559 - VLSI Project Report

1st P. Bradley
dept. Electrical and Computer Engineering
Purdue
Crampagna, France
bradl129@purdue.edu

2nd Jacob Martel
dept.Electrical and Computer Engineering)
Purdue
Texas, United States
martel0@purdue.edu

***Abstract*—Due to technological advances, alternative hardware architectures need to be created to mitigate the bottleneck issue created by the CPU and memory unit being separate entities. To alleviate this problem, we have designed a 64x64 array that offers computing in memory. This design uses 8-bit words created by 8TSRAM cells with the capability to perform in-memory addition of two words using peripheral circuitry. This design consists of worst-case latency for the write function being 38.43ps, for the read function being 180ps, and for the in-memory computing being 1.174ns. The total area used by the SRAM array was 3885.68μm^2, while the entire design had an area of 5037.32μm^2 making the array efficiency 77%**

***Index Terms*—8TSRAM, Array, Latency, Sense Amplifier, Adder**

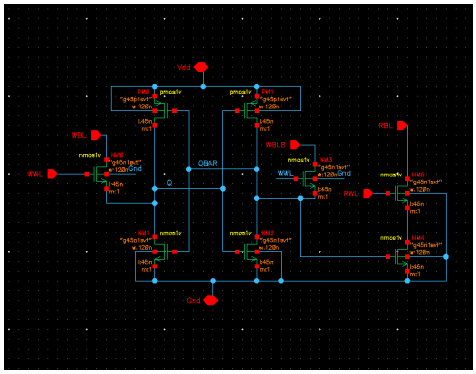

Fig. 1. 8TSRAM Schematic

I. Introduction

The design principles of the von Neumann architecture have been prolific in the digital age. Still, its limitations have become apparent with the rise of AI and other computationally intense applications. An issue with this architecture is the bottleneck created when the CPU needs to interact with the memory constantly. Resulting in considerable energy and performance losses. A solution to this bottleneck is to implement Computing-in-Memory(CiM). Instead of having separate compartments for processing and memory, some of the computation is done in the memory itself. Having localized logical computational circuitry will increase performance and decrease the need for the CPU to interact with remote memory units. This paper discusses the implementation and design of in-memory computation that performs parallel addition.

II. SRAM Design

A. Schematic and Functionality Testing

The basic design of the 8TSRAM cell, as seen in Figure one, was developed largely in line with Jaiswal's 8TSRAM layout and design. [1]. The 8TSRAM comprises of a standard 6TSRAM with two auxiliary read circuits. The advantage of this memory design is that it is ratioless and more reliable than the more common 6TSRAM cell. Also, because it is ratioless, it is possible to use the minimum-sized MOSFETS throughout the memory array. The general operation of this memory is as follows: for the write cycle, the WBL and WBLB lines are asserted in a complementary fashion, and then the WWL line is pulsed, which forces Q and Q Bar to be driven to the new value. Next, when the WWL Line goes low again, the cross-coupled inverter takes action and holds the new value. Finally, WBL and WBLB are driven to zero to prevent interference with the cross-coupled inverter. The read operation starts with charging the RBL line and then asserting the read word line. RBL will discharge through N4 if Q Bar equals one, implying that Q equals zero. From here, the RBL line is measured using a sense amplifier. By using multiple sense amplifiers, it is possible to measure if more than one bit in a column contains a one-on Q. In theory and under ideal conditions, it would be possible to measure N bits simultaneously. However, the number of bits that can be added is limited by the minimum noise margin of the sense amplifier as well as the sizing constraints of the compute module.

During the testing of both the layout and schematic, it was discovered that initializing Q and Q Bar could be avoided by starting WBL and WBLB in a complementary arrangement. This approach led to Q and Q Bar being driven randomly to zero or one during the simulation. Alternatively, not following this approach initially led to Q and Q Bar being simulated at fixed intermediary voltage levels. To prevent this issue, an initial write cycle was used to set the initial circuit conditions for subsequent testing.

B. Hierarchical Design

The goal of implementing the hierarchy is to reduce the steps required to develop the higher stages while retaining enough flexibility at the higher levels for testing and customization.

Level 1	Bit Cell
Level 2	8-Bit Word
Level 3	64 Word Column
Level 4	64 Word Column & Adder Unit
Level 5	The 64x64 Bit Array & Parallel Adder Unit

TABLE I
THE HIERARCHY OF THE 64X64 MEMORY ARRAY.

The process of revision and development involved creating a schematic and testing it for operation. By creating a test functional schematic design, it was possible to make copies of the design and revise the lower layouts, such as the bit cell, to improve the development of higher levels. This process considerably reduced the development time.

