Ternary Content-Addressable Memory (TCAM) Array with a Priority Encoder

Ivan Huang(qh229), Jiaying Zhang(jz2354), Kevin Wang(kw633), Qingmiao Xiao(qx99), Xiaoyu Liang(xl434)

Abstract

This project aims to design, implement, and verify a 32x32 ternary content-addressable memory (TCAM) integrated with a highest-index priority encoder. The system supports 32-bit searches and writes with "don't care" bits, and outputs a 5-bit encoded match address. We designed components, including TCAM cells, row architecture, and encoder, at the schematic level, optimized topology and sizing for speed and layout density, and integrated them into a full system with Verilog-A testbench. We verified functionality and performance through simulations of individual cells, rows, and the entire system at both schematic-level and layout extracted lavel. We achieved an extracted delay of 96.037 ps from search input to matchline and 241.419 ps through the encoder. Post-extraction total power consumption was 70.349 pJ, and the layout area was minimized to 97um x 54um. The final design demonstrates correct matching behaviour and timing performance as we expected.

Motivation

Modern data processing and networking systems demand rapid associative lookups – eg. IP routing and access control filtering, where the address of the desired data is not known in advance. In conventional RAMs, data is fetched by supplying an address. Hence, performing an associative search requires iterating through all the addresses, making it unsuitable for such tasks. Content-Addressable Memories (CAMs) invert this access model: the search data is compared to all stored words simultaneously, and any rows that match assert a hit line in the same clock cycle. Ternary CAMs (TCAMs) further extend this capability by allowing "don't care" bits.

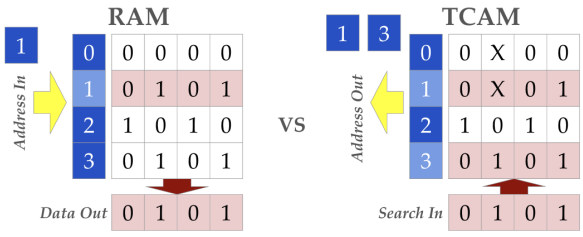

Figure 1. Comparison between RAM and TCAM.

Figure 1 is an example that shows the comparison between RAM and TCAM. For RAM, a specific address is provided as the input (in this case, address 1), and the RAM returns the data stored at that address. Since RAM operates like a simple lookup table, accessing data with unknown addresses would take O(n) time as we need to search through every entry in the memory in the worst case. In contrast, TCAM simultaneously compares the search key to all stored entries in parallel in a constant time O(1) search in theory (note that actual circuit delay increases with N due to longer

match lines and parasitic impedances). Overall, TCAM supports high speed, parallel search, and supports "don't care" bits for flexible and pattern matching. These features make it ideal for tasks such as packet filtering, cache tags, and AI model pattern matching. In our project, we implemented a 32x32 TCAM array with an integrated priority encoder, with a focus on achieving high speed and compact layout.

Theory of Operation

TCAM Cell is the basic unit that stores one bit in ternary form 0, 1, or X don't care.

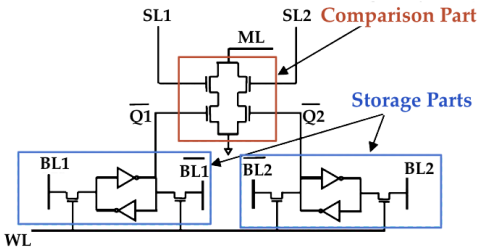

Figure 2a. Circuit Diagram of TCAM Cell

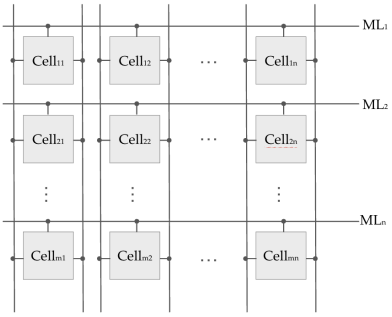

Figure 2b. Diagram of TCAM Array

Several signal lines are included in a typical TCAM cell:

- **BL1**, **BL2** (Bit Lines 1 and 2): These lines are used during write operations to program or update the stored value in the TCAM cell. They carry the data bits that will be written into the cell.
- **Q1**, **Q2**: These are the internal storage nodes within the TCAM cell that hold the stored bit and its complement. They maintain the stored ternary value (0, 1, or "don't care") for comparison.
- **SL1**, **SL2** (Search Lines 1 and 2): These lines carry the search key bits to the TCAM cell. They are used to compare the stored data against the input search pattern during a lookup operation.
- **ML** (Match Line): This line indicates whether the stored data in the TCAM cell matches the search input. If all bits match (considering "don't care" conditions), the ML remains active, signaling a successful match.

