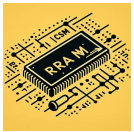
भारतीय प्रौद्योगिकी संस्थान तिरुपति

TIRUPATI

In-Memory Computing using ReRAM Devices Applied to 2-D Transforms



Guide
Dr. Vikram Kumar Pudi

Presented By:
Ayush Yadav (EE23M114)

Department of Electrical Engineering, IIT Tirupati

Contents

Introduction(Why In-Memory Computing?)

- The exponential growth in data processing causes Traditional architectures to suffer from the **"Memory Wall Problem"**, where data transfer dominates latency and power consumption.
- In-memory computing offers a paradigm shift by performing computations directly within memory, eliminating data movement.
- ReRAM (Resistive Random Access Memory) is a promising technology for efficient in-memory computing due to its non-volatility, high speed, and scalability..
- Applications: AI accelerators, real-time signal processing, and Power-efficient edge devices.

Objectives and Scope

Objectives:

- Explore ReRAM devices for implementing 2-D transforms like the Hadamard Transform.
- Design an automated Binary Addition mechanism using ReRAM cells to perform in-memory computations with a single start input.
- Automate various computations using a single-input mechanism.

Scope:

- Develop a scalable memory architecture that stores configuration codes, enabling the selective activation of Word-Lines for various computational tasks, optimizing system flexibility.
- Integrate circuits to execute automated in-memory computations, reducing hardware complexity while ensuring high-speed performance.
- Focus on minimizing power consumption and maximizing computation speed.

Literature Survey

- **Binary Addition Using Majority Logic in ReRAM Arrays**:
  - In- Memory binary addition directly within a ReRAM array using majority logic.
  - This in-memory computation technique reduces energy consumption and enhances performance by eliminating the need for data transfers between the processor and memory.[1]
- **Accelerated Addition in ReRAM Arrays Using Parallel-Friendly Majority Gates**:
  - Focuses on improving the addition operation in ReRAM arrays by using parallel-friendly majority gates.
  - This strategy reduces addition latency and leverages ReRAM's intrinsic properties for logic operations, improving computational efficiency.[2]

References:

[1]John Reuben, *Micromachines*, 2020, 11(5), 496. DOI: https://doi.org/10.3390/mi11050496

[2]John Reuben Prabahar and Stefan Pechmann, *IEEE Transactions on VLSI Systems*, 2021. DOI: https://dx.doi.org/10.1109/TVLSI.2021.3068470

- **Majority Logic-Based In-Memory Comparator Design**:
  - Uses ReRAM's majority logic capabilities to implement an in-memory comparator.
  - This design simplifies the comparator architecture, reduces complexity, and improves energy efficiency, making it suitable for binary operations like addition and comparison.[3]
- **Time-Based Sensing for Robust Read in STT-MRAM (Potential ReRAM Application)**:
  - Although focused on STT-MRAM, this research on time-based sensing may also be applicable to ReRAM.
  - The method improves read operation reliability and could be adapted to enhance the sensing mechanism in ReRAM arrays, particularly in binary addition scenarios.[4]

References:

[3] V. Lakshmi, J. Reuben, and V. Pudi, IEEE Transactions on Circuits and Systems I: Regular Papers, 2022, 69(3), 1148–1158. DOI: https://doi.org/10.1109/TCSI.2021.3129827

[4] J. R. Reuben, D. Fey, and C. Wenger, IEEE Transactions on Nanotechnology, 2019, 18, 647–656. DOI: https://doi.org/10.1109/TNANO.2019.2922838

Problem Statement

- **Recent advancements in automating complex computations:**
  - **Target:** To make automatic circuits for performing different computations like Binary Addition.
- **Existing 2-D transforms rely on power-intensive CPU-GPU systems.**
  - **Target:** To use automated computations for performing 2-D transforms.
- **ReRAM offers a low-power alternative, but challenges exist:**
  - Device non-idealities affecting accuracy and reliability.
  - Scalability for large-scale transformations in real-time applications.
  - Efficient integration of sense amplifiers for precise read and write operations.

Methodology(Binary Addition using ReRAM)

Design:

- Using Stanford PKU model of RRAM
- Testing of Writing and Reading data from ReRAM cells.
- Testing of Majoritry Logic using Sense Amplifier from ReRAM cells.
- Testing of Binary Addition using Sense Amplifier from ReRAM cells.
- Testing of Binary Addition Automated Circuit using Sense Amplifier from ReRAM cells.
- Designed using Cadence Virtuoso.

Simulation:

- Evaluated performance using Cadence Spectre simulations.

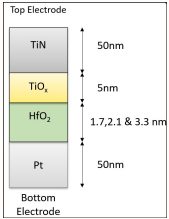
Automatic Computation using ReRAM Cells(Novelty)

- First attempt to implement the automated operation of Binary Addition using ReRAM arrays.
- Combines computational and memory functionalities into a single compact unit.
- Demonstrates automatic computation within the ReRAM array.
- It will be utilized for intermediate values in 2-D transforms like DCTs in the future.
- **Advantages:**
  - Reduced time and efforts.
  - High throughput for parallel processing.

Stanford-PKU (Peking University) RRAM Model

We used a Stanford PKU model of RRAM.

- The Stanford-PKU RRAM Model is a SPICE-compatible compact model that describes switching performance for bipolar metal oxide RRAM.
- In principle, this model has no limitations on the size of the RRAM cell.
- The complex process of ion and vacancy migration was simplified into the growth of a single dominant filament that preserved the essential switching physics.
- The size of the tunneling gap (g), which is the distance between the tip of the filament and the opposite electrode, is the primary variable determining device resistance.



Top Electrode Layers:

- TiN (**50nm**)
- TiO_x (**5nm**)
- HfO_2 (**1.7, 2.1, and 3.3nm**)
- Pt (**50nm**)

Not to Scale

1T1R Cell

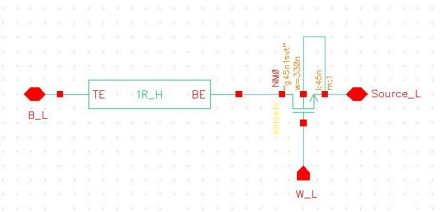


Figure: 1T1R Cell

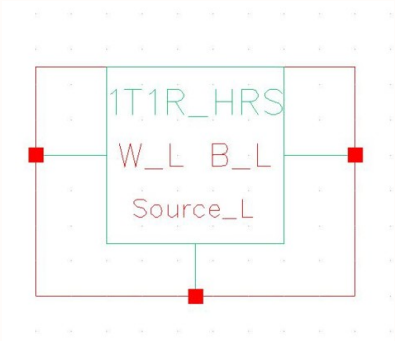


Figure: Symbol

Summarized Phenomenon: Bipolar Switching in ReRAM

RESET Process:

- Disrupts the conductive filament.
- Oxygen vacancies move away from the filament area.
- Results in a **high resistance state (HRS)**.