To aid in the division of labor, we designed around a theoretically optimized 64-word column. It was apparent from the first column design that the pin optimization was necessary; as such, we designed the sense amplifier and adder unit using a limited area budget based on the expected optimum layout of the word column. This ensured that no matter how optimized the final design of the memory array was, the area efficiency target would be achieved. The target area budget for a 2-bit adder circuit was $4.2\mu m * 7.9\mu m = 33.18\mu^2 m^2$, based on the assumption that the area of a 2-bit wide column with maximum pin sharing was approximately $77.12\mu^2 m^2$. In theory, the area for the full memory array is estimated to be $144.005 * 26.620 = 3833.413\mu^2 m^2$, and the total area used by the adder unit is estimated to be $144 * 7.6 = 1031.04\mu^2 m^2$, resulting in an estimated area efficiency of 79%.

The final design has two layout options: one where all the WWL and RWL lines are paired along each row and one where they remain separated, allowing for arbitrary word selection in the array. To achieve parallelism in the second design, all that needs to happen is the assertion of all word lines in a row simultaneously. Both designs are functional and achieve the area efficiency requirements.

C. Layout Design & Evolution

Four layouts were designed and tested in the development of the memory array. The first design was essentially a proof of concept. Crudely assembled and to demonstrate the design's fundamental capacity to retain information. The second design was to tidy up and clean the connections to reduce the difficulty of assembling the full memory array. However, it could not share pins due to the external wiring of the VDD and ground pins. Additionally, the internal pins for Q and Q Bar were too close to the exterior, preventing pin sharing. Designs 1, 2, and 4 were all implemented in 64-word columns. Based on column designs 1 &2, it was determined that there were several significant improvements to be made. It was possible to improve pin and contact sharing significantly, but it would require the number of body ties in the lower hierarchical structure to be removed. The removal of body ties in the lower hierarchy in design 3 results in DRC and LVS errors at lower hierarchical stages of the memory array. As such, design 3 is the design with DRC and LVS clear word and bit designs, whereas in design 4, the oxide ties are removed in the word and bit designs and then added back in at the 64-word column level. Design 4, as seen in figure 2, is the final optimized

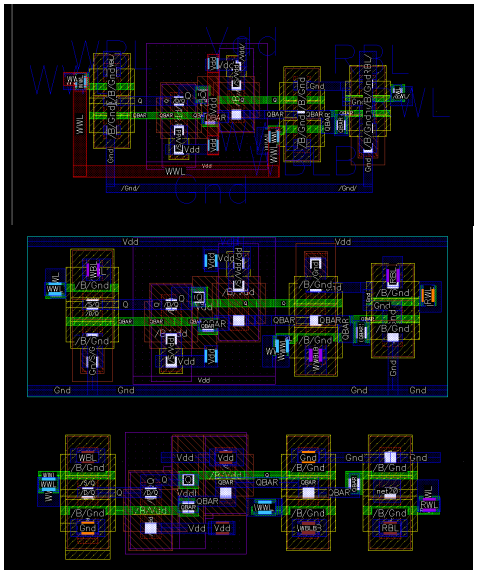

Fig. 2. 8TSRAM Layout Evolution Descending Order Version: 1, 2, & 4.

design for the memory array layout. Its features include the following: all of the external pins are placed precisely in parallel to one another, which allows for maximum contact sharing. However, this comes with a drawback in that the layout needs to be widened slightly for the internal contact pins for the write word line (WWL) and Q Bar to prevent interference between rows. From this, it was possible to reduce the area used by over 70% compared to designs 1 and 2 for a full word Column. Additionally, it uses five metal layers to route the internal connections for GND, Vdd, Q, Q Bar, WWL, and RWL. The rule for the design is that metal five is exclusively utilized for vertical connections between rows: Vdd, RBL, WBLB, and WBL. Metal 1, 2, 3, and 4 are utilized horizontally to connect GND, RWL, and WWL between bits in a word. GND is vertically connected in a word column via top and bottom pin sharing on the left and right of the memory cell. The word column is the fundamental unit of the final memory layout design; see Figure 3. It is designed to be implemented in two fashions higher in the hierarchy. Where all the WBL and RBL lines are connected across the eight words in the row or where they are disconnected between words. The advantage of the disconnected implementation is that while it can perform parallel editions, it can also perform nonparallel additions between separate words within each column. However, this comes with a disadvantage where each column has its word-select lines. The design of the column is such that the pins on the left and right can be shared between rows separately depending on the desired implementation.

