Design of an 8x8 SRAM Array

IRFANUL ISLAM
Department of Electrical & Computer Engineering
Concordia University
Student ID: 40207134

OMAR DABAYEH
Department of Electrical & Computer Engineering
Concordia University
Student ID: 40100195

***Abstract*— The objective of this project is to design and implement an 8-row by 8-bit SRAM memory array using TSMC 180nm CMOS technology. The memory array is accessed through a 3-to-8 decoder, driven by a 3-bit address. The SRAM cells are optimized to achieve low power consumption, while operating efficiently during read and write cycles. The test bench, design, and layout of the SRAM array and its peripheral circuitry, including the decoder, are developed and validated using Cadence Virtuoso's Analog Design Environment (ADE). Simulations are carried out to verify functionality, power efficiency, and timing performance, ensuring the design meets modern embedded system requirements.**

Keywords—Cadence, CMOS, Decoder, SRAM, Layout.

I. INTRODUCTION

Static Random Access Memory (SRAM) is a type of semiconductor memory widely used in digital electronics due to its high speed and low power consumption during operation. Unlike Dynamic Random Access Memory (DRAM), SRAM does not require periodic refreshing of stored data, making it "static" and more efficient for high-performance applications.

Key Features of SRAM:

1. Volatile Memory: Like other RAM types, SRAM is volatile, meaning it loses its data when power is removed. However, it retains data as long as power is supplied without requiring refresh cycles.
2. Structure: SRAM consists of bistable latching circuits, typically built using six transistors (6T) per cell. This design ensures faster access times and greater reliability compared to DRAM.
3. Speed: With lower latency and faster access times, SRAM is ideal for applications that demand quick data retrieval and processing.
4. Power Consumption: While idle power consumption is low, SRAM can consume more power during active operations compared to other types of memory like DRAM.
5. Integration: SRAM is often used as cache memory in CPUs, GPUs, and other high-speed processors due to its rapid access capabilities. It is also employed in embedded systems, networking devices, and other performance-critical systems.

Advantages of SRAM:

- High Speed: Faster than DRAM due to the absence of refresh cycles.
- Simpler Control Logic: SRAM requires simpler memory controllers compared to DRAM.
- Data Stability: Retains data without the need for frequent refreshing.

Disadvantages of SRAM:

- Lower Density: SRAM requires more transistors per memory cell, making it less dense than DRAM.
- Higher Cost: Due to its complexity, SRAM is more expensive to manufacture.
- Larger Physical Size: The 6T cell structure results in larger memory modules compared to DRAM.

Applications:

- Cache Memory: Used in processors for fast, temporary data storage.
- Embedded Systems: Integrated into microcontrollers and SoCs for specialized tasks.
- Networking: Deployed in routers and switches for rapid data handling.
- Graphics Processing: Used in GPUs for buffering and rendering operations.

SRAM's speed and reliability make it a cornerstone of modern memory technology, essential for systems requiring high-performance computation and efficient data access.

II. DESIGN DESCRIPTION

This project focuses on the design and development of an 8-bit SRAM (Static Random Access Memory) array using TSMC's 180nm CMOS technology. SRAM is a critical component in modern digital systems, providing high-speed, low-latency memory storage for a wide range of applications. The goal of this project is to create an optimized and efficient SRAM layout while maintaining robust functionality and high performance.

Key Design Features:

1. Memory Array: The core of the SRAM is an array of 8-bit word storage cells. These cells are organized to allow efficient storage and retrieval of data with minimal delay.
2. Wordline Decoder: This subsystem uses address input pins to select specific rows in the memory array. By activating the corresponding wordline, it enables access to the storage cells for reading or writing operations.
3. Bitline Drivers: During write operations, these drivers take data from the input pins and drive the corresponding bitlines, ensuring accurate data storage in the selected memory cells.

4. Sense Amplifier: Critical for read operations, the sense amplifier amplifies and stabilizes the small signals read from the bitlines. This ensures reliable data output, suitable for external circuitry.
5. Control Signals: Essential operational modes are determined by control signals:
    - Decoder Enable (DEN): Activates the Decoder for operation.
    - Write Enable (WEN): Specifies write operations.
    - Read Enable (REN): Specifies read operations.
6. Data Bus (D0-D7): The data bus provides an 8-bit bi-directional interface for data input and output, enabling seamless communication between the SRAM and external systems.

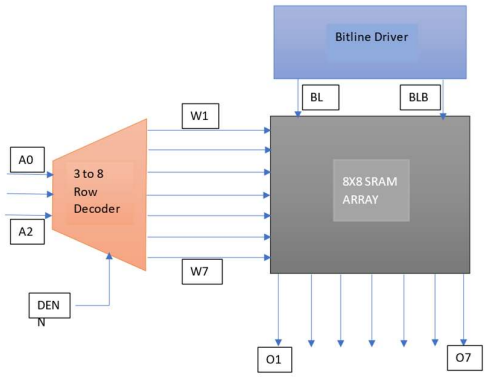

Fig 1: SRAM Array & Decoder Block Diagram

The 1mm wire is connected to each of the outputs O1-O7.

III. Design Hierarchy

The circuit is designed using three basic blocks.

