Automated Computations using ReRAM Operations for Full Adder

Ayush Yadav

November 23, 2024

IIT Tirupati

Contents

Introduction

Introduction

- This project aims to design and implement an automated full adder using Resistive Random Access Memory (ReRAM) cells in an in-memory computing paradigm.
- This design employs a counter-driven demultiplexer (DEMUX) and a triple row decoder to automate the selection of specific wordlines, enabling precise control over read and write operations.
- No need to perform all cycles one by one. Use a single start command to initialize all the cycles and get the results.

Objective and Scope

Objective and Scope

Objective:

- Design an automated full adder using ReRAM cells to perform in-memory computations with single command.
- Automate the selection of word-lines for specific read and write operations using a counter-driven demultiplexer and triple row decoder design.
- Make circuits so that Read anb write operations can be done from the single RRAM cell can do both and that also automatically.

Scope

Scope:

- Implement a Full Adder circuit with automated computation using majority logic.
- The circuit should be possible to select and on Word-lines as per our wish which can be implemented to other type of computations also like Comparator.
- Making a Memory to remember the codes to turn on the specific Word-Lines for different-different computations
- Explore the challenges of voltage control, timing synchronization, and scalability in peripheral circuit design.
- To do these automatic Computations using the single Sense Amplifier.

Circuit Overview

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

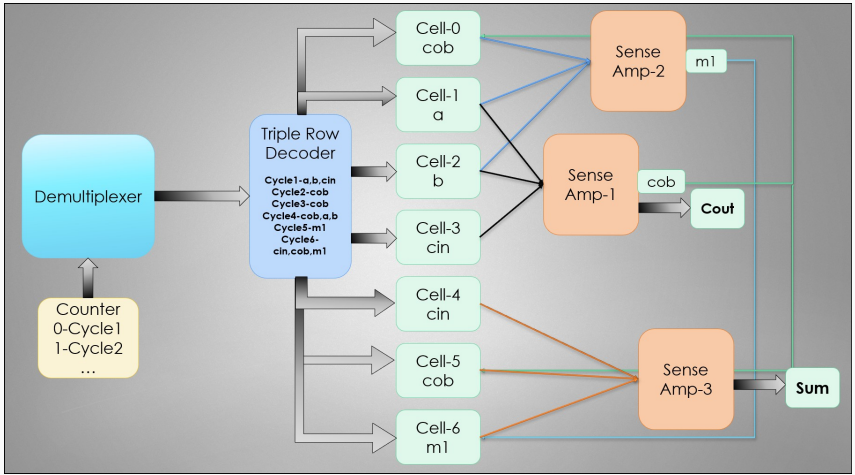


Figure 1: Flow-Chart for the Circuit

Counter

- The counter output determines the specific operation to perform in the circuit.
- Each count is passed to the demultiplexer to activate one output line at a time.
- The activated demultiplexer output is connected to the triple row decoder.
- The triple row decoder decodes the input signal to simultaneously activate the required rows in the memory array.

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

Demultiplexer

- The demultiplexer receives input from the counter.
- It maps the counter's output to one of its output lines (e.g., Cycle 1 to Cycle 6).
- Each demux 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

- The decoder receives demux output and controls word-lines for ReRAM cells.
- Supports:
  - Single-row WRITE operations for storing *cob* and *m*1.
  - 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

- 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.
- Computations and storage occur simultaneously in memory 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 *m*1 (majority of *a*, *b*, *cob*).
- **Cycle 5:** Write *m*1 to Cell-6.
- **Cycle 6:** Compute sum (majority of *cin*, *cob*, *m*1).