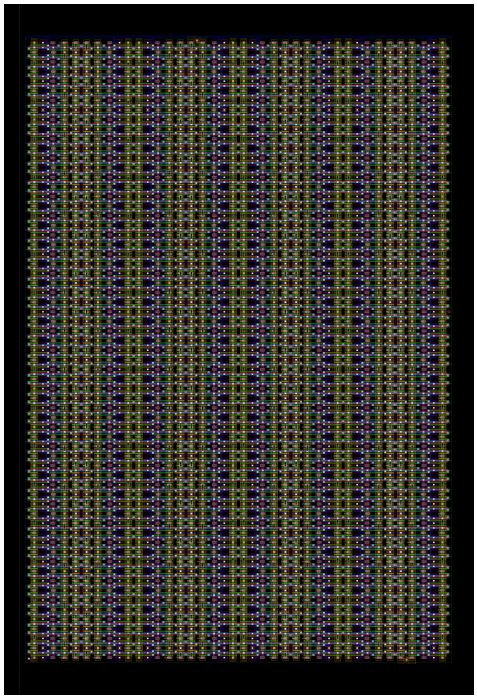

Fig. 3. The Full 64-Word Column Layout

III. Sense Amplifier

A. Design

The sense amplifier circuitry is an integral component for the read operation in SRAM cells. When data is read from a memory cell, the sense amplifier's role is to sense the low-power signal from the bit line, amplify the voltage swing to logic levels so the data can be used in the peripheral circuitry, reduce the energy consumed during the read cycle, and reduced the memory access propagation delay. There were two sense amplifier designs considered: a voltage-sensing or a current-sensing amplifier. Theoretically, the current base sense amplifier is faster; however, it would have required more development time. As such, the voltage-based sense amplifier was implemented for its relatively quick development lead time. Two further designs of the voltage sense amplifier were developed in parallel: a pull-up sense amplifier and a pull-down sense amplifier. Figure 4 shows the pull-up-based design while Figure 5 shows the pull-down-based design. The voltage-based sense amplifier is a differential circuit taking a reference voltage and the precharged RBL line from the SRAM array as the two inputs. This particular implementation of a sense amplifier produces complementary output signals when paired with the memory array, producing four separate logical expressions depending on the state of the RBL line, as seen in table II. Because of this, the sense amplifier can perform some of the computational processing of the full adder unit, reducing the computation time. For each bit column in the SRAM array, two sense amplifiers perform different logic functions based on their referenced voltages. For one sense amplifier circuit, the reference voltage will be 1.05V, and the other will be 950mV. The RBL voltage will be dependent on what is stored in the array. For each bit that is read at Q_{Bar} in the array, the RBL will discharge 100mv. When adding two bits that are high together, the RBL line will discharge VDD-200mV. The first sense amplifier will create OR/NOR logic, and the second sense amplifier will create AND/NAND logic depending on what is discharged on the RBL line. Table II depicts how this process works.

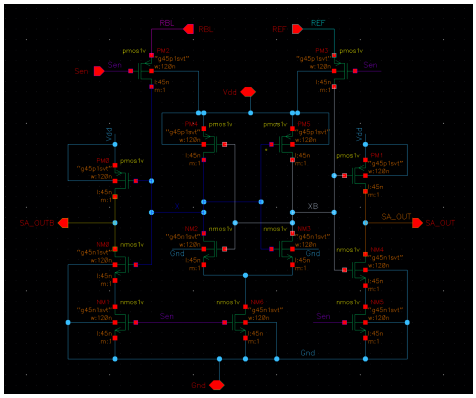

Fig. 4. Pull Up Sense Amplifier Design

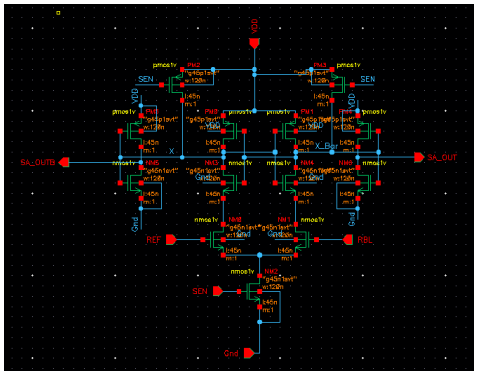

Fig. 5. Pull Down Sense Amplifier Design

A	B	I1	I2	OR	NOR	AND	NAND
0	0	I_{HRS}	I_{HRS}	0	1	0	1
0	1	I_{HRS}	I_{LRS}	1	0	0	1
1	0	I_{LRS}	I_{HRS}	1	0	0	1
1	1	I_{LRS}	I_{LRS}	1	0	1	0

TABLE II
Sense Amplifier Logic

From Table II it can be noted that the first sense amplifier follows OR/NOR logic because as long as one bit is high (low resistance) in the SRAM array, node X (from figures 4 and 5) will discharge faster than the opposite node (X_{BAR}) providing the OR functionality on the output SA_{OutB} and the NOR functionality on the other SA_{Out}. Similarly, the second sense amplifier follows AND/NAND logic because it needs to read two bits to have the AND functionality at the output SA_{OutB} while the inverse will happen at SA_{Out}.

B. Simulation Results: OR/NOR Functionality

The OR/NOR functionality was tested first for both the pull-up and pull-down designs. Figure 6 shows the results when RBL discharges by 100mV for the pull-up design. Figure 7 shows the results when RBL discharges by 100mV for the pull-down design. (The reference voltage for both designs was VDD-50mV).