The structure design of a TCAM cell has two parts:

- The **storage** blocks are two SRAM-like subcells, each made of cross-coupled inverters and pass transistors. Each subcell stores one bit, which allows the TCAM cell to encode three distinct states instead of just binary. The dual-storage structure of TCAM is what distinguishes it from CAM, and enables the handling of an additional "don't care" during both write and search operations.
- The **comparison** circuit in a TCAM cell is built around an XOR-based pull-down network (PDN) connected to the match line (ML). Prior to the search operation, all matchlines are pulled high via a PMOS transistor. During search, the stored values

(Q1, Q2) are compared with search lines SL1 and SL2, and any miss-matched bits in any row will pull the matchline low via the XOR PDN.

Stored Data / Match Line / Select Line	0 (BL1=0, BL2=1)	1 (BL1=1, BL2=0)	X (BL1=1, BL2=1)
0(SL1=0, SL2=1)	1	0	1
1(SL1=1, SL2=0)	0	1	1
X(SL1=0, SL2=0)	1	1	1

Table 1. Truth Table of Matchline

Table 1 shows how a TCAM cell determines whether input data matches the stored data. The rows represent the value stored in the cell, defined by select lines (SL1, SL2), while the columns represent input being searched, defined by the select lines (BL1, BL2). A '1' in the table means the input matches the stored value, and accordingly '0' means mismatch.

A TCAM array is built by organizing many individual TCAM cells into a structured grid of rows and columns shown in Figure 2b. To construct a TCAM word, multiple TCAM cells are placed side by side in a row, with each cell storing one bit of the full word. All cells in a row share a common match line, so the match line for a row is asserted only if all cells in that row match the corresponding input bits. All rows are checked in parallel during a search.

Expected specification dependencies:

- The array stores 32x32 bits and the PE converts the 32 matchlines into a 5-bit address.
- Delay for searching in the TCAM should be low (<=100ps) as the only logic is the pull-down NFETs. Delay through the PE should be high (>=200ps) due to density of combinational logic.
    - ***Search Delay***: $t_{PD} \sim t_{pre} + N \cdot t_{cell} + t_{encoder}$
- Power consumption for search, especially post-extraction, should be a magnitude higher than the writing for the lab4 SRAM, as search is done in parallel for all rows. Power consumption for the PE should be similar to that of the decoder to Lab2.
    - ***Search Power***: $P_s \sim \alpha \cdot V_{dd}^2 \cdot f_{clk} \cdot (N \cdot C_{ml} + N \cdot C_{sl} + C_{encoder})$
    - ***Write Power***: $P_w \sim \alpha \cdot V_{dd}^2 \cdot f_{clk} \cdot (2N \cdot C_{bl} + C_{wl})$
- Area for the TCAM should be >2x that of the Lab4 array due to every cell having two SRAM cells. PE area should be approx. that of the Lab2 decoder.
    - ***Area***: $A_{tot} \sim N \cdot N \cdot A_{cell} + A_{encoder}$

<u>Top level description</u>

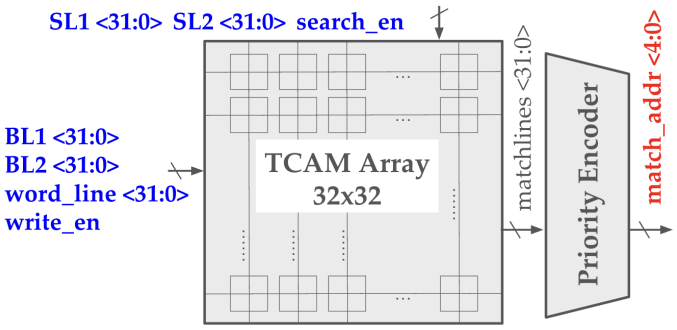

Figure 3. 32x32 TCAM Array Diagram

As shown in Figure 3, our memory system—composed of a TCAM array and a priority encoder—supports two modes of operation:

- **Write operation:** When write_en is asserted, one-hot word_line[31:0] selects one of the 32 rows, and the ternary data is written to that specific row using BL1[31:0] and BL2[31:0].
- **Search operation:** When search_en is asserted, all matchlines are precharged high, and the circuit compares SL1[31:0] and SL2[31:0] (search inputs) with stored data. Miss-matched bits in any given row will result in pulling the corresponding matchline low.

The resulting 32-bit matchlines[31:0] feed directly into a 32-bit Priority Encoder, which outputs match_addr[4:0] for the highest-indexed.

<u>Key Sub-circuits' Schematics</u>

1. **TCAM Cell**

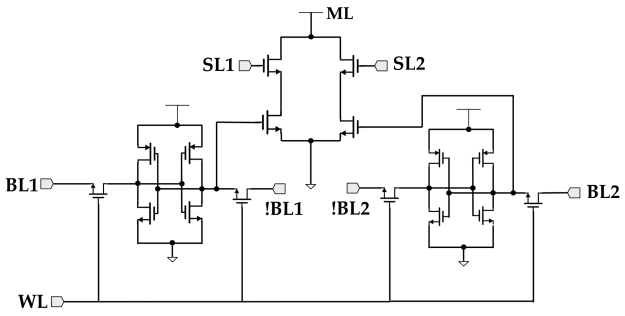

Figure 4. TCAM Cell Schematic

The TCAM cell consists of two SRAM cells and an XOR pull-down network. Key optimization challenge involved the sizing of the cells and the pull-down NFETS for matching, which on a single bit has to be strong enough to overcome the parasitics of the entire row. To ensure reliable pull-down in the worst-case scenario, the pull-down transistor was sized to 120nm x 4. The other transistors are kept at minimum sizing to save area and power.

2. **Bitline Precharge & Write Cell**

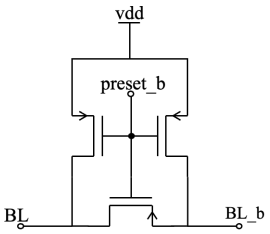

Figure 5a. Bitcell Precharge Schematic

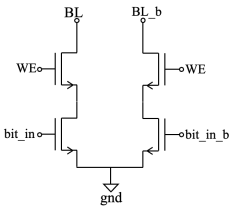

Figure 5b. Write Cell Schematic

The bitline precharge (Figure 5a) and the write cell (Figure 5b) support the writing operation for the SRAM cells. Due to the larger size of our array (32x32), the FET size has to be increased to between 3x and 4x to ensure correct writing behavioral post-extraction.

3. **TCAM Row w/ Matchline Precharge**

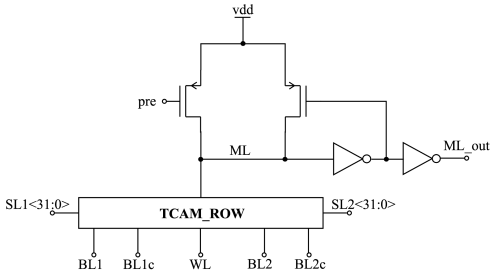

Figure 6. TCAM Row with Matchline Precharge Schematic

The TCAM cells are combined to form a 32-bit row (Figure 6), with their matchline output combined. The matchline is pre-charged via a PMOS and uses domino logic at the output. This is the same arrangement as introduced in lecture. Inverters are kept unit-sized and the two pull-up PMOS are scaled to ~3x to balance with the pull-down NMOS in the cells.