Circuit

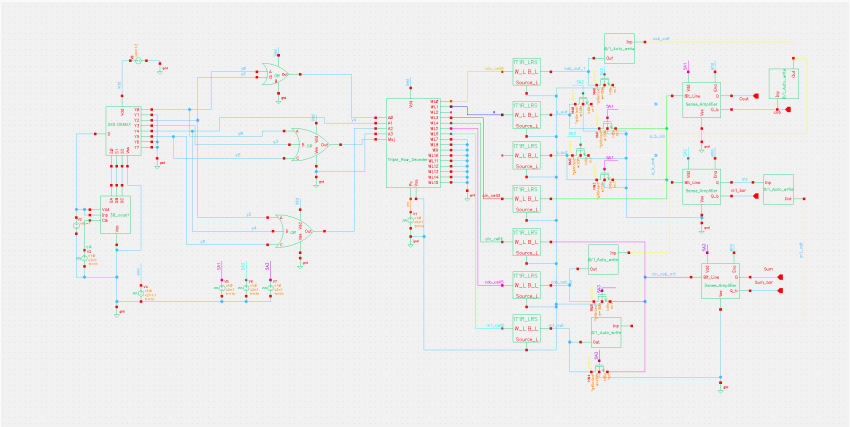


Figure 2: Example Circuit Overview

Results

Test Conditions

- **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.

Circuit Diagram for Test 0

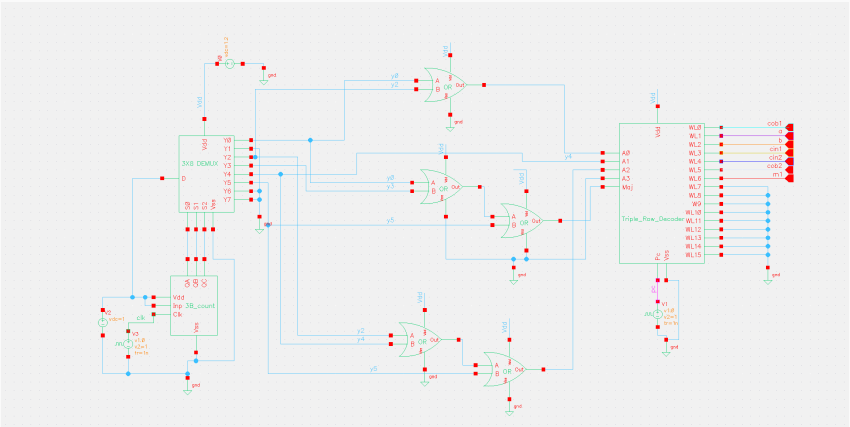


Figure 3: Circuit for Test 0: Testing of correctly turning ON the respective word lines.

Waveform for Test 0

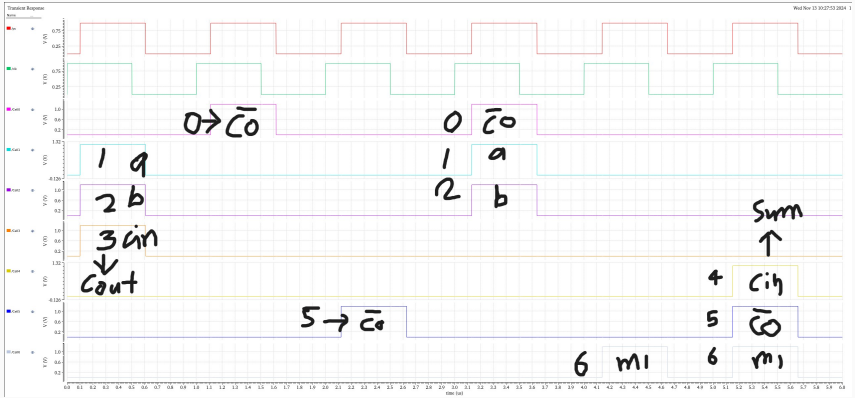


Figure 4: Waveform for Test 0: Testing of correctly turning ON the respective word lines.

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

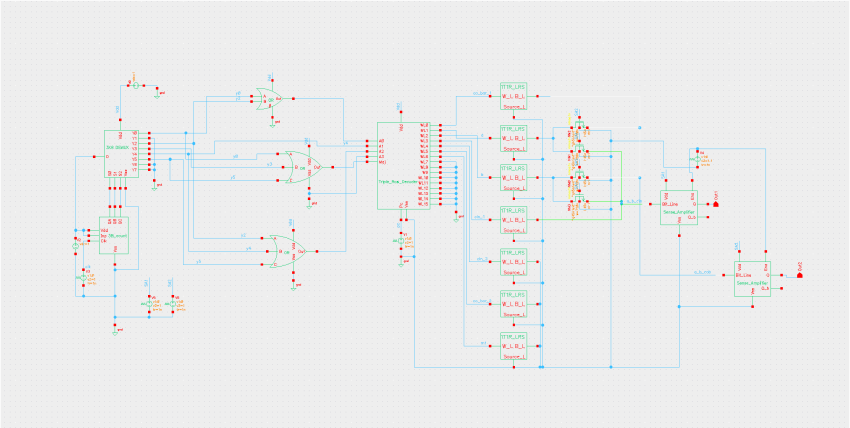


Figure 5: Circuit for Test 1: Testing of correctly sensing a,b,cin, and later sensing cob,a,b.