SET Process:

- Reforms the conductive filament.
- Oxygen vacancies and electrons are driven back into the filament region.
- Results in a **low resistance state (LRS)**.

Key Insights:

- Switching between HRS and LRS is controlled by the movement of oxygen vacancies and filament formation.
- Typical in oxide-based ReRAM (e.g., HfO_2).
- Data storage achieved by manipulating the resistance state of the memory cell.

Set Operation

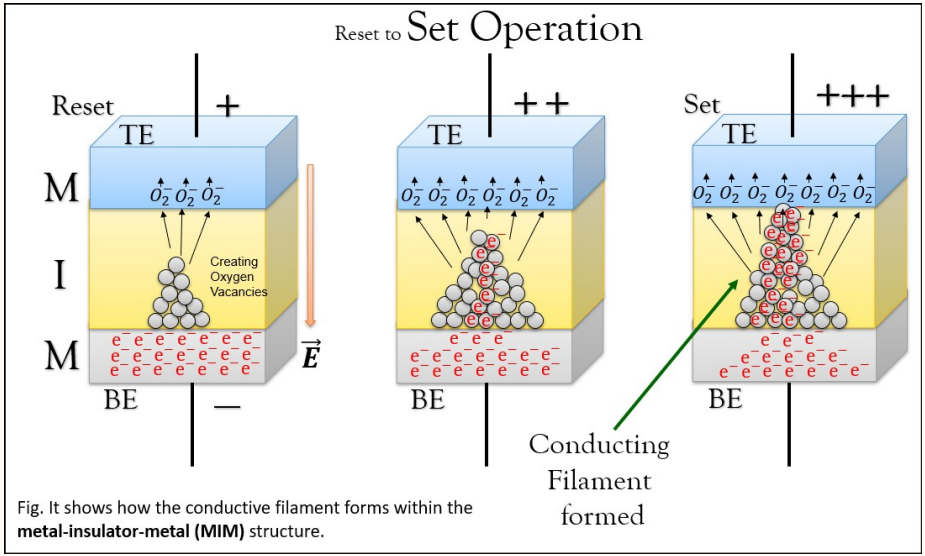


Fig. It shows how the conductive filament forms within the **metal-insulator-metal (MIM)** structure.

ReSet Operation

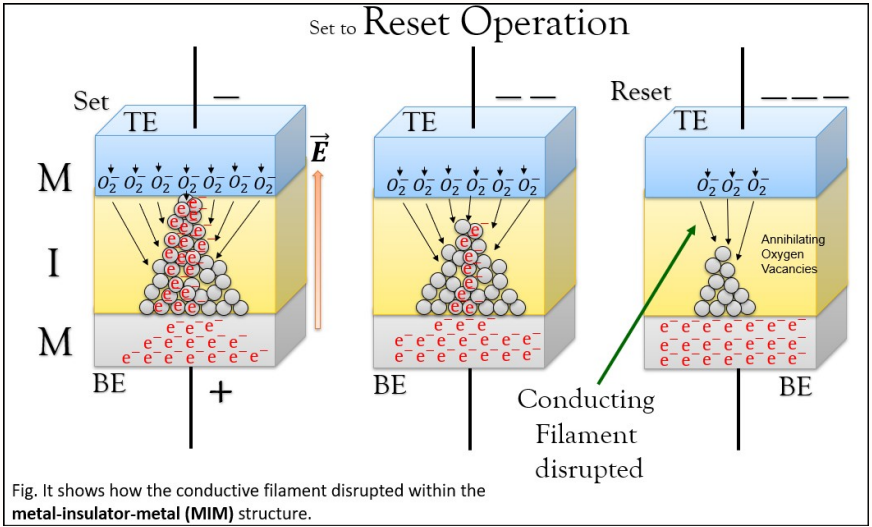


Figure: ReSet Operation

ReRAM Operation Table for Write-1

Time (μs)	Word Line Voltage (V_{wl})	Bit Line Voltage (V_{bl})	Source Line	Operation	Current (μA)	Resistance State / Value ($k\Omega$)
0 – 2	1 V	200 mV	GND	Initial Read	$\sim$1.5	$\sim$133.3 (HRS)
Description: Safe sensing of the ReRAM's resistance without altering its state.						
2 – 12	1.4 V	1 – 1.3 V	GND	Set Operation	$\sim$250	LRS
Description: Applying 1 V to the bit line initiates the set process, forming a conductive filament (LRS).						
12 – 14	1 V	200 mV	GND	Final Read	$\sim$20.5	$\sim$9.76 (LRS)
Description: Confirms that the set operation took place in the ReRAM cell without disturbing the state.						

ReRAM Operation Table for Write-0

Time (μs)	Word Line Voltage (V_{wl})	Bit Line Voltage (V_{bl})	Source Line	Operation	Current (μA)	Resistance State / Value ($k\Omega$)
0 – 2	1 V	200 mV	GND	Initial Read	$\sim$20.5	$\sim$9.76 (LRS)
Description: Safe sensing of the ReRAM's resistance without altering its state.						
2 – 12	1.4 V	$[-1.3, -1.8]$ V	GND	**Reset Operation**	$\sim$250	HRS
Description: Applying -1.8 V to the bit line initiates the reset process, disrupting the conductive filament (HRS).						
12 – 14	1 V	200 mV	GND	Final Read	$\sim$1.5	$\sim$133.3 (HRS)
Description: Confirms that the Reset operation took place in the ReRAM cell without disturbing the state.						

Overall Table for Different Operations

Word Line Voltage (V_{wl})	**Bit Line Voltage (V_{bl})**	**Source Line**	**Operation**	**Description**	**Resistance State**
1 V	200 mV	GND	Read Operation	Bit-line at 200mV and word-line at 1V allows for safe sensing of RRAM's resistance without changing its state.	**Unknown** (Assumed LRS if no reset operation has occurred)
1.4 V	-1.8 V	GND	**Reset Operation**	Wordline at 1.4V and bit-line at -1.8V create a large potential difference, breaking the conductive filament and switching RRAM to high-resistance state (HRS).	**HRS**
1 V	200 mV	GND	Read Operation	Again, set to 1V and 200mV, this read operation measures the RRAM resistance to confirm it is in HRS without disturbing the cell's state.	**HRS**
1.4 V	1–1.2 V	GND	**Set Operation**	Bit-line at 1.2V and word-line at 1.4V initiate the Set operation, forming a conductive filament and switching RRAM to LRS.	**LRS**
1 V	200 mV	GND	Read Operation	Again, set to 1V and 200mV, this read operation measures the RRAM resistance to confirm it is in LRS without disturbing the cell's state.	**LRS**

Testing of Majority Logic using Sense Amplifier from ReRAM cells.