A. Decoder:

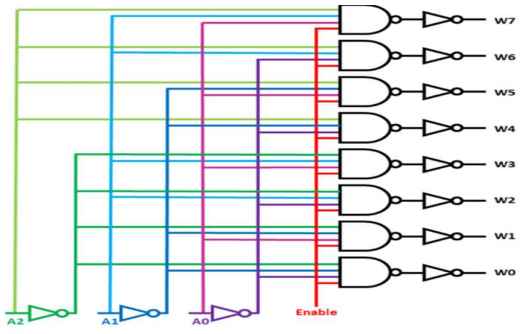

Fig 2: 3 to 8 Decoder Design

The decoder circuit diagram represents a 3-to-8 decoder with an enable signal. It is commonly used in digital systems to activate one of eight output lines (W0 to W7) based on a 3-bit binary input (A2, A1, A0) and the enable signal. Here's a breakdown of the components and functionality:

Components:

1. Inputs:
    - A2, A1, A0: These are the 3-bit address inputs, which determine which output (W0 to W7) is selected.
    - Enable: This signal enables or disables the decoder. When the enable signal is low, all outputs will remain high, regardless of the address inputs.
2. Inverters:
    - Each address input (A2, A1, A0) has an associated inverter to produce the complemented signals ($\overline{A2}, \overline{A1}, \overline{A0}$).
    - These complemented signals are used alongside the original signals for decoding.
3. NAND Gates:
    - There are 8 4-input NAND gates in the circuit. Each NAND gate takes the Enable signal as one input and a unique combination of the original and complemented address signals as the other three inputs.
4. Outputs:
    - W0,W1,…,W7: These are the 8 decoder outputs. Exactly one output will be active (low) at a time based on the 3-bit address input and the enable signal.

Functionality:

- Enable Signal:
    - If the Enable signal is high (1), the decoder is operational, and one of the outputs will be low based on the address inputs.
    - If the Enable signal is low (0), all outputs remain high, effectively disabling the decoder.
- Decoding Logic:
    - Each NAND gate output is determined by a specific combination of address inputs:

$$W_i = \overline{\text{Enable} \cdot \text{Combination of A2,A1,A0 \& their complements}}$$

For example:

- $W0 = \overline{\text{Enable} \cdot \overline{A2} \cdot \overline{A1} \cdot \overline{A0}}$
- $W1 = \overline{\text{Enable} \cdot \overline{A2} \cdot \overline{A1} \cdot A0}$
- $W7 = \overline{\text{Enable} \cdot A2 \cdot A1 \cdot A0}$

Circuit Behavior:

- Active-Low Outputs:
    - For a given input combination (e.g., A2A1A0=000), the corresponding output (W0) will go low (0), while all others (W1–W7) remain high (1).

- This is because a NAND gate produces a low output only when all its inputs are high.

- Example:
  - Suppose A2=0, A1=1, A0=1, and Enable = 1.
  - The decoder activates W3 by generating: $\overline{W3=Enable \cdot \overline{A2} \cdot A1 \cdot A0} = \overline{1 \cdot 1 \cdot 1 \cdot 1} = 0$

Advantages of Using NAND Gates:

1. Simpler Implementation: NAND gates are more commonly used in CMOS design due to their ease of fabrication.
2. Inherent Active-Low Logic: Many digital systems and devices use active-low control signals, making this implementation more practical for such systems.

Fig 3: Decoder Implementation

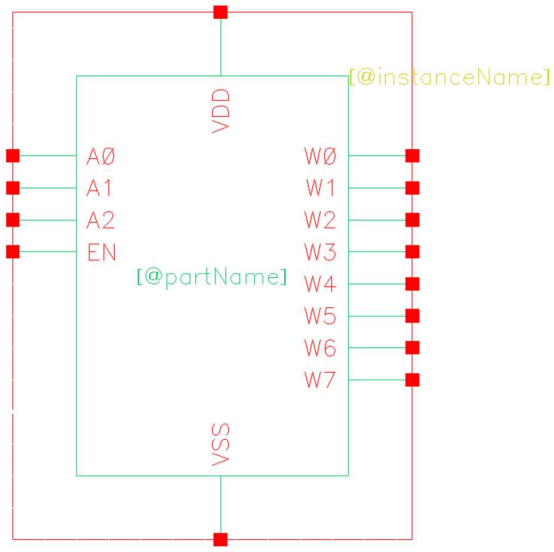

Fig 4: Decoder Symbol

The figure below shows the circuit used to implement the NAND gates for the 3 to 8 Decoder.

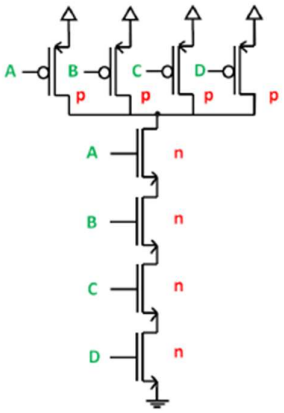

Fig 5: 4 input static CMOS NAND gate

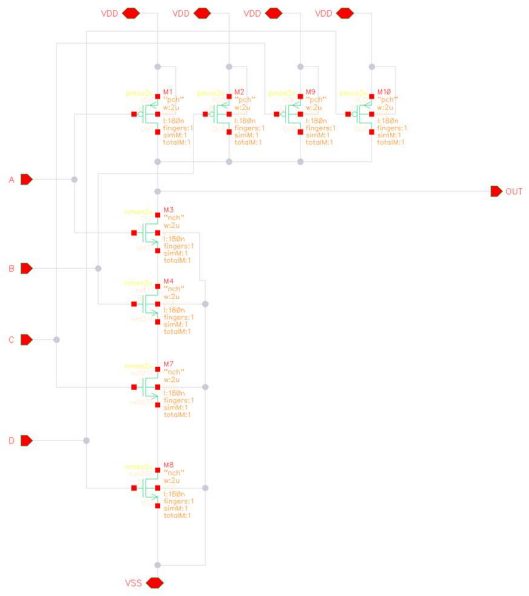

Fig 6: NAND Gate Implementation

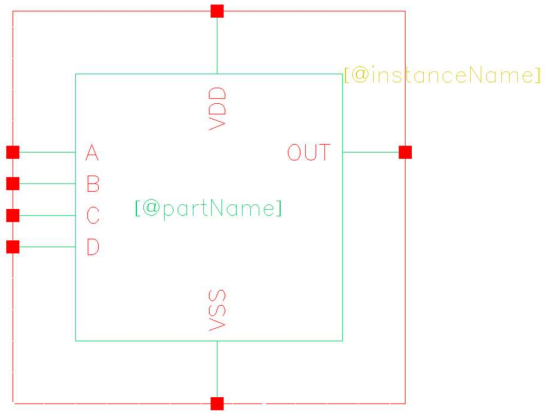

Fig 7: NAND Gate Symbol

B. SRAM:

The figure below shows the SRAM circuit design used to build the 8x8 SRAM array including the sense amplifier.