Waveform for Test 1

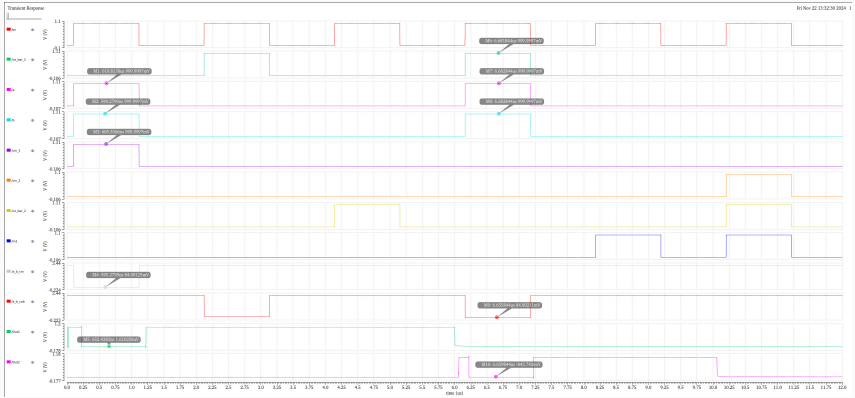


Figure 6: Waveform for Test 1: Testing of correctly sensing a,b,cin, and later sensing cob,a,b.

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

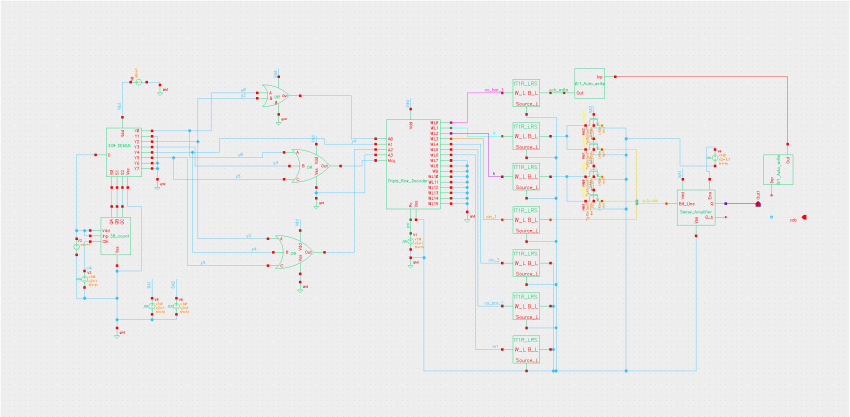
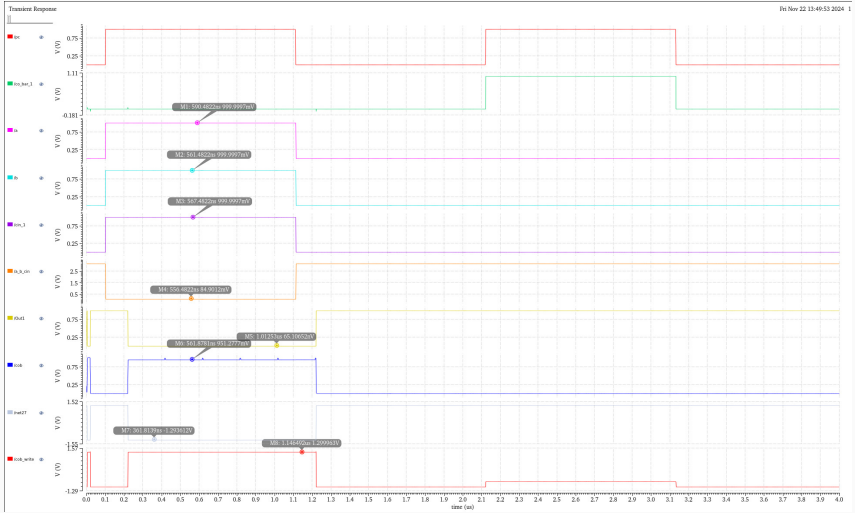


Figure 7: Circuit for Test 2: Testing of correctly sensing a,b,cin, and writing 1.3V in cob.