A	B	C	M_3(A, B, C)	V_{bl} (mV)	Digital Logic
0	0	0	0 (means LRS here)	86.2261	1
0	0	1	0	125.5681	1
0	1	0	0	125.5681	1
0	1	1	1	228.212	0
1	0	0	0	125.5681	1
1	0	1	1	228.212	0
1	1	0	1	228.212	0
1	1	1	1 (means HRS here)	648.261	0

Table: Truth Table for a 3-Input Majority Gate

Explanation:

- The output is 1 if the majority (two or more) of the inputs are 1.
- The output is 0 if the majority of the inputs are 0.

Binary Addition Flowchart and Details

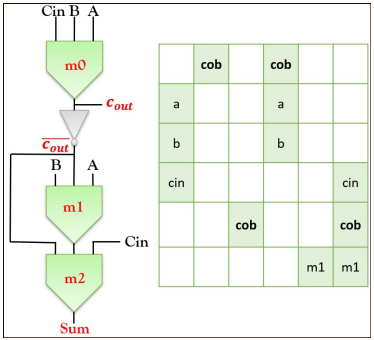


Figure: Flow Chart for Binary Addition

Details of the Operation:

- **Cycle 1:** Read `a`, `b`, and `cin`.
- **Cycle 2:** Write to `cob`.
- **Cycle 3:** Write to `cob` again.
- **Cycle 4:** Read `a`, `b`, and `cob` to compute `m1`.
- **Cycle 5:** Write to `m1`.
- **Cycle 6:** Read `cin`, `cob`, and `m1` to compute the sum.

References:

[1]John Reuben, *Micromachines*, 2020, 11(5), 496. DOI: https://doi.org/10.3390/mi11050496

Testing of Binary Arithmetic using Sense Amplifier from ReRAM cells.

1. **Step-1:** Making $A = \text{HRS}, B = \text{HRS}, C_{\text{IN}} = \text{HRS}$.
2. **Step-2:** Reading $A = \text{HRS}, B = \text{HRS}, C_{\text{IN}} = \text{HRS}$.
3. **Step-3:** Making/Reading $\overline{C_{\text{OUT}}}(1)$ and $\overline{C_{\text{OUT}}}(5)$ as LRS.
4. **Step-4:** Reading $A = \text{HRS}, B = \text{HRS}, \overline{C_{\text{OUT}}} = \text{LRS}$.
5. **Step-5:** Making $m_1(6) = \text{HRS}$.
6. **Step-6:** Reading $\overline{C_{\text{OUT}}} = \text{LRS}, m_1 = \text{HRS}, C_{\text{IN}} = \text{HRS}$.

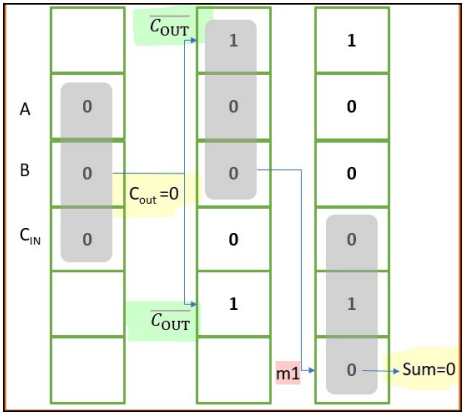


Figure: Testing of Binary Arithmetic

Circuit Overview

- Detailed circuit design for automated ReRAM operations.
- Integration of control logic for turning ON the Word-Lines.
- Integration of control logic for write/read cycles.
- Doing all the cycles with the help of counter where Counts = Cycles.

Flow-Chart

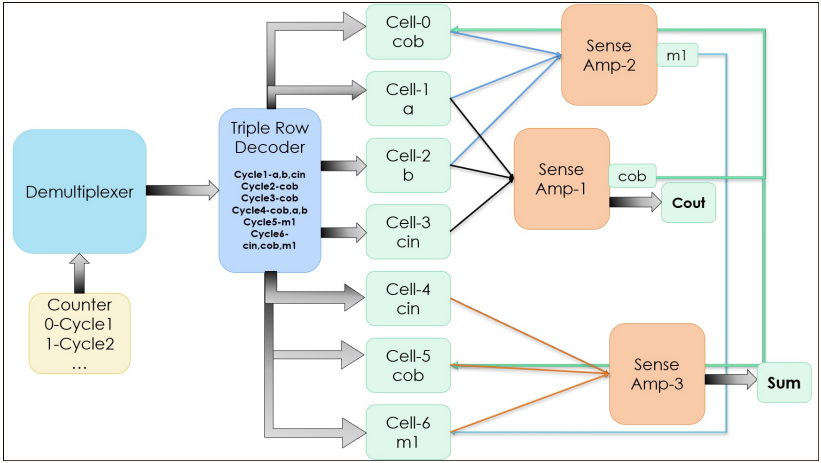


Figure: Flow-Chart for the Circuit

Cycle-by-Cycle Operations

Inputs are as follows: a = 0, b = 0, c = 0.

Cycle	Counter Output	DEMUX Output	MAJ	Triple Row Decoder Output	Operation	Output of Action
1	000	I_0	1	0001	Compute $cout, cob$ (Majority of a, b, cin)	$cout = 0$, $cob = 1$
2	001	I_1	0	0000	Write cob to Cell-0	$cob = 1$ stored
3	010	I_2	0	0010	Write cob to Cell-5	$cob = 1$ stored
4	011	I_3	1	0101	Compute $m1$ (Majority of a, b, cob)	$m1 = 0$
5	100	I_4	0	0100	Write $m1$ to Cell-6	$m1 = 0$ stored
6	101	I_5	1	0111	Compute Sum (Majority of $cin, cob, m1$)	Sum=0

Counter and Demultiplexer

Counter:

- Determines the specific operation to perform in the circuit.
- Outputs are passed to the demultiplexer to activate one output line at a time.
- Activated demultiplexer output is connected to the triple row decoder.
- Triple row decoder decodes the input to activate the required rows in the memory array.

Demultiplexer:

- Receives input from the counter and maps it to one of its output lines (e.g., Cycle 1 to Cycle 6).
- Each demultiplexer output corresponds to a specific operation:
  - **Cycle 1:** Read rows for a, b, and cin.
  - **Cycle 2:** Write cob to Cell-0.
  - **Cycle 3:** Write cob to Cell-5.
  - **Cycle 4:** Compute $m1$ by reading rows for a, b, and cob.
  - **Cycle 5:** Write $m1$ to Cell-6.
  - **Cycle 6:** Compute the sum by reading rows for cin, cob, and $m1$.