Fig. 6. Pull-Up Design: RBL Discharges By 100mV

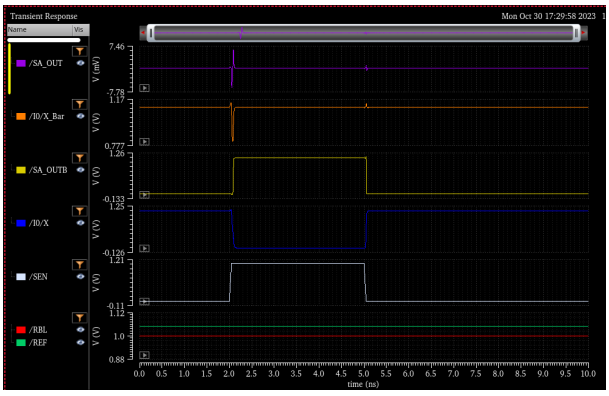

Fig. 7. Pull-Down Design: RBL Discharges By 100mV

Both circuits achieved the desired functionality. When RBL discharges by at least 100mV, the X node is intended to discharge first, and SA_{OutBar} is intended to reach VDD. However, in the pull-up design, the SA_{OutBar} does not fully discharge when the sense line goes low. Based on these results, it was determined that the pull-up design was less reliable And would take more time to develop. As such, the pull-down design was selected for layout development.

C. Simulation Results: AND/NAND Functionality

Next, the AND/NAND functionality of both the pull-up and pull-down designs was tested. Figure 8 shows the results of the pull-up design when RBL discharges by 200mV. Figure 9 shows the results of the pull-down design when RBL discharges by 200mV. (The reference voltage is set to VDD-150mV in both designs).

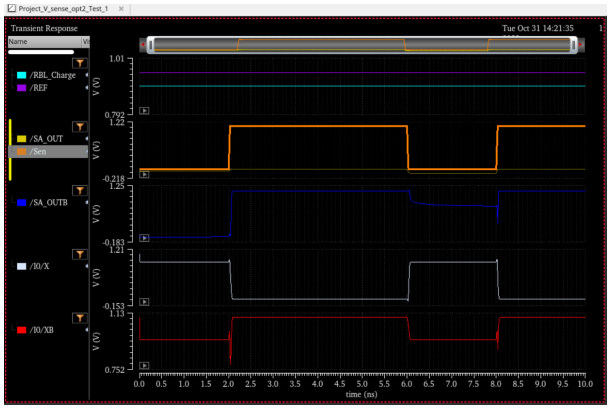

Fig. 8. Pull-Up Design: RBL Discharges 200mV

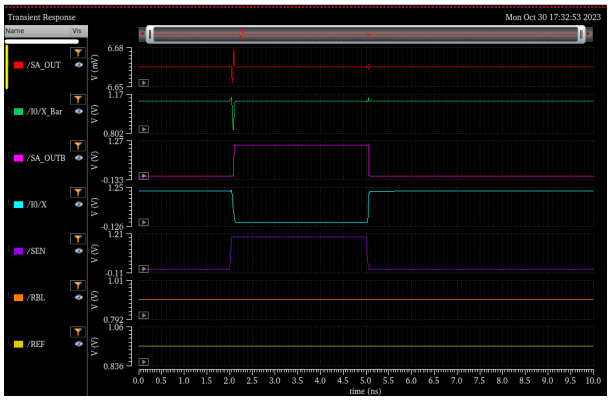

Fig. 9. Pull-Down Design: RBL Discharges 200mV

For the AND/NAND functionality, both circuits achieved the desired functionality. However, similarly to the results for the OR/NOR functionality, the pull-up design did not produce reliable results. Therefore, it was decided that the pull-down circuit was the best option for the final design.

D. Layout

The prototype layout for the sense amplifier is shown in Figure 10. The area for this layout was $5.106\mu m^2$. Because the total budget for peripheral circuits was 30% of the total layout, the sense amplifier's layout was selected for further optimization. The optimization was focused on maximizing pin sharing wherever possible while also taking into consideration the integration of the unit with the memory array. The second layout was designed with the experience gained from the second memory array design in mind; as such, metal four was reserved for connecting to the vertical RBL lines, while a higher metal was reserved for the horizontal reference voltage lines; with the access pins placed for ease of access. The final sense amplifier layout is shown in Figure 11, which has an area of $2.8\mu m^2$.

