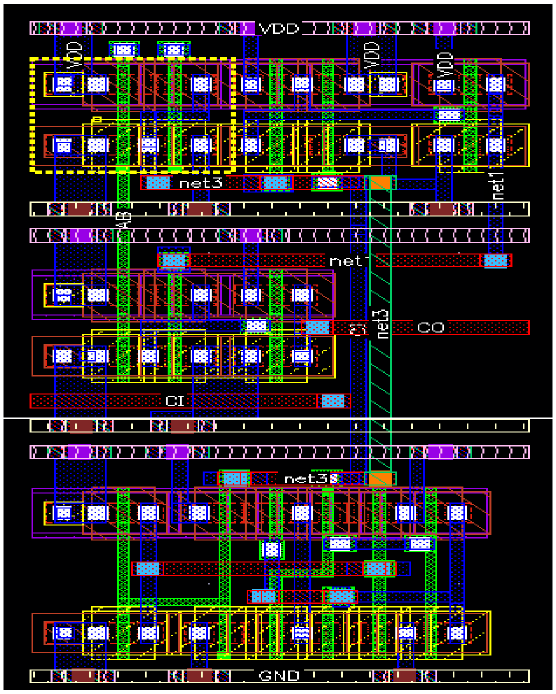


Fig. 10: Layout of the Compute Module

C. *Design Strategy*

The memory cell architecture demonstrates a highly repeatable design paradigm, facilitating rapid scalability. The foundational unit consists of an 8T SRAM cell. Concurrently, cells corresponding to logical operations, namely AND, XOR, OR, and NOR gates, were designed. Notably, the AND, OR, and NOR gates share a standard height, while the XOR gate exhibits double the size.

Leveraging these fundamental logic cells, an Adder was designed and seamlessly integrated into a vertical column comprising 64 8T SRAM cells, accompanied by 2 Voltage Level Sense Amplifiers VLSA and a precharge cell. The routing of the Read Bitline RBL, Write Bitline (WBL), and Write Bitline Bar WBLB occurred in the vertical direction, while the Read Wordline RWL and Write Wordline WWL were horizontally positioned. The column's design incorporated tolerances and placement considerations, ensuring compatibility within an 8x64 cell without necessitating additional routing.

Eight such blocks were serially arranged to achieve the targeted 64x64 configuration, and the Carry-In (CIN) and Carry-

Out (CO) signals were chained among the cells, completing the overall design.

This design methodology, characterized by its systematic modular approach, enabled the expeditious realization of a 4-kilobyte (KB) memory within the stipulated timeframe for the project. However, it is imperative to acknowledge that the current design may not optimize contact-sharing efficiency, a topic explored more comprehensively in Section VII of this academic paper.

D. *Routing Strategy*

In the Layout of our integrated circuit (IC), the strategic utilization of metal layers is a crucial consideration, emphasizing the concept of metal layer directionality. Notably, Metal 1 (M1), Metal 2 (M2), and Metal 3 (M3) are employed for signal routing, with M2 primarily facilitating horizontal interconnects and M3 dedicated to vertical routing. This deliberate directionality streamlines the manufacturing process and ensures consistency while capitalizing on optimized resistance and capacitance for efficient signal propagation. Moreover, M4 is exclusively designated for establishing the V_{DD} grid, contributing to robust power distribution. Concurrently, M5 is allocated explicitly for constructing the GND grid, thereby augmenting the overall reliability and functionality of our IC design.

VI. Results

Our circuit successfully implements a 64x64 array of 8T SRAM cells capable of parallel performing in-memory addition of eight 8-bit words.

A. *Area*

The computed area of the 64x64 SRAM array with the Compute module is dictated by specific metrics, with a measured length of 147.84 µm and a corresponding height of 77.365 µm. The resultant area is measured at 11,439.24 $\mu\mathrm{m}^2$ units, constituting 90.1% of the design. These metrics serve as quantitative indicators, providing a comprehensive understanding of the physical dimensions of our IC. The Compute module comprises the VLSA adder and 8.9443%, whereas the precharge circuit takes up 0.885% of the total area. Table III presents the overall breakdown of the measured area.