Triple Row Decoder and ReRAM Cells

Triple Row Decoder:

- Receives demultiplexer output and controls word-lines for ReRAM cells.
- Supports:
  - Single-row WRITE operations for storing *cob* and $m1$.
  - Multi-row READ operations for computing majority results.
- Example Operations:
  - **Cycle 1:** Activates rows for *a*, *b*, and *cin* (Cells 1, 2, 3).
  - **Cycle 2:** Activates row for Cell-0 to write *cob*.

ReRAM Cells:

- Act as storage and computational elements:
  - **Cell-0:** Stores *cob*.
  - **Cell-1:** Stores *a*.
  - **Cell-2:** Stores *b*.
  - **Cell-3:** Stores *cin*.
  - **Cell-4:** Stores another *cin* for computations.
  - **Cell-5:** Stores *cob*.
  - **Cell-6:** Stores $m1$, the intermediate majority result.
- Enable simultaneous computations and storage for energy efficiency.

Sense Amplifiers

- The selected rows interface with sense amplifiers, which read or compute the data stored in ReRAM cells.
- Outputs from the sense amplifiers represent computed values (e.g., carry-out, majority, sum) or signals to be written back to the memory cells.
- The entire operation is synchronized with the clock to ensure seamless automation across cycles.
- Sense amplifiers perform READ operations to compute outputs:
  - Sense Amp-1: Computes co and cob (from a, b, cin).
  - Sense Amp-2: Computes $m1$ (from a, b, cob).
  - Sense Amp-3: Computes the sum (from cin, cob, $m1$).
- Amplifiers ensure accurate read operations by detecting ReRAM resistance levels.

Operation Summary (Cycle by Cycle)

- **Cycle 1:** Compute co and cob (majority of a, b, cin).
- **Cycle 2:** Write cob to Cell-0.
- **Cycle 3:** Write cob to Cell-5.
- **Cycle 4:** Compute $m1$ (majority of a, b, cob).
- **Cycle 5:** Write $m1$ to Cell-6.
- **Cycle 6:** Compute sum (majority of cin, cob, $m1$).

Circuit

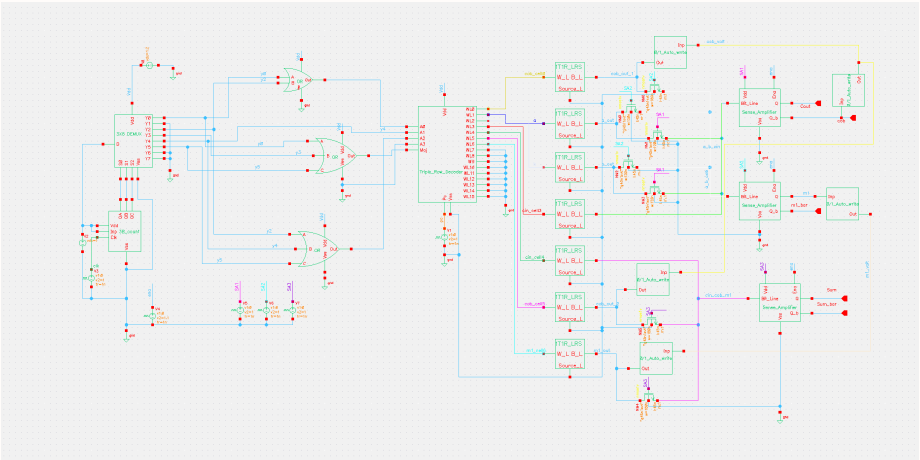


Figure: Example Circuit Overview

Results

- **Test 0:** Testing of correctly turning ON the respective word lines.
- **Test 1:** Testing of correctly sensing a,b,cin, and later sensing cob,a,b.
- **Test 2:** Testing of correctly sensing a,b,cin, and writing 1.3V in cob.
- **Test 3:** Testing of sensing a,b,cin, writing 1.3V in cob and again sensing cob,a,b.
- **Test 4:** Testing of writing 1.3V in cob with 2nd Clock Pulse.
- **Test 5:** Testing of writing 1.3V in cob with 2nd Clock Pulse with a delay element.

Conclusion

- **Conclusion 1:** Automated full adder design using ReRAM cells is feasible and effective for in-memory computation.
- **Conclusion 2:** Synchronization between WRITE and READ operations is critical:
  - Introduce delay elements to separate Sense Amplifier READ and WRITE operations.
  - Ensure precise clock pulse management for stable operation.
- **Conclusion 3:** Counter-driven automation reduces the need for manual cycle management:
  - Each cycle automatically controls corresponding word-lines and operations.
- **Conclusion 4:** Improvements needed for real-time scalability:
  - Address timing mismatches and voltage control for large-scale circuits.

Future Work

- **Scalability:**
  - Extend the automated full adder design to multi-bit binary addition.
  - Design larger automated arithmetic units like multipliers and accumulators using ReRAM arrays.
- **Integration:**
  - Develop hybrid architectures combining ReRAM with other emerging memory technologies.
  - Explore integration with machine learning accelerators for in-memory AI computations.
- **Applications:**
  - Implement advanced digital signal processing tasks like Fourier Transforms and convolution after completing Hadamard Transform.
  - Explore real-world use cases such as cryptography, image processing, and neural network inference.

References

1. John Reuben, *Micromachines*, 2020, 11(5), 496. DOI: https://doi.org/10.3390/mi11050496
2. John Reuben Prabahar and Stefan Pechmann, *IEEE Transactions on VLSI Systems*, 2021. DOI: https://dx.doi.org/10.1109/TVLSI.2021.3068470
3. V. Lakshmi, J. Reuben, and V. Pudi, IEEE Transactions on Circuits and Systems I: Regular Papers, 2022, 69(3), 1148–1158. DOI: https://doi.org/10.1109/TCSI.2021.3129827
4. J. R. Reuben, D. Fey, and C. Wenger, IEEE Transactions on Nanotechnology, 2019, 18, 647–656. DOI: https://doi.org/10.1109/TNANO.2019.2922838

Thank You!

Questions?

Problems

- **Problem 1:** Synchronization issues in *cob* writing:
  - Writing 1.3V/-1.3V to *cob* overlaps with Sense Amplifier READ operations in cycle 1.
  - Output sensed incorrectly due to timing mismatches. I need this writing voltage at bit-line of *cob* in cycle 2.
- **Problem 2:** Lack of proper delay in circuit operations:
  - Bit-line voltage for *cob* reaches prematurely in cycle 1 only, causing data corruption in intermediate steps.
  - I need to send the output of sense amplifier *cob* in cycle 2 and not in cycle 1 with the help of proper delay at output of sense amplifier.