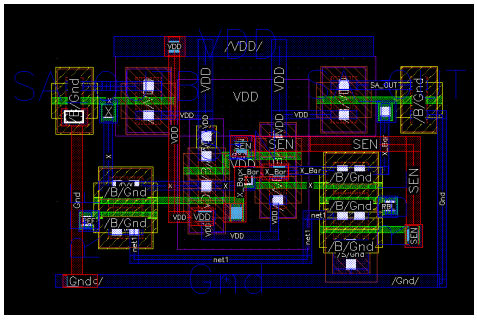

Fig. 10. Initial Sense Amplifier Layout

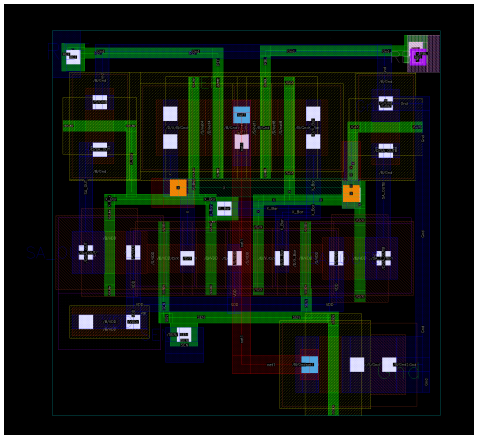

Fig. 11. Final Sense Amplifier Layout

IV. Logic Adder Derivation & Design

The sense amplifier produces the logical expressions AND/NAND and OR/NOR, as discussed in section 3. Because the sense amplifiers work as logic gates in and of themselves, it is both beneficial and essential to utilize them in developing the logic for the ripple adder array. This section articulates the derivation of the boolean expression needed to perform addition using the outputs of the sense amplifiers. First, consider the equations for the sum and carry of a full adder.

$$Sum = (AxorB)xorC \tag{1}$$

$$Carry = AB + BC + AC \tag{2}$$

Note that equations 1 and 2 can be written as:

$$Sum = (AB_{Bar} + A_{Bar}B)xorC \tag{3}$$

$$Carry = (AB) + (A + B)C \tag{4}$$

Since two reference voltages are needed to get the OR/NOR and AND/NAND functionality, the compute module will need to use two sense amplifiers. The first sense amplifier ($S1$) will have a reference voltage of 1.05V for the OR/NOR gates and the second sense amplifier ($S2$) will have a reference voltage of 950mV for the AND/NAND gates. Using the logical expression derived from the sense amplifiers we can write the Sum and Carry expressions as:

$$\begin{aligned} Sum &= (AB_{Bar} + A_{Bar}B)xorC \\ &= ((AB)_{Bar}(A + B))xorC \\ &= ((S2_{Out})(S1_{OutBar}))xorC \end{aligned} \tag{5}$$

$$\begin{aligned} Carry &= (AB) + (A + B)C \\ &= (S2_{OutBar}) + (S1_{OutBar})C \end{aligned} \tag{6}$$

Equations 4 and 5 are derived using equation 3 (for the Sum), equation 4 (for the Carry), and the outputs from the sense amplifier circuit in Figure 5. Depending on the reference voltage, the AND/OR logical expressions are the outputs for SA_{OutBar} and the NAND/NOR logical expressions are the outputs for SA_{Out} from the sense amplifier circuit in Figure 5. Based on equations 4 and 5, the circuit in Figure 12 was created to implement a full adder.

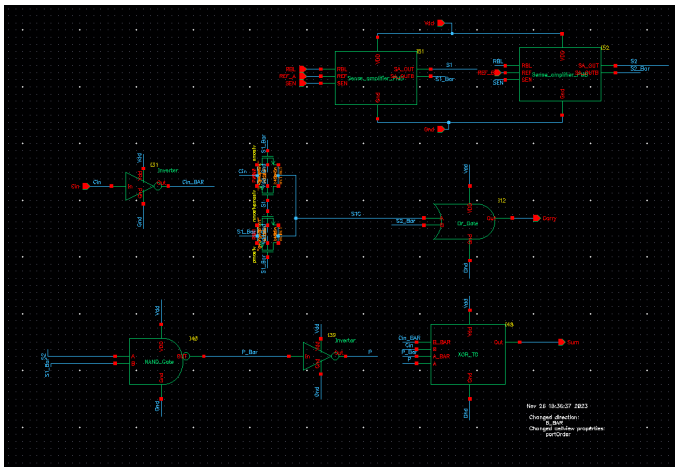

Fig. 12. Adder Module

A mixture of transmission gate logic, as well as CMOS logic, was used to create the circuit in Figure 12. The reason mixed logic was used was to leverage the benefits of both logic families. It was noted that using full transmission gate logic did acquire the results intended and minimized the number of MOSFETs; however, this logic produced longer rise and fall delays. Meanwhile, the complementary circuit configuration created more reliable results but at the cost of increasing area. That is why in the Carry circuit, transmission gate logic is used for the AND gate, and a complementary configuration (NOR gate with an inverter) is used for the OR gate. For the Sum circuit, transmission gate logic was used for the XOR gate. Figure 13 shows an example of the adder modules functionality when 1 and 1 are added. $S1_{Out}$ and $S2_{Out}$ goes low while $S1_{OutBar}$ and $S2_{OutBar}$ goes high, making the Sum go low and the Carry go high. The layout for this circuit is shown in Figure 14. The area for this layout was $13.505\mu m^2$. Unfortunately, due to time constraints, an optimized version was not created. That said, the final adder unit was well under the estimated area budget calculated in Section II B.