4. **Full TCAM array + PE with test signal generator implemented in Verilog-A**

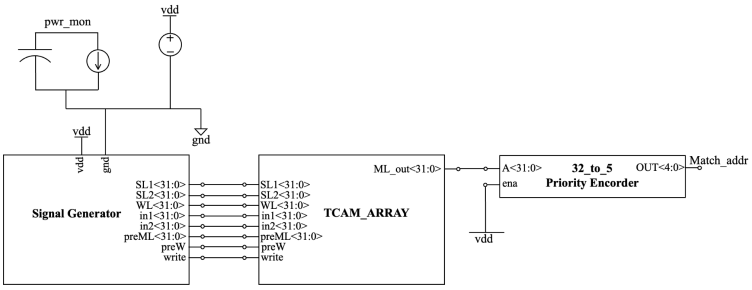

Figure 7. Full TCAM Array + PE with Test Signal Generator

The inputs of the TCAM array are driven by a Verilog-A module, which assumes sufficient driving strength. It includes control signals for writing, which uses the same timing as Lab4, and for searching. The output of the array is connected to the PE.

5. **32-to-5 Highest-Index Priority Encoder**

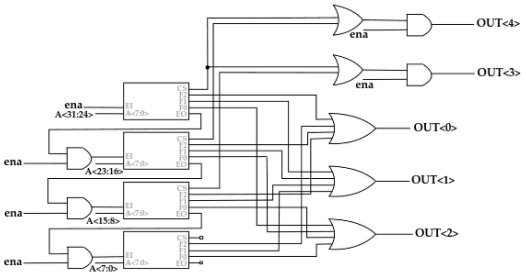

Figure 8. 32-to-5 Highest-Index Priority Encoder

The 32 to 5 encoder identifies the highest priority active input (from 32 inputs) and outputs a 5-bit binary code corresponding to that input's position. We used four 8 to 3 priority encoders. Each encoder handles 8 inputs and outputs a 3-bit binary code indicating the highest active input among its group. These encoders are chained, so one encoder is active at a time, and they are checked in order of the descending priority.

Layout

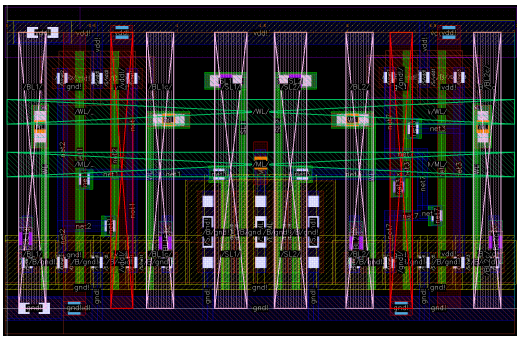

Figure 9. TCAM Cell Layout

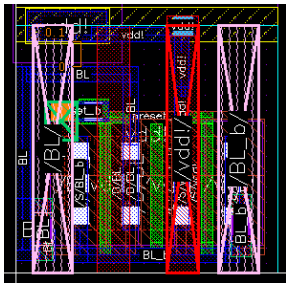

Figure 10a. Bitline Precharge

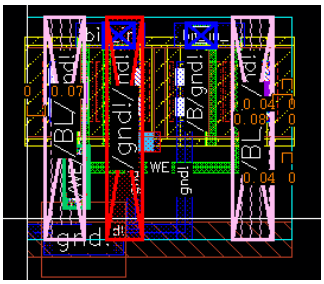

Figure 10b. Write Cell Layout

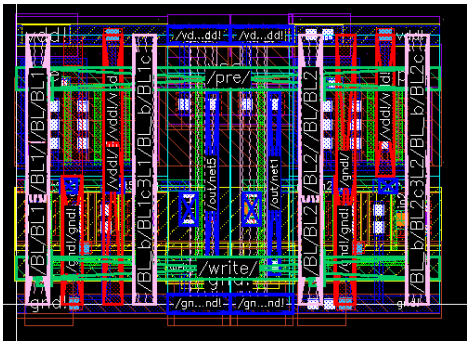

Figure 10c. Column Write Unit Layout

TCAM Cell

- Size: 2.98um x 1.625um
- To make the cell tileable, we use vertical Metal 2 and 4 for all the bitlines (and their complements) and search lines. Wordlines and matchlines use horizontal metal 3.