Hadamard Transform

- Introduction Understanding the Basics and Applications : The Hadamard Transform is a mathematical operation used in various fields like signal processing, image compression, and quantum computing. It transforms a set of values into a new set of values using orthogonal basis vectors, simplifying many computational problems.
- Hadamard Matrix The Hadamard matrix is an $N \times N$ matrix, where N is a power of 2. It contains only $+1$ and -1 entries and is defined recursively:

$$H_1 = [1]$$

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

- Performing the Hadamard Transform The Hadamard Transform of a vector x is given by multiplying it by the Hadamard matrix H:

$$y = H_N \times x$$

  This operation converts the input vector into a new vector, which is a combination of the Hadamard basis vectors.
- Applications The Hadamard Transform has a wide range of applications, including: Signal Processing: Image compression, filtering, and analysis. Quantum Computing: Creating superposition states and implementing quantum algorithms. Error Detection and Correction: Constructing error-correcting codes.

Testing of Writing 1 to ReRAM cell.

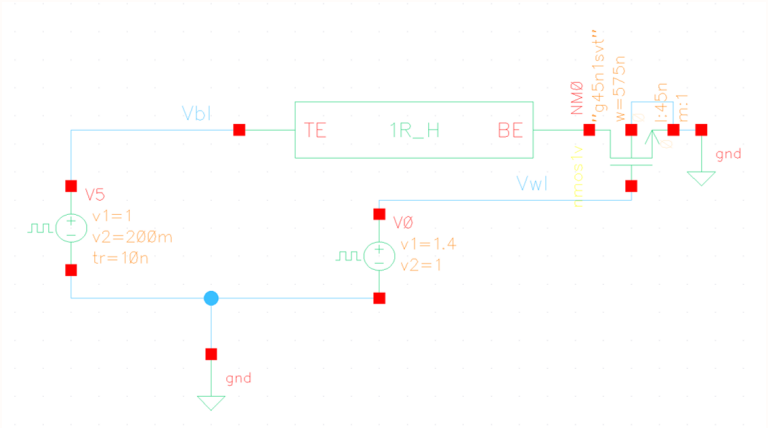


Figure: Write 1

Testing of Writing 0 to ReRAM cell.

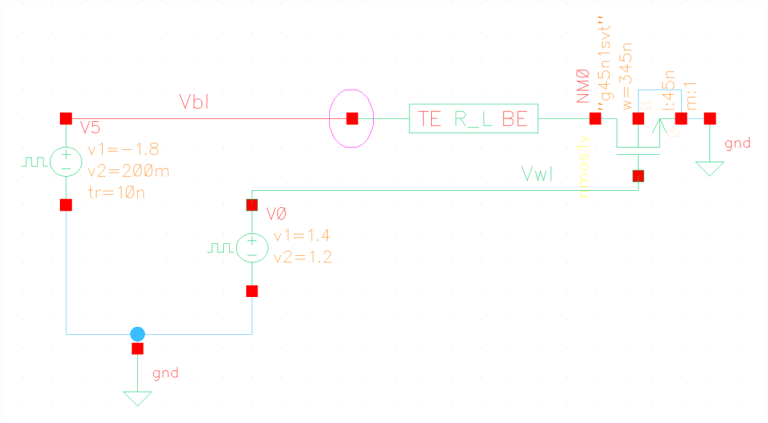


Figure: Write 0

Sense Amplifier Schematic

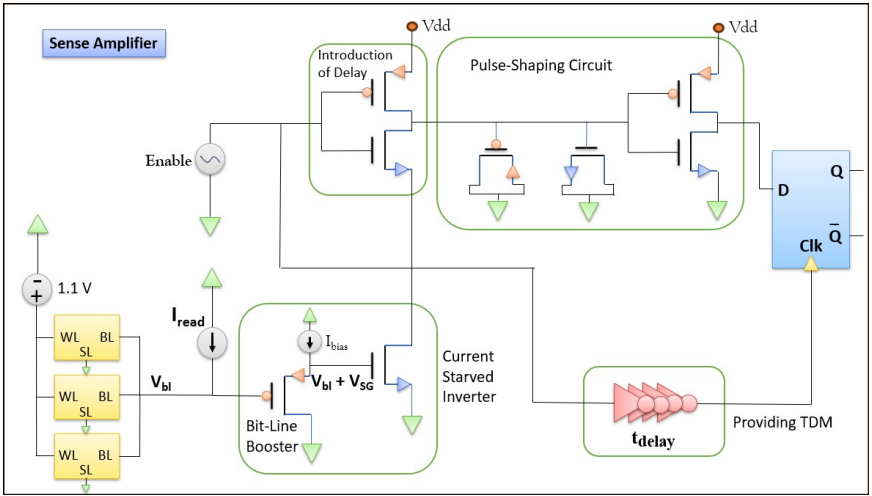


Figure: Sense Amplifier Schematic Diagram

Components of Sense Amplifier

- Common Drain MOS (BL Booster) The PMOS transistor (M1) boosts the bit-line voltage for time-based comparison. It operates as a voltage follower, buffering the bit-line voltage without significant loading. The output of M1 is the sum of the bit-line voltage (V_{BL}) and the source-gate voltage (V_{SG}).
- Current Starved Inverters The current-starving transistor (M2) controls the current through the M3-M4 inverter stage. The gate of M2 is controlled by the boosted bit-line voltage ($V_{BL} + V_{SG}$).
- MOS Capacitors (M5, M6) MOS capacitors exploit the inherent gate and drain-source capacitance of MOSFETs. These capacitors add load capacitance to shape the timing characteristics of the circuit and smooth signal transitions.

- Pulse-Shaping Inverter Stage (M7, M8) The pulse-shaping inverter stage sharpens and cleans up the signal before it reaches the D-flip flop. It also reduces noise and distortion from earlier stages, ensuring a clean output.
- Chain of Inverters The chain of inverters amplifies and stabilizes signals to ensure the reliable operation of the circuit.
- D Flip Flop (D-FF) The D-flip flop captures the state of the signal at the critical Time Decision Moment (TDM). It ensures accurate memory state detection based on the timing of the input signal.
- Programmable Delay Line (PDL) The programmable delay line adjusts the signal propagation time for precise operation. It compensates for variations in bit-line voltage and resistances, ensuring proper timing.
- Time-Domain Comparator (D-FF) The time-domain comparator samples the input signal at TDM for decision-making. It outputs a low signal for low-resistance states (LRS) and a high signal for high-resistance states (HRS).

Key Concepts in Time-Based Sensing

- Time Decision Moment (TDM) The TDM refers to a specific point in time for decision-making in time-based sensing circuits.
- Voltage-to-Time Conversion Current-starved inverters are used to convert voltage differences into timing signals.
- Decision Process For low-resistance states (LRS), higher current results in shorter delays and earlier TDMs. For high-resistance states
- Timing Characteristics The Time Decision Moment (TDM) typically occurs between $4.8k\Omega$ and $8.7k\Omega$, ensuring accurate resistance state sensing.