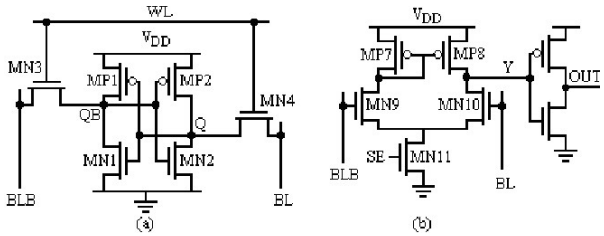

Fig 8: An SRAM Cell & Sense Amplifier

a) 6T SRAM Cell Circuit

This is a standard 6-transistor (6T) SRAM cell, the building block of SRAM arrays. Here's a description:

1. Components:
   - Two cross-coupled inverters:
     - MP1 and MN1 form the first inverter.
     - MP2 and MN2 form the second inverter.
   - Two access transistors:
     - MN3 and MN4 act as the access transistors.
2. Inputs/Outputs:
   - Wordline (WL): Controls access to the cell. When high, it enables the access transistors.
   - Bitline (BL) and Complementary Bitline (BLB): These are used for reading and writing data into the cell.
   - Q and QB: The internal storage nodes that hold complementary data (0 and 1).
3. Operation:
   - Write Operation:
     - The wordline (WL) is asserted high, turning on MN3 and MN4.
     - Data is written into the cell via the bitlines (BL and BLB), forcing the internal nodes Q and QB to the desired values.
   - Read Operation:
     - The wordline (WL) is asserted high, turning on MN3 and MN4.
     - The stored value is sensed by monitoring the bitlines (BL and BLB), which are precharged and then slightly discharged depending on the data stored in the cell.
4. Advantages:
   - Robust data retention due to the cross-coupled inverter structure.
   - High-speed access for both read and write operations.

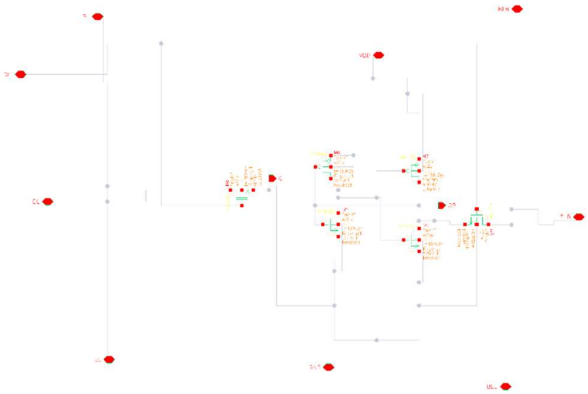

Fig 9: SRAM Cell Implementation

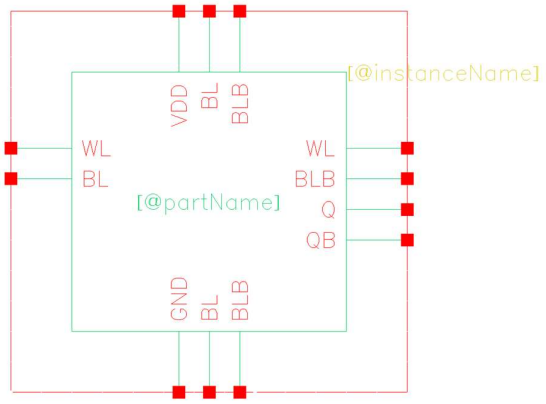

Fig 10: SRAM Cell Symbol

b) Sense Amplifier Circuit

This is a sense amplifier, used to detect and amplify the small voltage difference on the bitlines during a read operation.

1. Components:
   - PMOS Transistors:
     - MP7 and MP8: Act as pull-up devices, ensuring proper operation during amplification.
   - NMOS Transistors:
     - MN9, MN10: Act as the differential pair to sense the voltage difference between BL and BLB.
     - MN11: Acts as an enable switch, controlled by the sense enable (SE) signal.
2. Inputs/Outputs:
   - Bitline (BL) and Complementary Bitline (BLB): Carry the small voltage difference from the SRAM cell.
   - Sense Enable (SE): Activates the sense amplifier when high.
   - Output (OUT): Provides the amplified output signal.
3. Operation:
   - Before reading, the bitlines (BL and BLB) are precharged to the same voltage level.
   - When the sense amplifier is enabled (SE is high), MN11MN11MN11 turns on, and the circuit begins sensing.
   - The differential pair (MN9MN9MN9 and MN10MN10MN10) detects the slight voltage difference between BL and BLB.
   - The output signal (OUT) is driven to a logic level (high or low) based on the sensed difference.
4. Advantages:
   - Amplifies small voltage differences on the bitlines for reliable data readout.
   - Operates with high speed and precision.

Combined Functionality:

The 6T SRAM Cell stores the data, while the Sense Amplifier ensures the stored data is read correctly, even when the voltage differences on the bitlines are small. These circuits are critical in achieving fast and reliable memory operation in SRAM systems.