Bitline Precharge & Write Cell & Combined Column Write

- Sizes: 1um x 1um (Figure 10a), 1um x 0.8um (Figure 10b), 3um x 1.9um (Figure 10c)
- The column write unit consists of two bitline precharge and two write cells, since there are two SRAM cells per TCAM cell. We match the bitline widths on them to tile with the TCAM cells. Metal 3 is again used for horizontal routing, here for the pre and write signals.

Full TCAM array

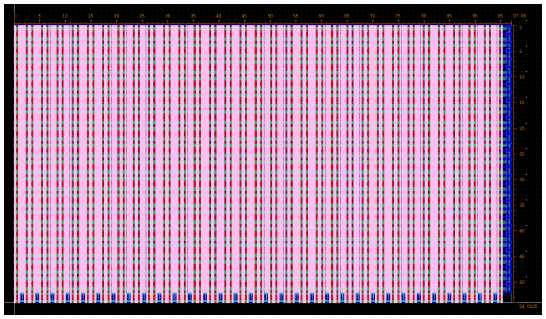

Figure 11. TCAM Array Layout

- Size: 97um x 54um
- The 32x32 array is tiled by aligning the bitlines and searchlines mentioned above. The precharge and matchline out domino logic circuitry are placed on the right side. The bottom row consists of the bitline precharge and write circuitry. This layout minimizes area.

Simulation Result

Pre-extraction Schematics:

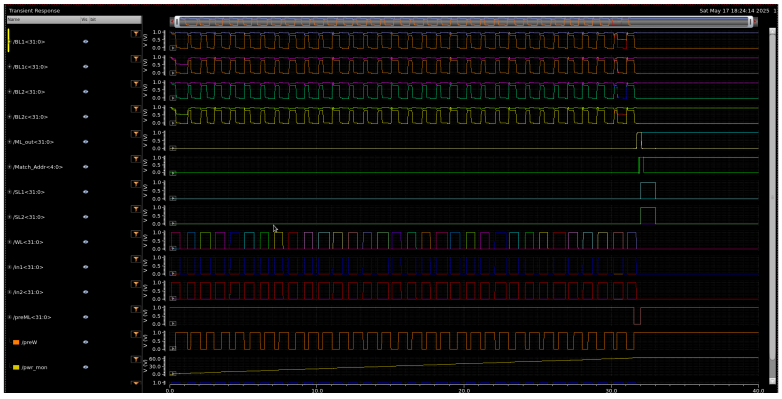

Figure 12. Array+PE Simulation Waveform (schematic)

Figure 13. Zoomed-in markers showing delay from SL to ML (schematic)

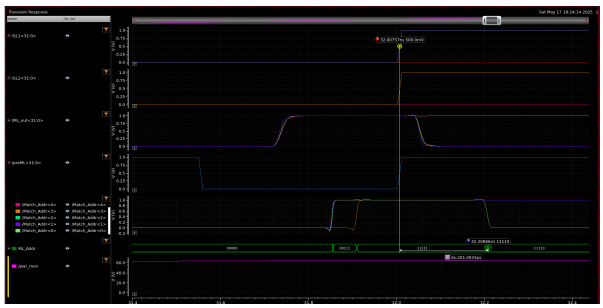

Figure 14. Zoomed-in markers showing delay from SL to PE (schematic)

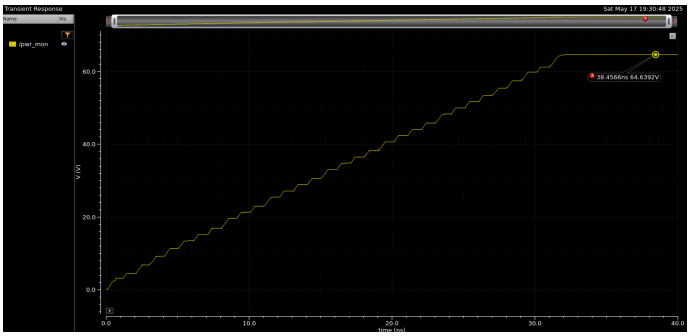

Figure 15. Power Consumption (schematic)