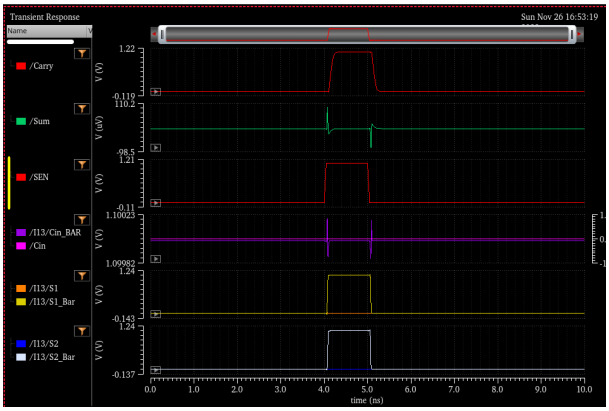

Fig. 13. Adder Module Functionality

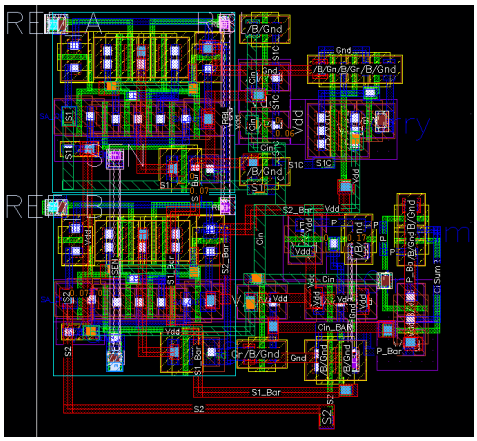

Fig. 14. Adder Module Layout

The next step was creating the full adder module for an 8-bit word.Figure 15 shows the full adder module's schematic and Figure 16 shows its layout. The final area for the 8-bit word adder module was $123.56\mu m^2$

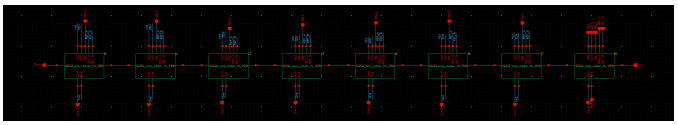

Fig. 15. Full Word Adder Schematic

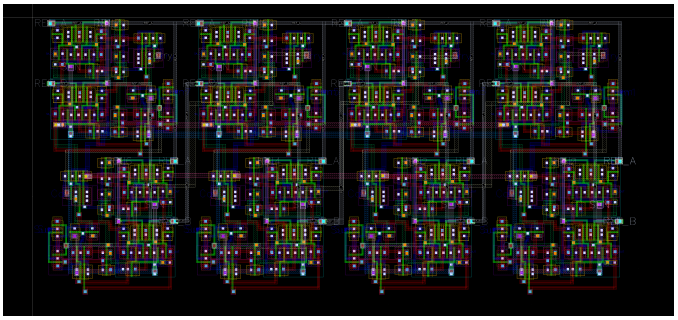

Fig. 16. Full Word Adder Layout

V. Final Design and Simulations

The final design of the full memory array, with built-in computing, consists of eight-word columns with 8-word adder units. All of the word columns are connected, sharing left and right contacts to reduce the area utilized by the memory array. It was decided that consistent wiring between each word column and their respective compute module would take priority over area efficiency. As such, all full adder units are placed precisely in the same location relative to their respective word columns. This increases the spacing of the computer modules, increasing the area used for in-memory computing. However, in exchange, all computations for any particular column occur nearly simultaneously, with minimal differences in propagation delay as the writing for each compute module is identical. In total, this design decision reduces the area efficiency by three percent, down from the 79% estimated in Section II B to 77% in exchange for consistent parallel computations. That said, there were some improvements to the implementation after the demo: the wiring distance between the adder units and the memory array was decreased, and it was noted that some of the side pins in the memory array were spaced out more than necessary. However, the propagation delays remained nominally unchanged. Figure 17 shows the final layout implemented and tested on the 8th of December 2023. Simulation results of the final design are shown in Figures 18 and 19. Figure 18 shows the addition of the values $9A_{hex}$ and $B3_{hex}$ where Sum0 is the least significant bit, Sum7 is the most significant bit, and Cout is the carryout. The final result is the value $14D_{hex}$ with a summation delay of 645ps. Figure 19 shows the addition of the worst case scenario, adding the value FF_{hex} and 1. As seen in the figure, all the sums are 0 when the sense line is asserted, and the carryout becomes one. Producing the expected result of 100_{hex}.