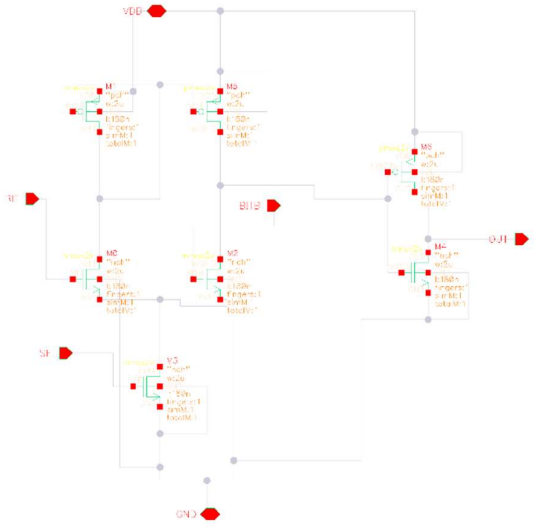

Fig 11: Sense Amplifier Implementation

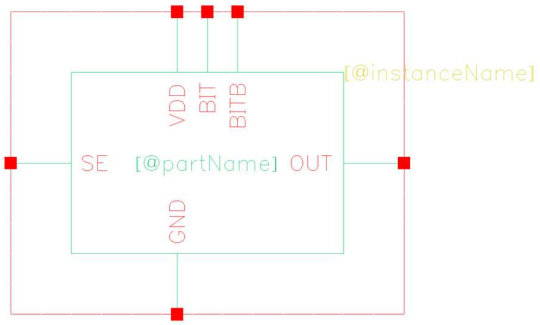

Fig 12: Sense Amplifier Symbol

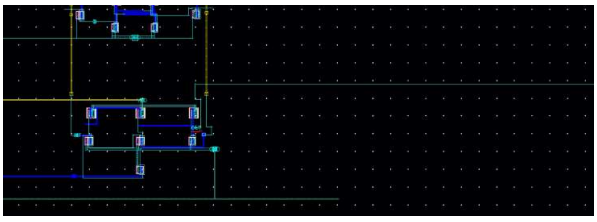

Fig 13: 1mm wire on 1 Sense Amplifier

C. Bitline Drivers:

This component acts as a bitline driver in an SRAM system, ensuring proper control of the bitlines during read and write operations. Its primary roles are:

1. To drive the bitlines during a write operation with the correct logic levels corresponding to the input data.
2. To isolate the bitlines during a read operation, allowing the sense amplifier to detect small voltage differences accurately.

Circuit Functionality:

1. Write Operation:

- When WEN (Write Enable) is asserted:
  - The circuit actively drives the bitlines (BL and NOTBL) with the input data (IN).
  - The inverters and the PMOS/NMOS transistor pairs ensure that the correct voltage levels are driven onto the bitlines.
  - For example:
    - If IN = 1, BL is driven high (VDD), and NOTBL is driven low (VSS).
    - If IN = 0, BL is driven low (VSS), and NOTBL is driven high (VDD).
  - This ensures that the correct logic levels are applied to the SRAM cell for writing data.

2. Read Operation:

- When REN (Read Enable) is asserted:
  - The circuit likely disables the driving of the bitlines (BL and NOTBL) to avoid interfering with the small voltage signals generated by the SRAM cell during a read.
  - Instead, the bitlines are left in a high-impedance state or precharged to a specific voltage level (e.g., VDD) before sensing.

3. Idle State:

- When neither WEN nor REN is active:

The circuit may enter a low-power state, where it does not drive the bitlines actively, reducing power consumption.

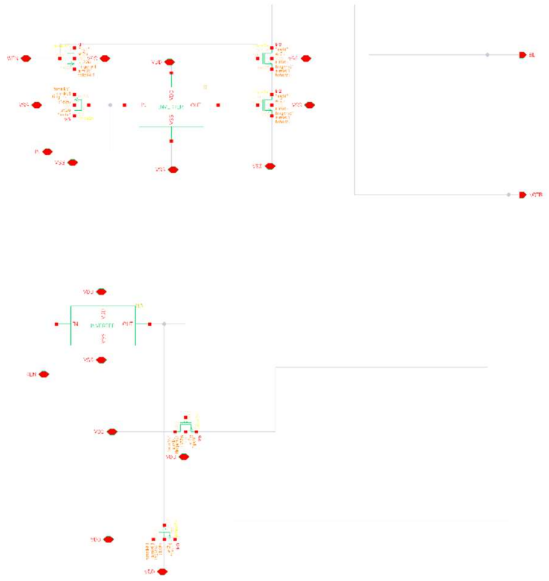

Fig 14: Bitline Drivers Implementation

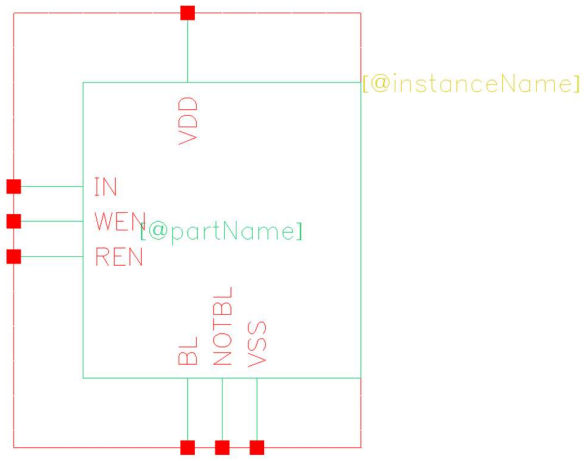

Fig 15: Bitline Drivers Symbol

Final Circuit:

The circuit below shows the final implementation in Cadence using all the components listed above.