$T_{sl->ml}$(ps)	47.088
$T_{sl->pe}$(ps)	201.093
Total Power (pJ)	64.639
Average Write Power (pJ)	1.980
Search Power (pJ)	1.276

Table 2. Performance of TCAM (schematic)

Verilog-A is used to generate all the input data and control signals. Figure 12 shows the full waveform. First, we write data into all 32 rows of the TCAM. During each write cycle, all bitlines are first precharged. The corresponding WL is asserted high, and data is written to the cells. After the 32 write operations, preML<31:0> are asserted low to precharge the matchlines, and the search signals pull down mismatched lines.In this case, we are writing 32'0F0F0F0F to the first 29 rows and the 31st row, and writing 32'55555555 to the others, while searching for 32'55555555. Therefore, there are 31 mismatches and a single match in row 30. The PE output Match_Addr[4:0] is at 30 (11110), which is correct. Tests for edge cases are performed. For example, mismatches on only one bit tests for sufficient driving on the pull-down NMOS. Ternary searches and writes were also considered. Pre-extraction, our worst-delay from the search line to the matchlines is ~47ps, and ~200ps through priority encoder. This is consistent with our expectations. The total power consumption in the test case mentioned above is **64.639** pJ, including 32 write and 1 search. As shown in the Figure 16, the point where

the **preML** signal is pulled low serves as a boundary: the power consumption before this point is due to the 32 write operations (63.362 pJ), while the power consumption after it is caused by the search operation (1.276 pJ). Therefore, as in Table 2, the average write power can be calculated by 63.362/32 = 1.98 pJ.

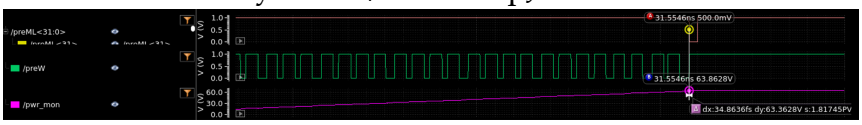

Figure 16. Power Consumption Breakdown Between Write and Search Operations

Post-Extraction:

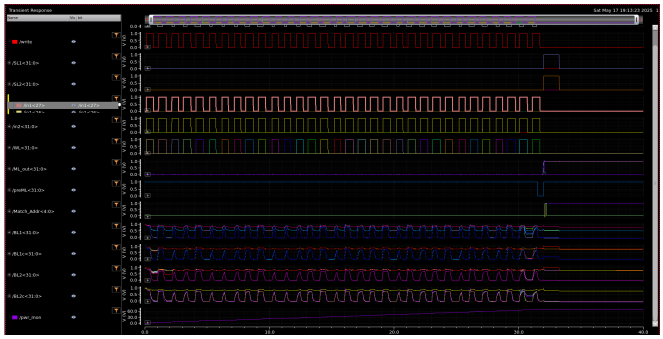

Figure 17. Array+PE Simulation Waveform (extracted)

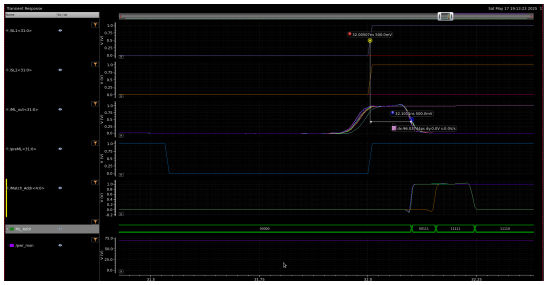

Figure 18. Zoomed-in markers showing delay from SL to ML (extracted)

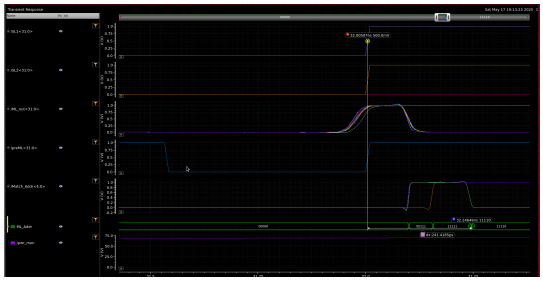

Figure 19. Zoomed-in markers showing delay from SL to PE (extracted)