Sizing of the Transistors

- Role of W/L Ratios in CMOS Design The W/L ratio affects a transistor's electrical characteristics. A larger W/L ratio improves current-carrying capacity and switching speed, but may lead to higher power consumption. It also enhances noise immunity.

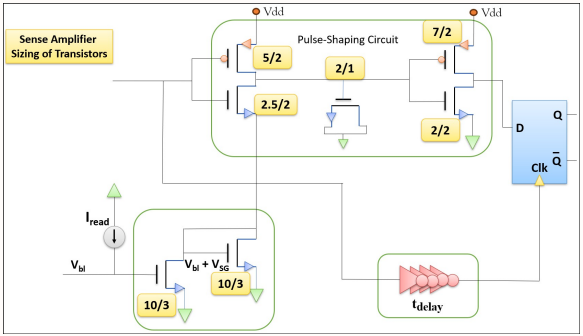


Figure: Sizing of the Transistors

Circuit Diagram for Test 0

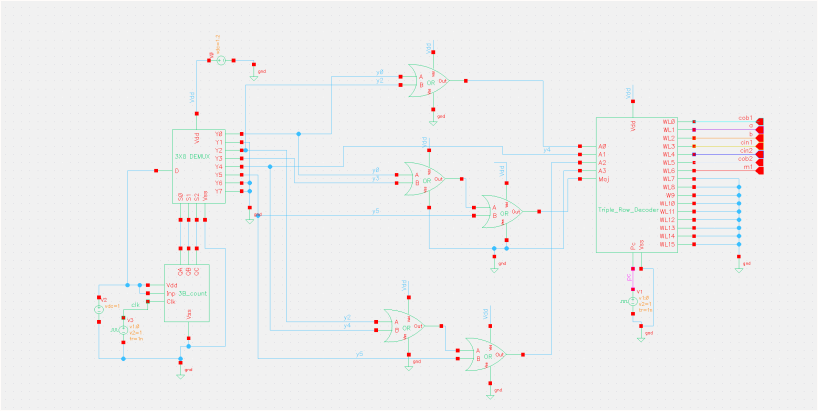


Figure: Circuit for Test 0: Testing of correctly turning ON the respective word lines.

Waveform for Test 0

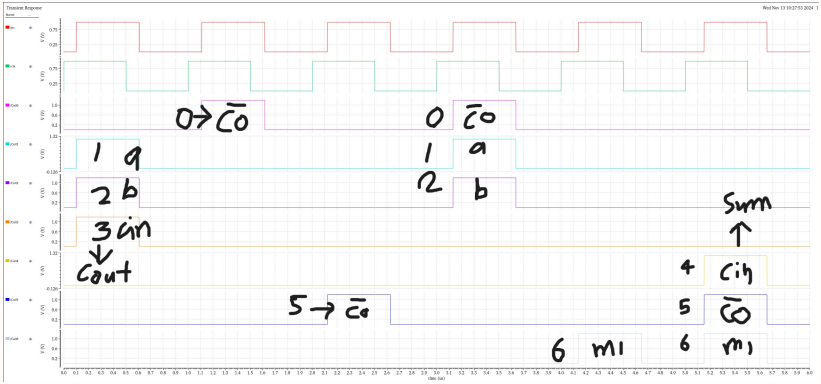


Figure: Waveform for Test 0: Testing of correctly turning ON the respective word lines.

Results Table for Test 0

- **Test Condition:** Turning ON respective word lines based on clock pulses.
- **Cycle 1:** Cells for *a*, *b*, and *cin* are activated (Cell-1, Cell-2, and Cell-3).
- **Cycle 2:** *Cell-0* (cob) is activated for writing.
- **Cycle 3:** *Cell-5* (cob) is activated for writing.
- **Cycle 4:** Cells for *a*, *b*, and *cob* (Cell-1, Cell-2, and Cell-5) are read to compute *m1*.
- **Cycle 5:** *Cell-6* (m1) is activated for writing.
- **Cycle 6:** Cells for *cin*, *cob*, and *m1* (Cell-3, Cell-5, and Cell-6) are read to compute the sum.

Circuit Diagram for Test 1

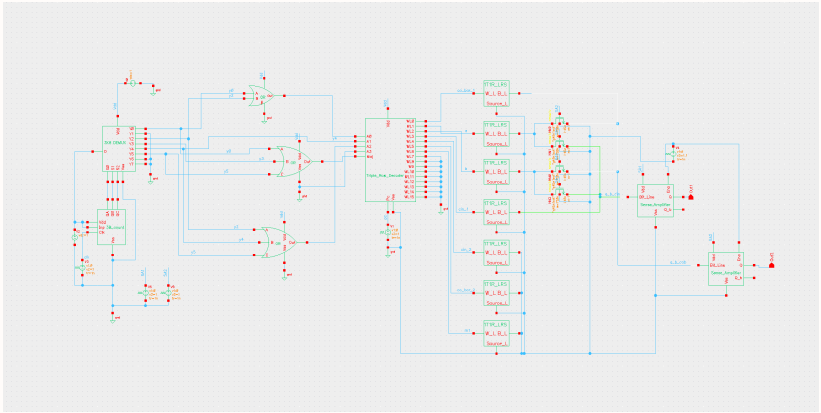


Figure: Circuit for Test 1: Testing of correctly sensing a,b,cin, and later sensing cob,a,b.

Waveform for Test 1

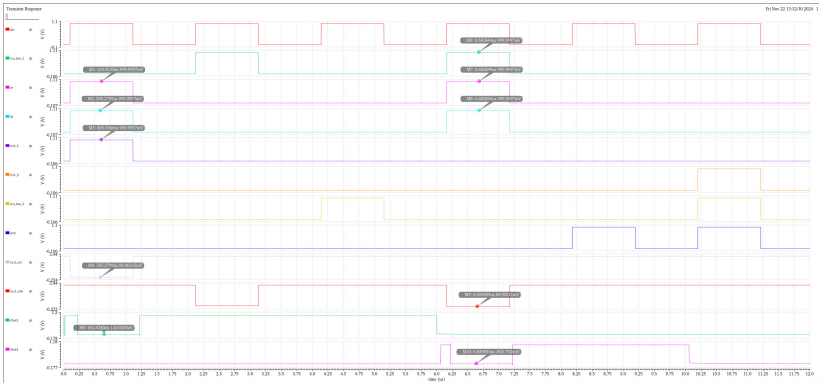


Figure: Waveform for Test 1: Testing of correctly sensing a,b,cin, and later sensing cob,a,b.