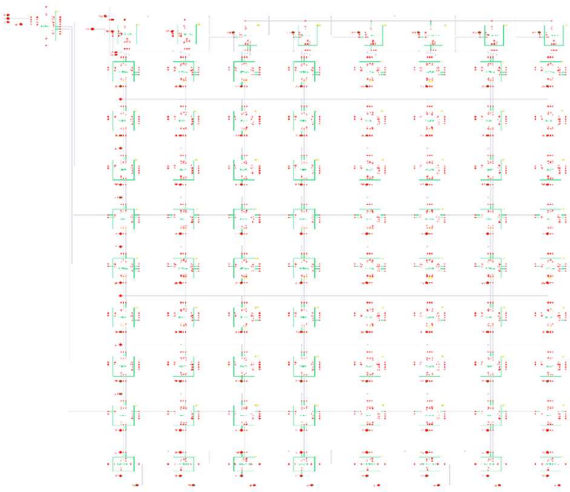

Fig 16: Final Circuit Implementation

IV. SIZING

We chose not to perform detailed sizing for the SRAM cell because the performance and stability of the circuit were found to be satisfactory. Through our simulations and initial testing, we observed that the design met the required operational criteria without the need for further optimization. Additionally, due to time constraints, we prioritized completing the schematic, layout, and other essential tasks over fine-tuning the sizing, as it was deemed non-critical to achieving the project objectives within the given timeline.

V. LAYOUT (LVS/DRC)

Below are the screenshots for LVS & DRC for each component & sub units including the final circuit. The appendix folder attached to the report contains the screenshots too for a better view of the images.

A word file has also been attached in the appendix containing all the layouts.