Module	**Length** (µm)	**Height** (µm)	**Area** ($\mu\mathrm{m}^2$)	**Total** (%)
Pre-charge	147.84	0.685	101.27	0.885
Compute Module	147.84	6.92	1023.05	8.943
SRAM Array	147.84	69.77	10314.92	90.172
Total	147.84	77.365	11,439.24	100.00

TABLE III: Module Dimensions and Area Allocation

B. *Write Latency*

In our design, the efficiency of memory operations hinges on the careful consideration of write latency. Specifically, our system achieves a write latency of 27 ps following the Write Word Line (WWL) assertion. This duration represents the time required for a successful write operation. The write latency is measured as the duration after which Q and Q_B reach stability on the assertion of WWL. Figure 11 presents the timing diagram for writing data.

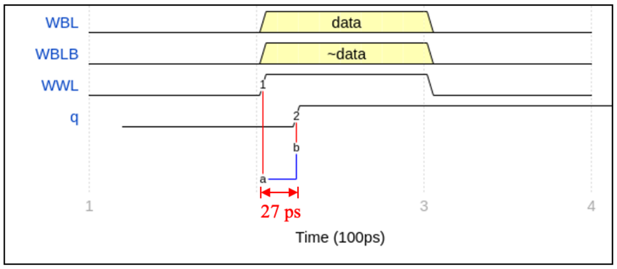


Fig. 11: Timing Diagram for the Write Operation

C. *Read Latency*

Ensuring the accuracy of identified values in our computational framework requires carefully deploying the sense signal at approximately 75 ps [Fig. 12] after initiating the Read Word Line (RWL). This timing is critical for the VLSA to latch and propagate the computed logic between 2-word lines correctly. When evaluating the overall latency for worst-case scenarios, the upper cells (WWL0 and WWL1) of the last column (RBL$<$ 0:7 $>$) are engaged in full ripple addition, specifically addressing the operation 0xFF + 0x00 with a Carry-in (C_{in}) as 1. The latency for obtaining the SUM is determined to be 823 ps, while the Carry-out (C_{out}) latency is measured as 886 ps.

D. *Energy Consumption*

Memories typically consist of four major energy consumptions: Energy associated with reads (RBL), Energy associated with writes (WWL), Energy associated with peripheral circuits, and Energy associated with internal capacitors.

Energy	Single-bit	64-bits
Write Energy	110 pJ	7.6 nJ
Read Energy	210.8 pJ	13.491 nJ

TABLE IV: Energy Consumption

In our analysis, we present the Energy associated with WWL by integrating the product of the current on each word line with V_{DD} while storing 1 to all 64 lines. We also present the total computed Energy calculated from the assertion of RWL till the Sum is obtained. This is calculated by adding the individual energies consumed by the VLSA. Energy of each VLSA is computed by integrating the product of the Current at REF with V_{DD} when 0xFF is added with 0x00 for all eight words. The results are tabulated below in Table IV.

E. *DRC and LVS Checks*

The execution of our design is characterized by a successful completion of the Layout versus Schematic (LVS) and Design Rule Check (DRC) processes.