Results Table for Test 1

- **Test Condition:** Sensing `a`, `b`, `cin`, and `cob` in sequential cycles.
- **Cycle 1:** Sensed `a`, `b`, `cin` with `Sense Amp` Input readings around 100 mV.
- **Cycle 2:** Sensed `cob`, `a`, and `b` with `cob` at around 100 mV.
- **Data Integrity:** Values of `a`, `b`, and `cin` correctly retained across cycles.
- **Circuit Behavior:** Sequential read operations completed successfully with no data corruption with the help of Pass Transistors.

Circuit Diagram for Test 2

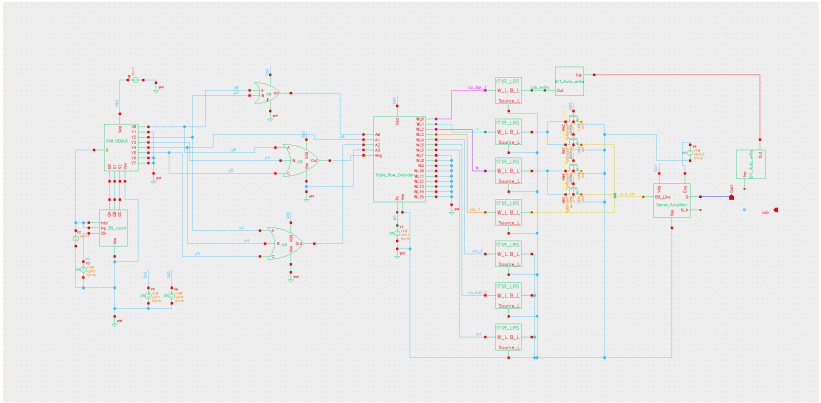


Figure: Circuit for Test 2: Testing of correctly sensing a,b,cin, and writing 1.3V in cob.

Waveform for Test 2

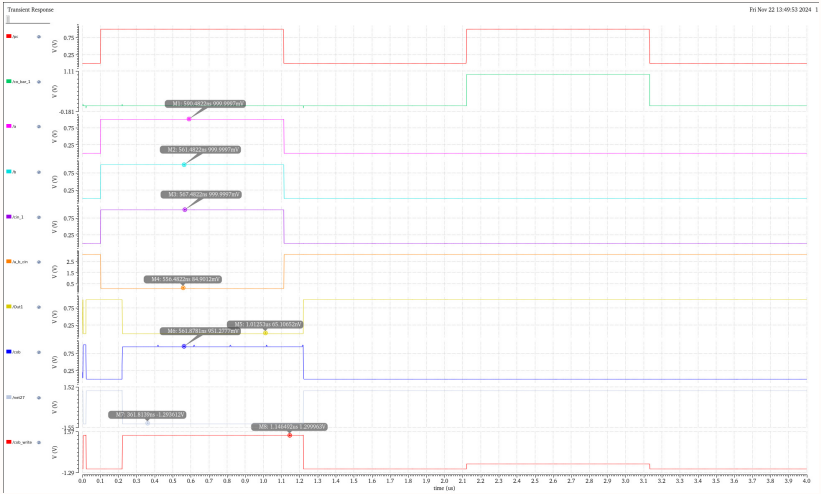


Figure: Waveform for Test 2: Testing of correctly sensing a,b,cin, and writing 1.3V in cob.

Results Table for Test 2

- **Cycle 1:** Cells for *a*, *b*, and *cin* (Cell-1, Cell-2, and Cell-3) are sensed, and *Cell-0* (cob) is written back with 1.3V in the same cycle.
- If $cob = 1$, you write 1.3V into *cob*.If $cob = 0$, you write -1.3V into *cob*.
- I am successfully doing that with the help of two inverters in series.
- **Cycle 2:** *cob 0* became high but 1.3V at bit-line reached earlier only in cycle 1.
- **Cycle 3:** *cob 5* became high but 1.3V at bit-line reached earlier only in cycle 1.
- This is the problem associated that as soon as the Sense Amplifier is sensing the outpur, our circuit is sending 1.3/-1.3V at *cob* in same cycle.
- Key Point: We are able to send 1.3/-1.3V at *cob* bit-line.

Circuit Diagram for Test 3

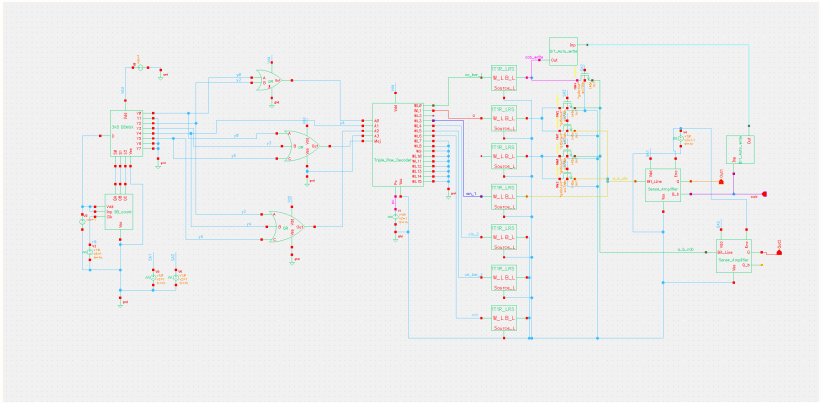


Figure: Circuit for Test 3: Testing of sensing a,b,cin, writing 1.3V in cob and again sensing cob,a,b.

Waveform for Test 3

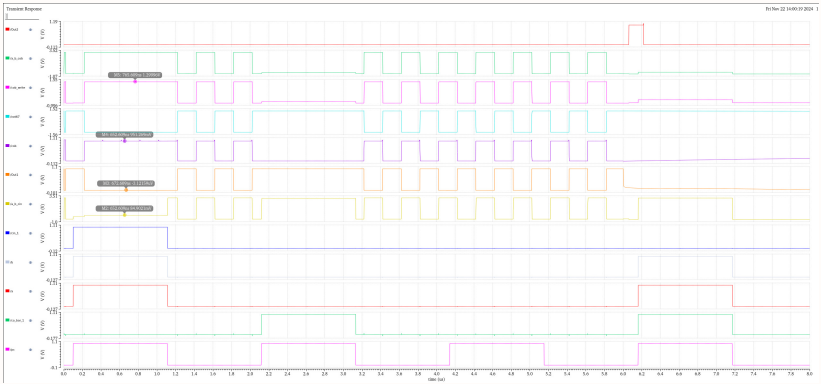


Figure: Waveform for Test 3: Testing of sensing a,b,cin, writing 1.3V in cob and again sensing cob,a,b.