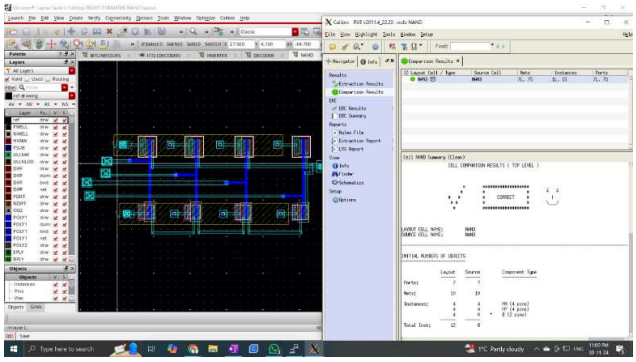

Fig 17: LVS NAND Gate

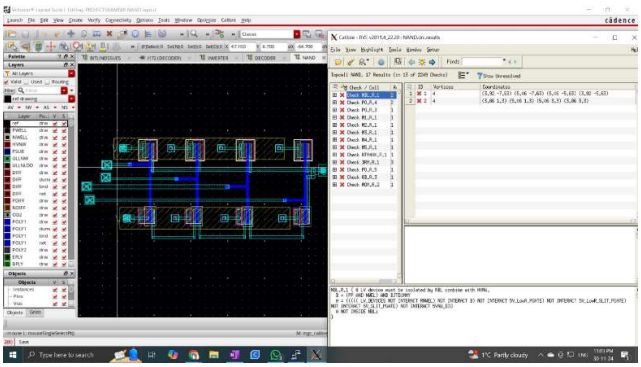

Fig 18: DRC NAND Gate

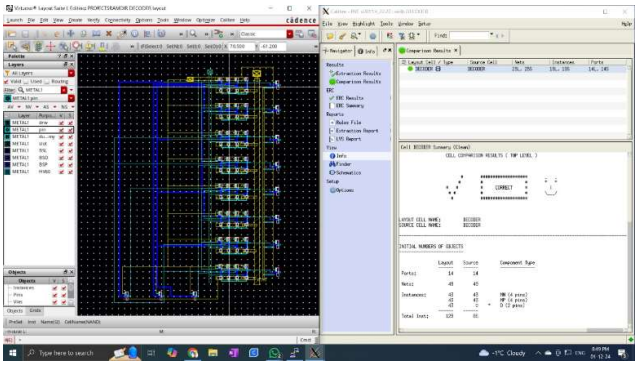

Fig 19: LVS DECODER

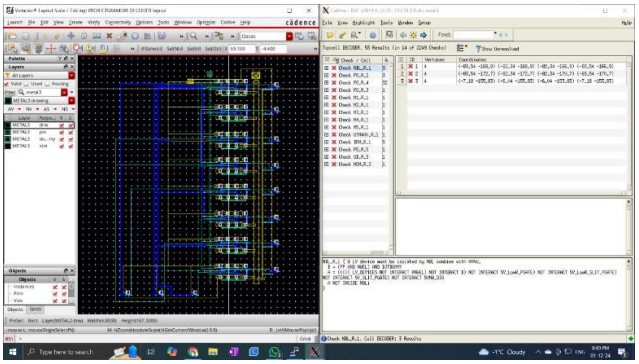

Fig 20: DRC DECODER

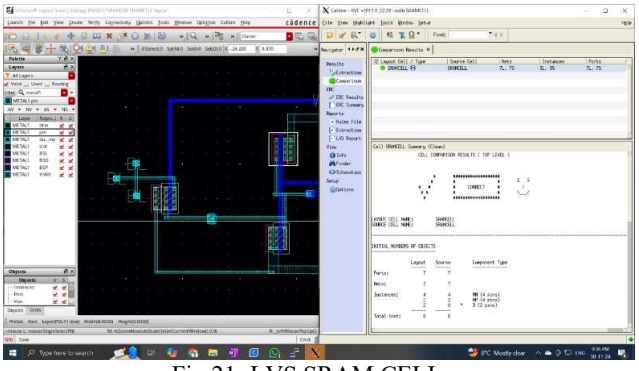

Fig 21: LVS SRAM CELL

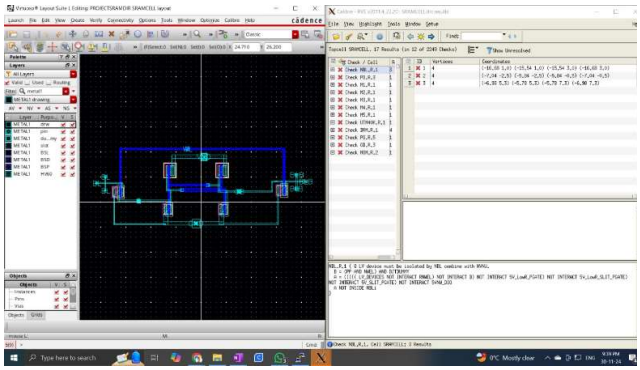

Fig 22: DRC SRAM Cell

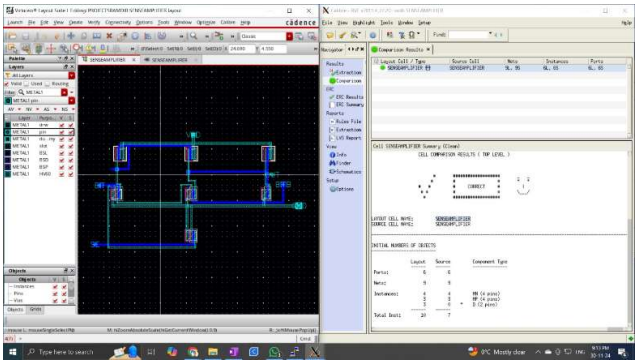

Fig 23: LVS Sense Amplifier

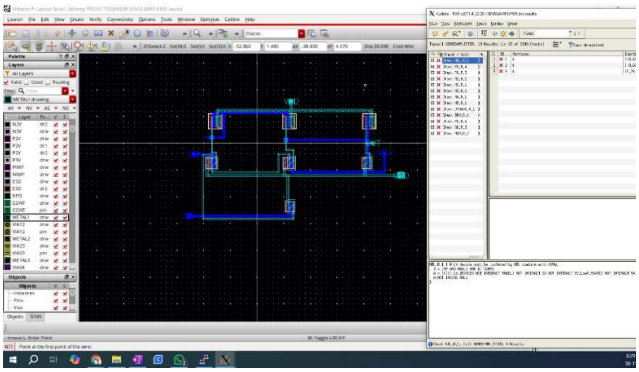

Fig 24: DRC Sense Amplifier

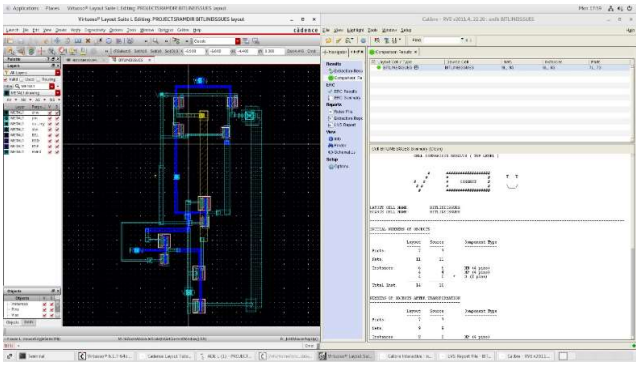

Fig 25: LVS Bitline Drivers

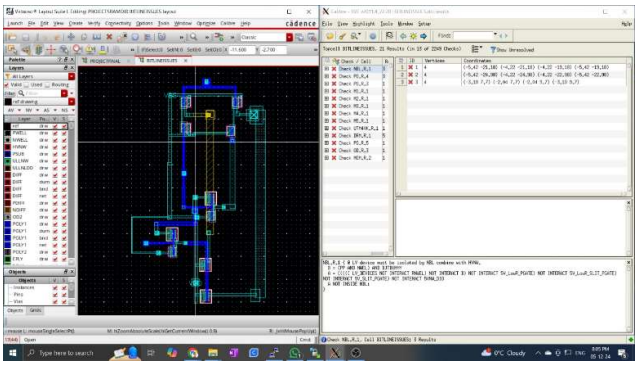

Fig 26: DRC Bitline Drivers

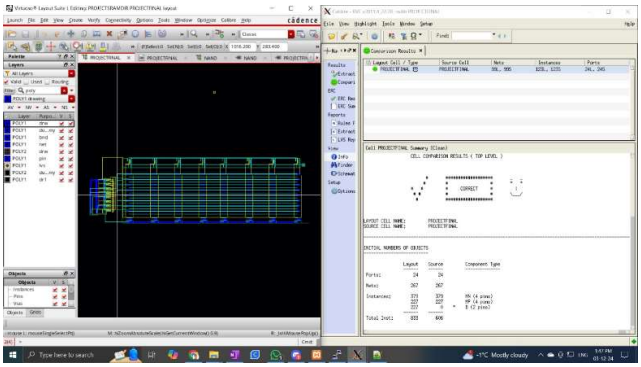

Fig 27: LVS Final Layout

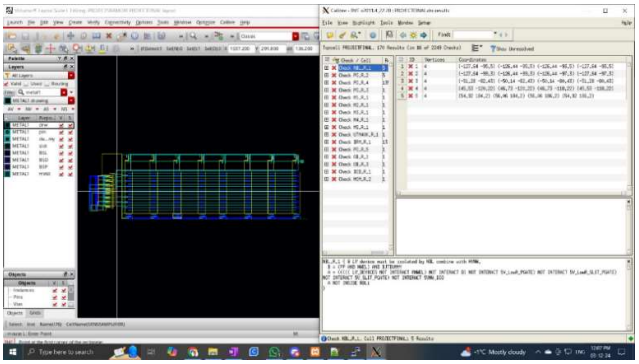

Fig 28: DRC Final Layout

VI. SIMULATIONS

Below are some simulations performed on the circuit with their results given.