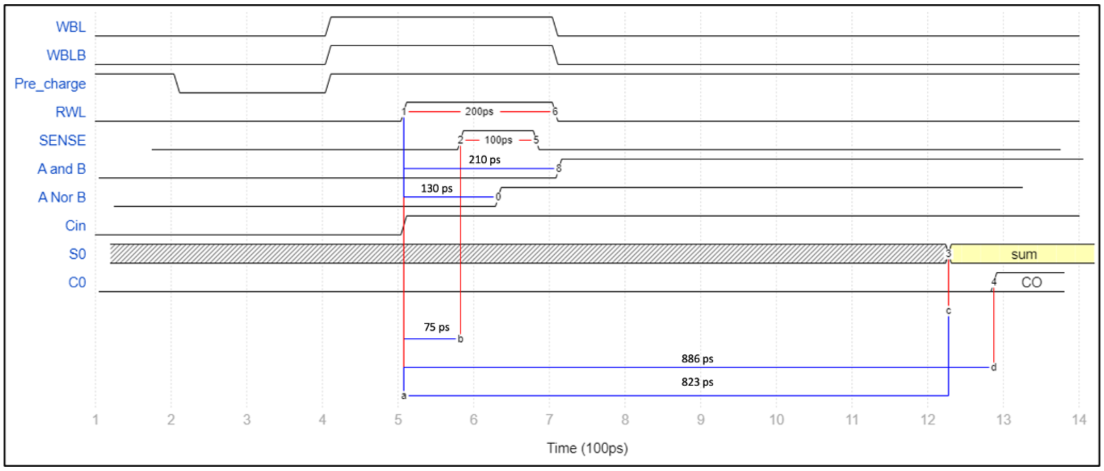
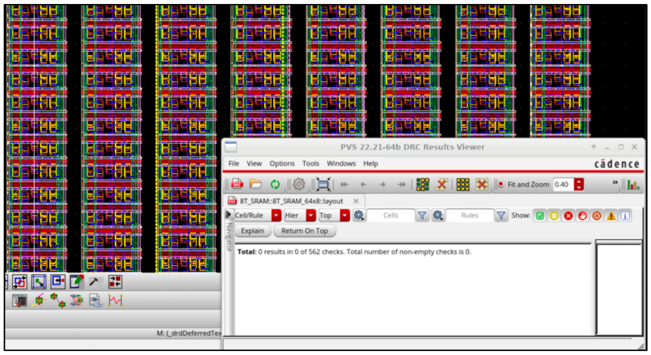
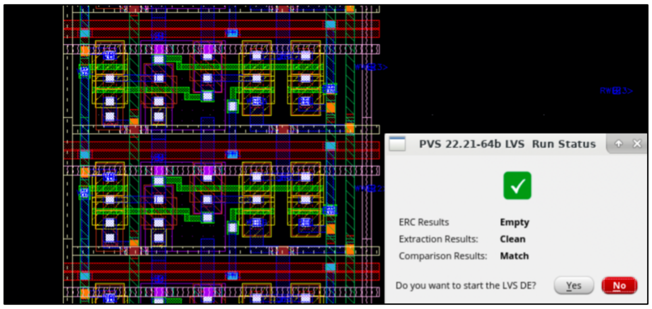


Fig. 12: Timing Diagram for the Read Operation


PVS 22.21-64b DRC Results Viewer
File View Options Tools Windows Help
cādence
Fit and Zoom 0.40
8T_SRAM::8T_SRAM_64x8::layout
Cell/Rule Hier Top Cells Rules Show:
Explain
Return On Top
Navigator

Total: 0 results in 0 of 562 checks. Total number of non-empty checks is 0.

Fig. 13: Result of Design Rule Checks (DRC)



ERC Results: Empty
Extraction Results: Clean
Comparison Results: Match

Do you want to start the LVS DE? Yes No

Fig. 14: Result of Layout v/s Schematic (LVS) checks

VII. Parallel Compute

We successfully executed computations involving eight 8-bit words within a single cycle by harnessing the parallel processing capabilities inherent in the Computing-in-Memory (CiM) architecture. The array structure, with each row containing eight words and a compute module for each column. 64 bits of data can be added parallelly.

This functionality was achieved by ensuring the width of the compute module was less than the width of a single 8T SRAM cell, which allowed each column to have its own Compute module.