Waveform for Test 2



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

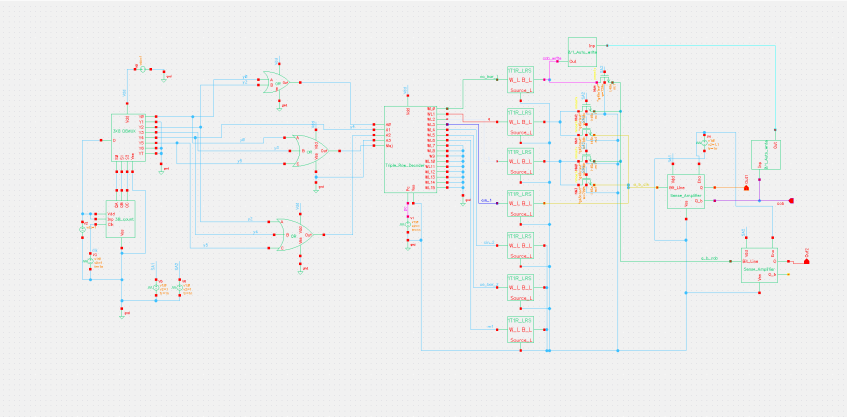


Figure 9: Circuit for Test 3: Testing of sensing a,b,cin, writing 1.3V in cob and again sensing cob,a,b.

Waveform for Test 3

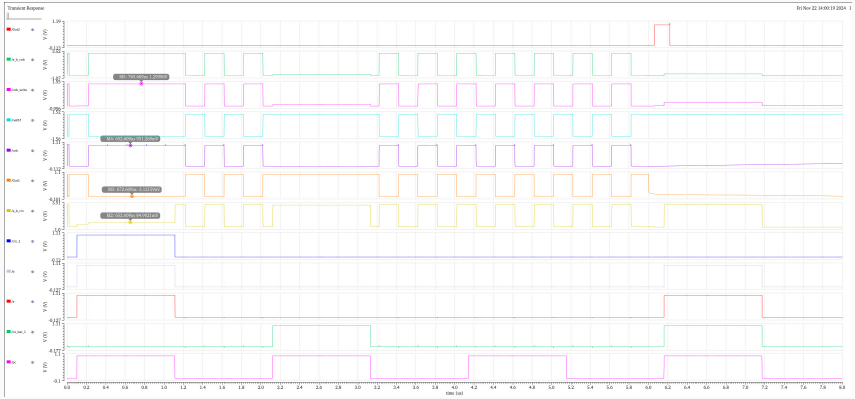


Figure 10: Waveform for Test 3: Testing of sensing a,b,cin, writing 1.3V in cob and again sensing cob,a,b.