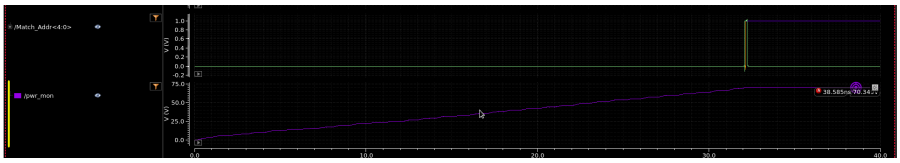

Figure 20. Zoomed-in markers showing delays and power (extracted)

$T_{sl->ml}$(ps)	69.037
$T_{sl->pe}$(ps)	241.419
Total Power (pJ)	70.349
Average Write Power (pJ)	2.155
Search Power (pJ)	1.390

Table 3. Performance of TCAM (extracted)

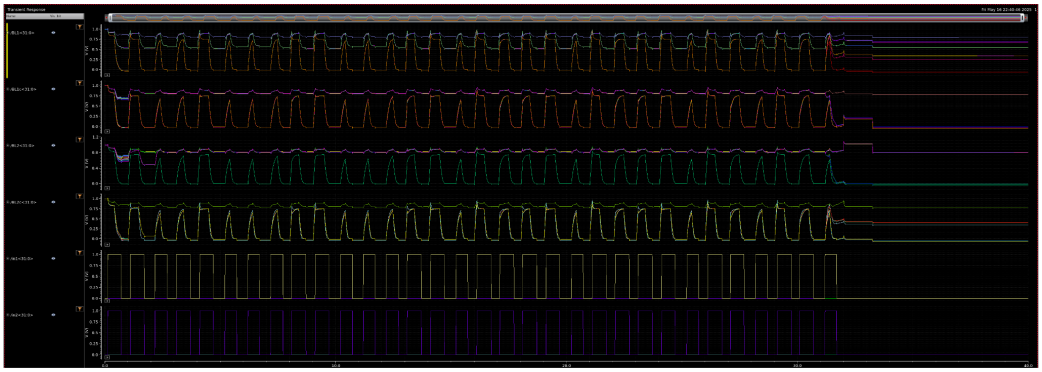

Figure 21. Bitline swings as a result of extraction

Using the same test case, we simulate the post-layout extracted version of the circuit. The full waveform is shown in Figure 17. As expected, extraction increased delays and power consumption (Table 3) due to accounting for parasitic impedances. It also introduced bitline swings (Figure 21), which were previously idealized in schematic simulations, leading to more realistic signal degradation and slower sensing behavior.

Discussion

The Verilog-A testbench was tricky to set up: we could not find a way to generate signals with loops. We utilized Python (!) scripts to unroll the ~5000 lines of Verilog-A code needed to write full 32-bit values to the 32 entries. On the schematic side, we had to experiment sizing for the ML and SL transistors. The size of our array meant that these transistors have to be strong in order to pull to the correct value. For example, inefficient size in the pull-down NMOS will result in the matchline staying high when only one bit misses. This is also the worst-case delay case as more NMOS engaging in pull-down is faster. Extraction posed numerous problems. As mentioned previously, the bitlines became slow at pulling down, with some bits showing uncertain values. This affects the values being written into the TCAM cells. Although they pass most of our array_tb tests, we do not expect the system to be completely correct post-extraction.

Citation

- S. Kumar, A. Noor, B. K. Kaushik and B. Kumar, "Design of Ternary Content Addressable Memory (TCAM) with 180 nm," 2011 International Conference on Devices and Communications (ICDeCom), Mesra, India, 2011, pp. 1-5, doi: 10.1109/ICDECOM.2011.5738528.
- K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: a tutorial and survey," in IEEE Journal of Solid-State Circuits, vol. 41, no. 3, pp. 712-727, March 2006, doi: 10.1109/JSSC.2005.864128.
- B. Agrawal and T. Sherwood, "Ternary CAM Power and Delay Model: Extensions and Uses," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 16, no. 5, pp. 554-564, May 2008, doi: 10.1109/TVLSI.2008.917538.

Appendix :

- TCAM array layout passing LVS and DRC