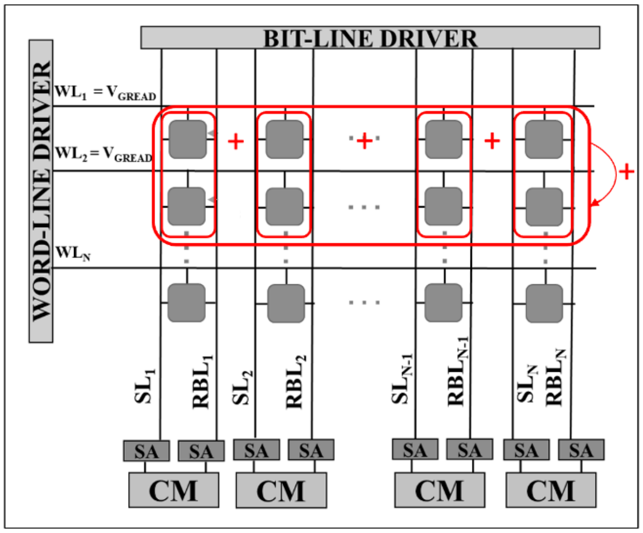


Fig. 15: Parallel Addition of 8 8-bit Words from Two Rows in a Single Cycle

VIII. Optimizations

A. *Contact sharing*

In CMOS layout design, using 8T SRAM cells with a thin cell layout is a promising approach to enhance connectivity and minimize silicon footprint. The unique characteristic of the 8T SRAM cell design allows for efficient vertical contact sharing between the Read Word Line (RWL), Write Bit Line (WBL), and Write Bit Line Bar (WBLB), leading to a more compact arrangement. Additionally, the thin cell layout facilitates row-wise column sharing between the RWL and Write Word Line (WWL), further optimizing the use of available space. This innovative configuration streamlines the physical Layout of SRAM cells and contributes to improved performance and reduced power consumption, making it a valuable design choice in advanced CMOS integrated circuits.

Unfortunately, during the project, we had to overcome

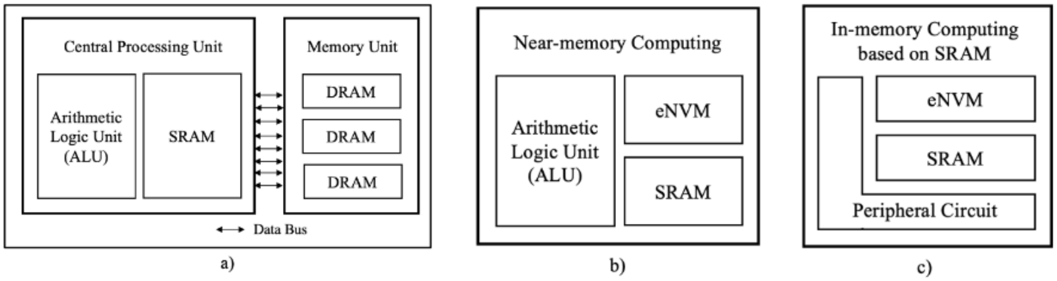


Fig. 16: (a) Von Neumann architecture - CPU and memory are distinct, linked by a bus, (b) Near-memory computing - integrates nonvolatile memory on the same silicon as the CPU, enhancing bandwidth, and (c) In-memory computing - SRAM-based, with computation directly within the memory array.

numerous challenges to put the thin cell structure with vertical contact sharing into practice. The design process was hindered even with its potential advantages due to the discovery of numerous N-Well (NTAP) and P-Well TAP (PTAP) Design Rule Check (DRC) problems. Due to the project's time constraints, fixing these problems would require a significant redesign, impairing the project's ability to meet its deadlines. As a result, we had to defer the implementation of row-wise column sharing and vertical contact sharing in our 64x64 8T SRAM cells.

B. *Additional Margins for latency's accounting for Non-ideal Drive signals*

In our investigation, we assumed that all input signals were ideal, with a rise and fall period of 10 ps. However, real-world signals exhibit inherent nonidealities. It becomes essential to consider variations in the rise and fall durations to guarantee the effective functioning of the SRAM array, especially for crucial signals like the Sense Enable (SEN), ensuring that the sensing amplifier latches onto the correct values.

To simulate nonidealities, buffering all input signals with inverters could introduce delays. While these tests are crucial for production projects, they were beyond the scope of this particular project.