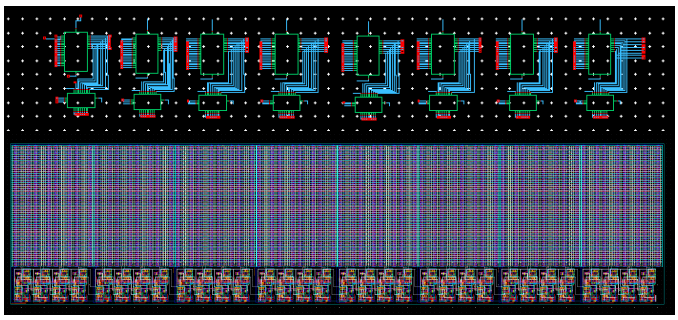

Fig. 17. Top: The Full Project Schematic. Bottom: The Final Project Layout of the Memory Array With The Parallel Adder Unit.

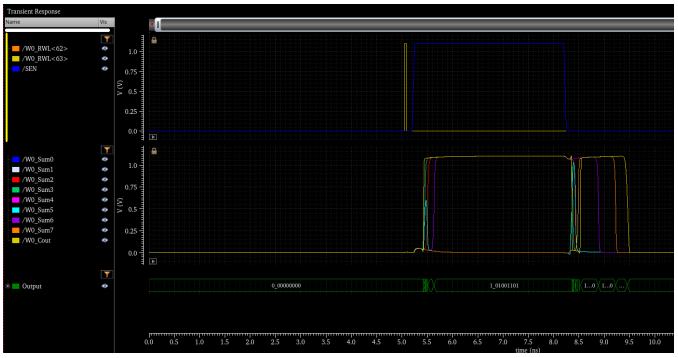

Fig. 18. Addition Results of $9A_{hex}$ and $B3_{hex}$

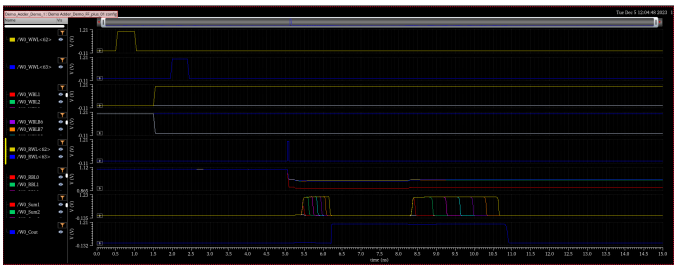

Fig. 19. Addition Results of $9A_{hex}$ and 1

VI. Energy-Latency-Area Analysis

A. Energy Analysis

The total energy consumed for each process is listed in table III. The energy analysis was performed by taking the integral of the supply voltage with the current for the time in which the current spiked and then settled in each of the functions. The energy analysis for the write function was taken from when the WWL was asserted to when the current settled. The total write energy consumed was 69.8fJ. The energy for the read function appears negligible or immeasurable because there is a significant amount of capacitance coupling present in the design. It appears that a considerable amount of energy is stored in the circuit before reading the bit cells, resulting in unexpectedly high negative currents. A full and thorough energy analysis is outside the scope of this project due to time constraints. The energy measurement for the compute process was taken from when the RWL was asserted to when the current settled. The total energy from this process would differ depending on what data was stored in the rows, but for the worst energy consumption (Adding $FF_{hex} + FF_{hex}$), the total energy consumed was 1.87pJ.

Process	Energy(J)
Write	69.8f
Read	Negligible
Summation	1.87p

TABLE III
Energy Analysis

B. Latency Analysis

The latency for the write, read, and summation functions was recorded for the worst-case scenario of adding the hexadecimal value FF_{hex} with the value 01_{hex}. Table IV shows the process latency with the parasitic capacitance extracted from the layout.

Process	Delay(s)
Write	38.43p
Read	180p
Summation	1.174n

TABLE IV
Worst Case Latency Analysis

The write latency is the time it takes for a bit to be stored in the SRAM cell. The write latency was measured from when the WWL line was asserted to when the values were stored in Q and Q_{Bar} of the SRAM cell, which was measured to be 38.43ps. The read latency is the time it takes for the bit to be read from the cell, or it could be defined as the time it takes for RBL to discharge to the proper value. The read latency was measured from when the RWL line was asserted to when the RBL line was discharged to the correct value, which was measured to be 180ps. Finally, the compute latency is the time it takes for the full word adder to stabilize at the correct summation. The compute latency was measured from when the RWL was asserted to when the final sum was stabilized at the output. Depending on the addition of two words, the compute latency will vary. However, the worst-case compute latency was 1.174ns. For the worst-case scenario, the delay from when the sense line was asserted to when the final sum was stabilized was 994.1ps.