A. General Simulations:

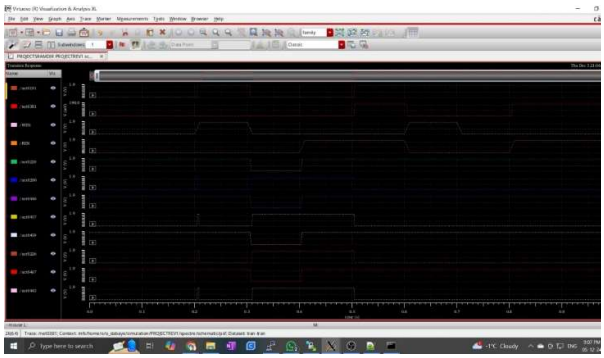

Fig 29: Typical Typical Simulation

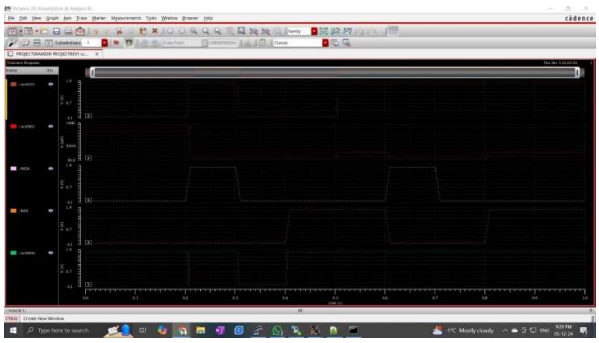

Fig 30: Fast Fast Simulation

B. Read Time:

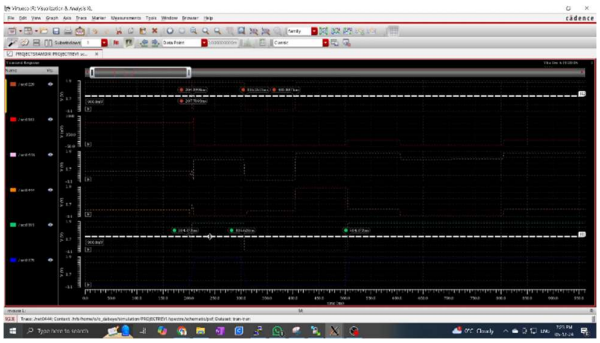

Fig 31: Read Delay Simulation

Brown wire: Output

Red wire: Q

Pink wire: BLB

Orange wire: BL

Green wire: Word Line

Blue wire: Decoder Enable

A video is attached in the appendix explaining the results for a better understanding of the simulation.

C. Write time:

Fig 32: Write Delay Simulation

A video is attached in the appendix explaining the results for a better understanding of the simulation.

D. Parasitic Capacitance:

The code below was used to generate the parasitic capacitance for each node.

Code:

```
# Path to the PEX file
pex_file = '/mnt/data/PROJECTFINAL.pex.txt'


# Regex to match capacitance entries
cap_regex = r"mr_pp\s+'c\s+\"(\S+)\"\s+'\((\".+?\" \"?.*?\")\)\s+([\d.eE+-]+)f"


# Store extracted capacitance data
capacitance_data = []


with open(pex_file, 'r') as file:
    for line in file:
        match = re.match(cap_regex, line)
        if match:
            component = match.group(1)
            nodes = match.group(2).strip('()').split(' ')
            value = float(match.group(3)) * 1e-15  # Convert femtofarads to farads
            capacitance_data.append({'component': component, 'nodes': nodes, 'value': value})


# Display extracted capacitances
print(f"Extracted {len(capacitance_data)} capacitances:")
for cap in capacitance_data[:10]:  # Show the first 10 entries
    print(f"Component: {cap['component']}, Nodes: {cap['nodes']}, Value: {cap['value']} F")
```

Result:

Extracted 21060 capacitances:

Component: ciXI196/NET17_34, Nodes: [‘”c_4_n”’, ‘”0”’], Value: 4.7541e-17 F

Component: ciXI196/NET17_35, Nodes: [‘”c_3_n”’, ‘”0”’], Value: 6.492350000000001e-16 F

Component: ciXI196/NET17_36, Nodes: [‘”I196/MM8_g”’, ‘”0”’], Value: 2.8134800000000005e-17 F

Component: ciXI196/NET17_37, Nodes: [‘”I196/NET17_68”’, ‘”0”’], Value: 3.5797500000000004e-17 F

Component: ciXI196/NET17_38, Nodes: [‘”I196/NET17_67”’, ‘”0”’], Value: 5.12249e-16 F

Component: ciXI196/NET17_39, Nodes: [‘”I196/NET17_66”’, ‘”0”’], Value: 3.5797500000000004e-17 F

Component: ciXI196/NET17_40, Nodes: [‘”c_10_n”’, ‘”0”’], Value: 3.52253e-16 F

Component: ciXI196/NET17_41, Nodes: [‘”c_5_n”’, ‘”0”’], Value: 4.1651500000000004e-17 F

Component: ciXI196/NET17_42, Nodes: [‘”c_30_n”’, ‘”0”’], Value: 4.017020000000001e-17 F

Component: ciXI196/NET17_43, Nodes: [‘”c_29_n”’, ‘”0”’], Value: 1.1852800000000002e-16 F

Key Components

1. Component Name (ciXI196/NET17_xx):
    - This is a unique identifier for each extracted capacitance. The prefix (ciXI196) might represent a hierarchical block in your design (e.g., a submodule or instance).
    - NET17_xx specifies the specific net associated with the parasitic capacitance.
2. Nodes:
    - The parasitic capacitance is extracted between the listed nodes.
    - Example: [‘”c_4_n”’, ‘”0”’] means the capacitance exists between node c_4_n and the ground node (denoted as “0”).
3. Value:
    - The capacitance value in Farads.
    - Example: 4.7541e-17 F (47.541 attofarads, a very small capacitance).