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

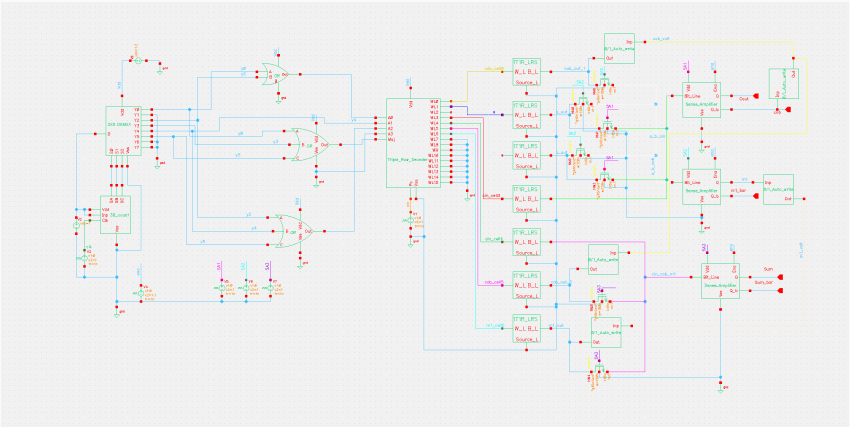
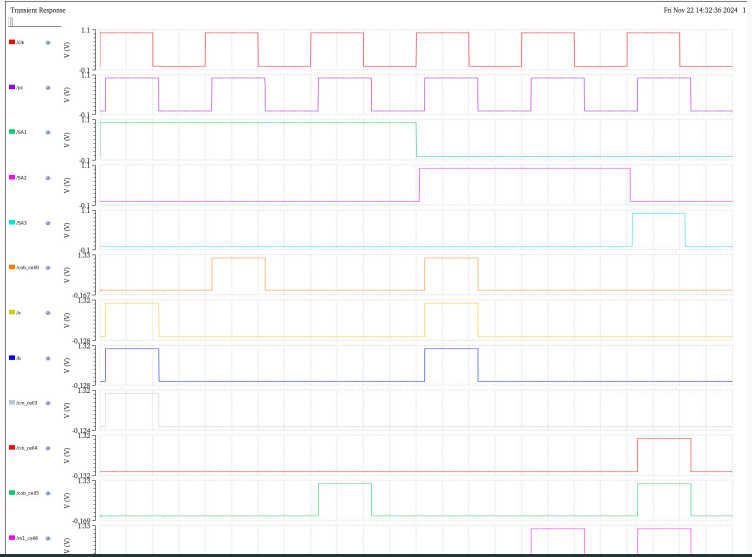


Figure 11: Circuit for Test 4: Testing of writing 1.3V in cob with 2nd Clock Pulse.

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

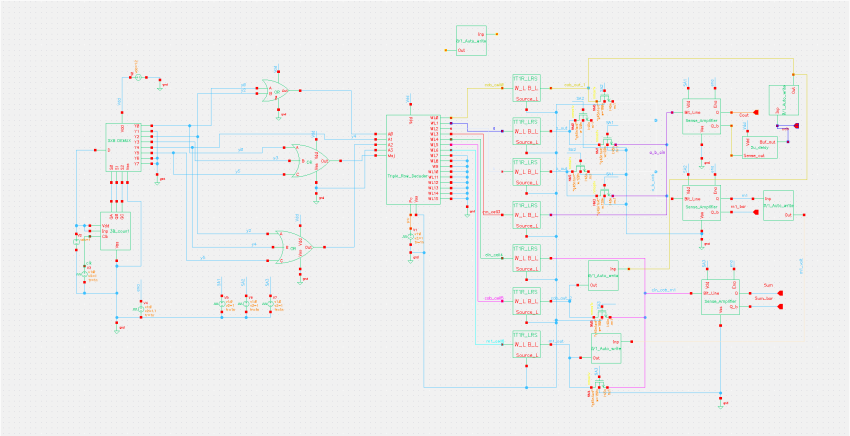


Figure 13: Circuit for Test 5: Testing of writing 1.3V in cob with 2nd Clock Pulse with a delay element.

Waveform for Test 5

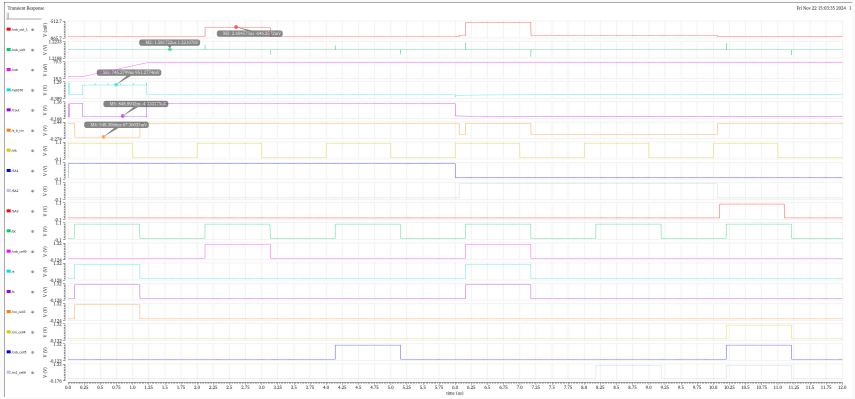


Figure 14: Waveform for Test 5: Testing of writing 1.3V in cob with 2nd Clock Pulse with a delay element.

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.**

Problems

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.

Conclusion

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.