IX. Comparison with Near-Memory Computing and Von-Neumann Architecture

In the von Neumann architecture, a foundational computing model, denoted as (Fig. 16 (a)), extensive data is stored within a memory unit intricately linked to the computational unit through a data bus. The persistent data flow between these processing and memory components is a predominant bottleneck, characterized by constrained bandwidth, prolonged latency, sequential data processing, and elevated energy consumption. [3] [4]

In contrast, near-memory computing (Fig. 16 (b)) represents a departure from the von Neumann model by integrating processing elements closely with memory units [5]. This architectural approach aims to mitigate data transfer overhead by enabling parallelism through the execution of computations in proximity to where data is stored. Near-memory computing is designed to enhance energy efficiency, reduce latency, and leverage data-centric computation models, offering advantages over traditional von Neumann architectures, especially in scenarios with high memory access patterns. [5]

In-memory computing, another innovative paradigm, takes a step further by performing computations directly within the memory space. This eliminates the need for frequent data movement, a characteristic challenge of von Neumann architectures. In-memory computing optimizes performance by capitalizing on high-speed memory access, which is particularly beneficial for data-intensive tasks. It exploits parallelism within the memory architecture, significantly improving overall throughput and energy efficiency.

Distinguishing these architectures lies in their data movement, parallelism, energy efficiency, and latency approaches. With their sequential nature, Von Neumann architectures often face challenges related to data transfer delays and energy inefficiencies. Near-memory computing addresses these issues by minimizing data movement and introducing parallelism, enhancing energy efficiency and latency. [6] However, with its direct computation within the memory space, in-memory computing surpasses optimal solutions for data-centric computing, large-scale analytics, and memory-bound applications.

Metrics such as throughput improvement and energy efficiency can be used to demonstrate the superiority of in-memory computing. Studies have shown a significant increase in throughput, with values reaching up to 3 times faster than traditional von Neumann architectures. Energy efficiency metrics also favor in-memory computing, with reductions in power consumption by up to 20% compared to conventional models. [7] These metrics highlight the superior performance of in-memory computing, making it a compelling choice for modern computational needs.

X. Conclusion

In conclusion, our research has successfully navigated the intricate landscape of Computing-in-Memory (CiM) architectures, emphasizing the transformative potential of integrating computation capabilities directly within the memory subsystem. The optimization of 8T-SRAM cells for parallel 8-bit word addition not only addresses the persistent challenges of data transfer bottlenecks in traditional von Neumann architectures but also paves the way for more efficient and performant computing solutions. Our design, implementation, and evaluation phases have demonstrated the operational efficacy of the proposed CiM architecture, providing valuable insights into the delicate balance between memory efficiency and computational speed.

While our work showcases promising results in terms of parallel computing capabilities and operational efficacy, it is important to acknowledge the challenges we encountered, such as those related to thin cell structure with vertical contact sharing. Despite these hurdles, our research contributes valuable insights to the evolving landscape of in-memory computing, providing a foundation for future advancements in efficient and performant computing architectures, particularly in the context of 8-bit word addition operations. As technology advances, the exploration of in-memory computing models becomes increasingly crucial for overcoming the limitations of traditional architectures and unlocking new possibilities for high-throughput computing scenarios.

XI. Contribution of each team member

A. *Shalvi*

- 8T SRAM Schematic.
- One word (64x8 array with CiM) and Full design (64x64 Array with CiM) Schematic.
- Logic Gates and Adder Layout.
- Final Design Layout.
- Adder functional simulation and 8T SRAM functional simulation.

B. *Yatharth Agarwal*

- Adder and VLSA Schematic.
- Single column (64x1 array with CiM) Schematic.
- VLSA and 8T SRAM Layout.
- Single column Layout and One word (64x8 array with CiM) Layout.
- VLSA functional Simulation and Single column functional simulation.
- Full design (64x64 Array with CiM) Post-extraction simulation.

XII. Acknowledgements

We would like to express our sincere gratitude to Professor Sumeet Gupta for his invaluable guidance and insightful input throughout the duration of this project. His expertise and mentorship have been instrumental in shaping the direction of this work.

We extend our heartfelt thanks to our Teaching Assistants, Bhargav and Ashish, for their unwavering support and patience in clarifying doubts during various project stages. Their dedication has contributed significantly to the successful completion of this project.

Special thanks to the Department of Electrical and Computer Engineering and the ECN for providing access to Cadence tools and offering invaluable support in the setup of these tools.

We are also grateful to our classmates on Piazza for their collaboration and knowledge-sharing. A particular mention goes to Patrick Bradley for generously sharing a highly beneficial tutorial on the automatic placement of multiple pins in a layout. Such collaborative efforts within the academic community have enriched the quality of our project.