Results Table for Test 3

- **Cycle 1:** Cells for *a*, *b*, and *cin* (Cell-1, Cell-2, and Cell-3) are sensed, and *Cell-0* (cob) is written back with 1.3V in the same cycle.
- If $cob = 1$, you write 1.3V into *cob*.If $cob = 0$, you write -1.3V into *cob*.
- I am successfully doing that with the help of two inverters in series in test-2 already.
- **Cycle 2:** *cob 0* became high but 1.3V at bit-line reached earlier only in cycle 1.
- **Cycle 4:** We are trying to sense *a*, *b*, and *cob* but *cob* is not written properly.
- This is the problem associated that as soon as the Sense Amplifier is sensing the outpur, our circuit is sending 1.3/-1.3V at *cob* in same cycle.
- Key Point: We need to synchronize the writing of cob in cycle 2 and cycle 3 so that in cycle 4 we can read them to generate m1.

Circuit Diagram for Test 4

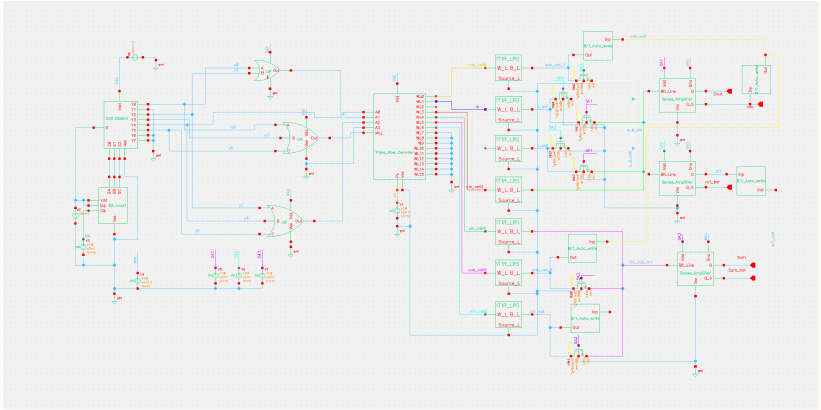
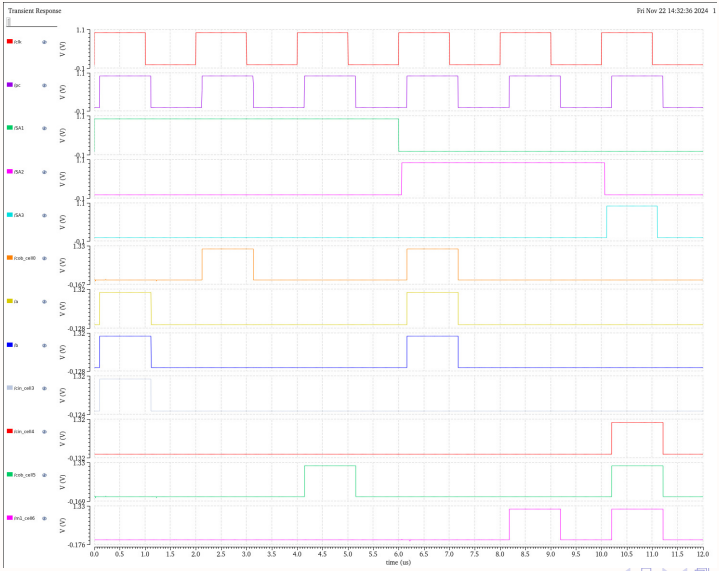


Figure: Circuit for Test 4: Testing of writing 1.3V in cob with 2nd Clock Pulse.

Waveform for Test 4



Results Table for Test 4

- This is the complete Circuit which should work as automated Full Adder using ReRAM Cells.
- Individually, we are able to turn ON the respective word-line in different cycles.
- Individually, we are able to sense a,b,cin or cob,a,b before the writing of cob.
- Individually, we are able to do the writing of cob but in cycle 1 only and that is the problem.
- **If we can do the writing in the correct cycle then the circuit will work properly and we will get Cout and Sum for any set of inputs.**

Circuit Diagram for Test 5

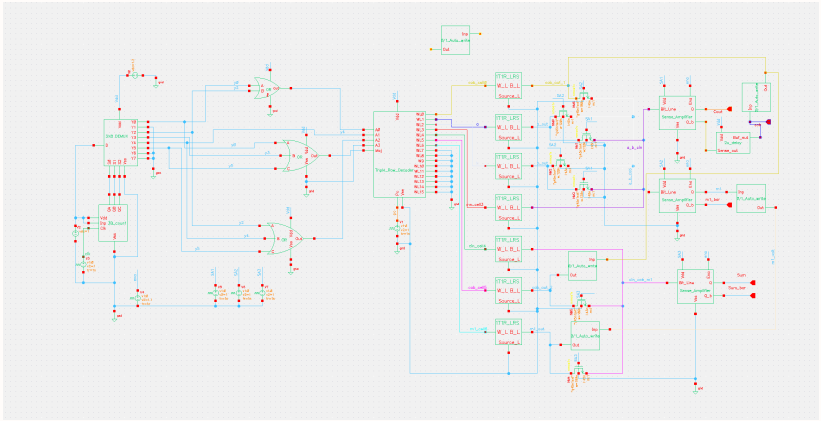


Figure: Circuit for Test 5: Testing of writing 1.3V in cob with 2nd Clock Pulse with a delay element.

Waveform for Test 5

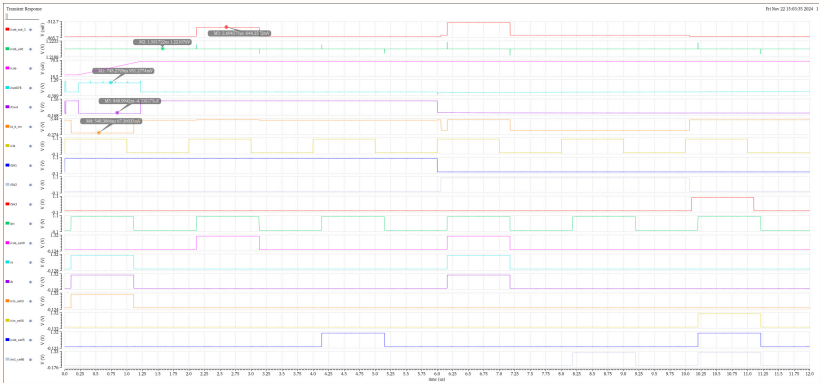


Figure: Waveform for Test 5: Testing of writing 1.3V in cob with 2nd Clock Pulse with a delay element.

Results Table for Test 5

- If $cob = 1$, you write 1.3V into cob.If $cob = 0$, you write -1.3V into cob.
- I am successfully doing that with the help of two inverters in series in test-2 already.
- I tried to add a buffer with 2-microsecond delay, which is the duration of one clock pulse at the output of Sense Amplifier.
- I added a RC circuit at the end of buffer but not getting the correct response.
- **If we can do the writing in the correct cycle then the circuit will work properly and we will get Cout and Sum for any set of inputs.**