C. Area Analysis

For any design, one of the most important aspects is the area taken up by the layout. To optimize the layout, pin sharing was used to minimize the total area. However, due to time constraints, not every design aspect could be optimized. The major concern for this design was to optimize the SRAM array. The adder module has room for development. Table V shows the length, height, and total area used for each major component in this design. The single SRAM cell had a length of $2.39\mu m$ and a height of $0.725\ \mu m$ making the total area to be $1.732\mu m^2$. The 8-bit word had a length of $18.1\mu m$ and a height of $0.725\mu m$ making the total area to be $13.122\mu m^2$. The SRAM column, which includes 64 rows of an 8-bit word, had a length of $18.1\mu m$ and a height of $26.9\mu m$ making the total area used $486.89\mu m^2$. The full SRAM Array's length measured $143.33\mu m$ and is height measured $27.11\mu m$ making its total area $3885.68\mu m^2$. The total area was reduced for the SRAM column, as well as the full SRAM array, by sharing the WWL and RWL pins for the surrounding words. The adder module's length was measured at $3.65\mu m$, and its height was measured at $3.7\mu m$ for the total area used being $13.505\mu m^2$. The full word adder module, which consists of the adder circuit for each bit in a word, length was measured at $16.11\mu m$, and its height was measured at $7.67\mu m$ for the

total area being $123.56\mu m^2$. Area was conceded for the full word adder module due to spacing and wiring considerations. The final layout ended up having a length of $143.55\mu m$ and a height of $35.09\mu m$ for the total area being $5037.32\mu m^2$

Layout	Length(μm)	Height(μm)	Area(μm^2)
SRAM Cell	2.39	0.725	1.732
SRAM Word	18.1	0.725	13.122
SRAM Column	18.1	26.9	486.9
SRAM Array	143.33	27.11	3885.68
Adder Module	3.65	3.7	13.505
Full Word Adder Module	16.11	7.67	123.56
Final Layout	143.55	35.09	5037.32

TABLE V
AREA ANALYSIS

VII. PARALLEL COMPUTING TEST RESULTS

The benefit of computing in memory is the parallelism that is offered by this architecture. Simultaneously adding words will further decrease computing latency. Figures 20 and 21 show how this process is possible by adding two 8-word rows together that contain the hexadecimal value FF. In Figure 20, the WWL is asserted at 1.5ns, and the proper bits are stored in each SRAM cell. Once the RWL is asserted at 5ns the correct bits are read from the SRAM cells. Then the sense line is asserted and starts the arithmetic process. As seen in Figures 20 and 21 the sum for each word is the same with the same delay of 414ps.

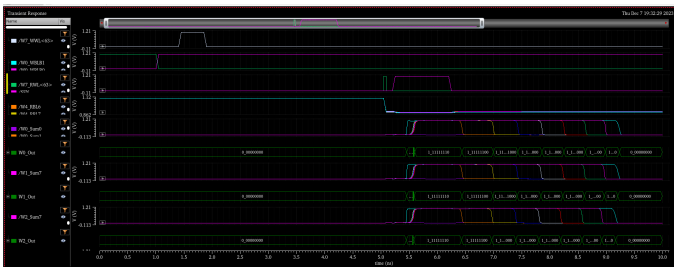

Fig. 20. 8 Word Addition Results

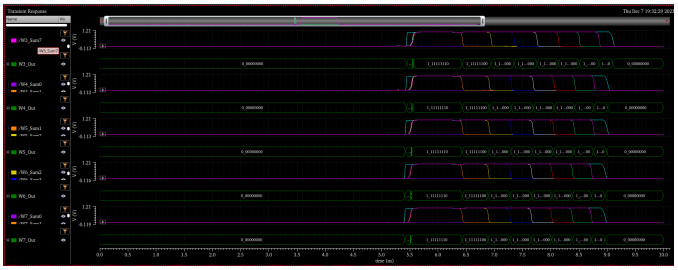

Fig. 21. 8 Word Addition Results

VIII. CONCLUSION

Using in-memory computing is a path to mitigating the bottleneck caused by the Von Neumann architecture. This paper has presented a 64 by 64-bit memory array with built-in parallel computing capable of full 8-parallel Word additions. During testing, this paper has shown that the worst-case propagation delay for any addition was 1.174ns, with a memory access delay Of 180ps And a memory write delay of 38.43ps. The layout implementation achieved a memory efficiency of 77%. In-memory computing shows the promise of improving the power and latency of computationally intensive tasks such as AI neural network processing. This implementation of in-memory computing hardware demonstrates the potential for optimizing integrated circuits and their power to calculate linear operations at scale. It is quite possible to scale the design presented herein by several orders of magnitude without significantly increasing the computational time required for adding rows of words together. Based on this design, it would be possible to implement more nuanced logical functions into the memory array, further reducing the CPU bottleneck for specific computational tasks.

REFERENCES

[1] A. Jaiswal, I. Chakraborty, A. Agrawal, and K. Roy, "8t SRAM cell as a multibit dot-product engine for beyond von neumann computing," vol. 27, no. 11, pp. 2556–2567, conference Name: IEEE Transactions on Very Large Scale Integration (VLSI) Systems. [Online]. Available: https://ieeexplore.ieee.org/document/8802267/authorsauthors