Appendix

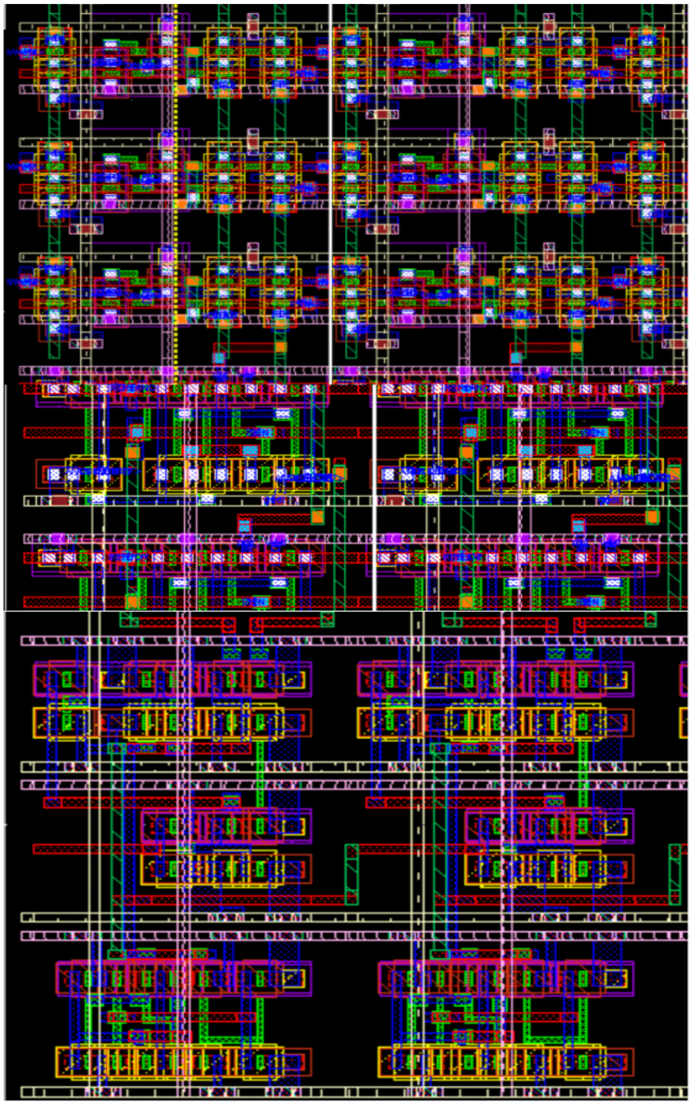


Fig. 17: The arrangement consists of three 8T SRAM cells organized in two columns, accompanied by the two Voltage Latch Sense Amplifier (VLSA) and Compute Module for each column.

References

[1] J. K. H. J. Taehui Na, Seung-Han Woo and S.-O. Jung, "Comparative study of various latch-type sense amplifiers," *IEEE*, vol. 22, pp. 425 – 429, 2014.

[2] K. L. Baker Mohammad, Percy Dadabhoy and P. Bassett, "Comparative study of current mode and voltage mode sense amplifier used for 28nm sram," *IEEE*, pp. 1–6, 2005.

[3] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," *IEEE*, vol. 26, pp. 10–14, 2014.

[4] E. T. T. M. Conte and E. DeBenedictis, "Rebooting computing:new strategies for technology scaling," *IEEE*, vol. 48, pp. 10–13, 2015.

[5] T. X. Y. Z. e. a. Zhenhua Zhu, Hanbo Sun, "Mnsim 2.0: A behavior-level modeling tool for processing-in-memory architectures," *IEEE*, vol. 42, pp. 4112–4125, 2023.

[6] A. A. Akhilesh Jaiswal, Indranil Chakraborty and K. Roy, "8t sram cell as a multibit dot-product engine for beyond von neumann computing," *IEEE*, 2019.

[7] R. M. Manuel Le Gallo, Abu Sebastian, "Mixed-precision in-memory computing," *IBM Research - Zurich*, 2018.