Interpretation of Each Entry

10. ciXI196/NET17_34

- Nodes: c_4_n to 0 (ground).
- Value: 4.7541e-17 F (very small, likely due to a small parasitic path, e.g., fringe capacitance).

11. ciXI196/NET17_35

- Nodes: c_3_n to 0 (ground).
- Value: 6.492350000000001e-16 F (significantly larger, could be due to a wider or longer interconnect).

12. ciXI196/NET17_36

- Nodes: I196/MM8_g to 0 (ground).
- Value: 2.8134800000000005e-17 F (small gate parasitic capacitance on MM8 transistor).

13. ciXI196/NET17_37

- Nodes: I196/NET17_68 to 0 (ground).
- Value: 3.5797500000000004e-17 F (small capacitance, likely due to a short wire or minimal coupling).

14. ciXI196/NET17_38

- Nodes: I196/NET17_67 to 0 (ground).
- Value: 5.12249e-16 F (larger capacitance, might indicate a significant overlap or coupling effect).

15. ciXI196/NET17_39

- Nodes: I196/NET17_66 to 0 (ground).
- Value: 3.5797500000000004e-17 F (similar to NET17_37).

16. ciXI196/NET17_40

- Nodes: c_10_n to 0 (ground).
- Value: 3.52253e-16 F (medium-size parasitic capacitance).

17. ciXI196/NET17_41

- Nodes: c_5_n to 0 (ground).
- Value: 4.1651500000000004e-17 F.

9. ciXI196/NET17_42

- Nodes: c_30_n to 0 (ground).
- Value: 4.017020000000001e-17 F.

10. ciXI196/NET17_43

- Nodes: c_29_n to 0 (ground).
- Value: 1.1852800000000002e-16 F (larger than most, might indicate significant coupling or area overlap).

General Observations

1. Values and Scale:
    - The capacitances are on the order of attofarads (10^{-18} F) to femtofarads (10^{-15} F), which is typical for parasitic capacitances in modern nanometer-scale designs.
2. Dominant Contributions:
    - Larger capacitance values (e.g., 6.49235e-16 F) often arise from:
        - Wider or longer interconnects.

- Coupling between closely spaced conductors.
- Overlap of interconnect layers.

3. Small Values:
    - Smaller capacitances (e.g., 2.81348e-17 F) might be due to fringe effects or minimal coupling.
4. Hierarchy and Nodes:
    - The node names (e.g., I196/MM8_g) indicate parasitics related to specific parts of the circuit, such as transistor gates (_g) or interconnect nets.

E. Area of the circuitry:

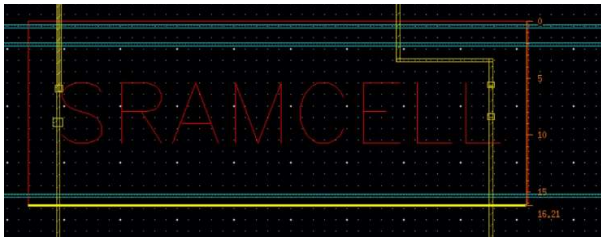

Fig 33: Length of a SRAM Cell

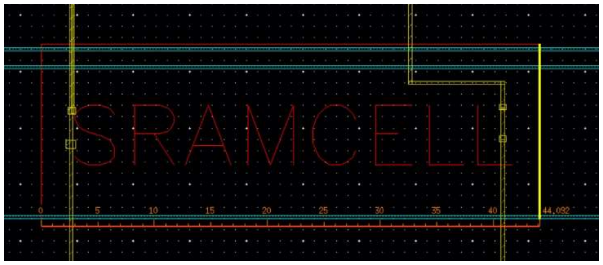

Fig 34: Width of a SRAM Cell

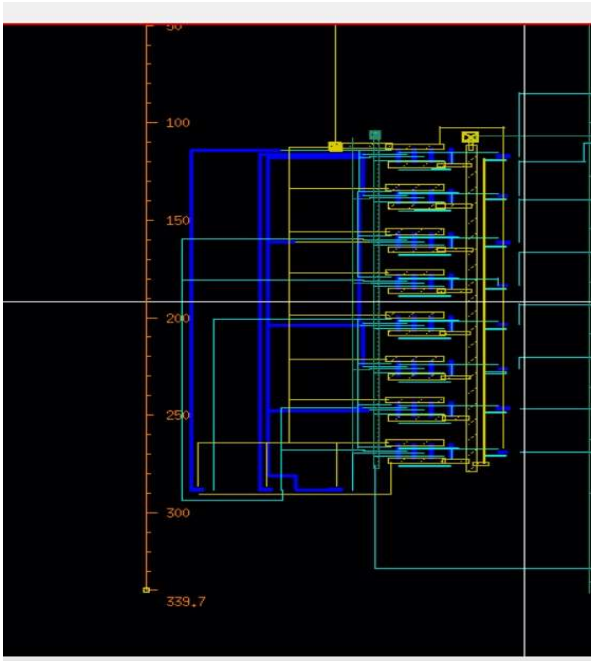

Fig 35: Length of Final Circuit

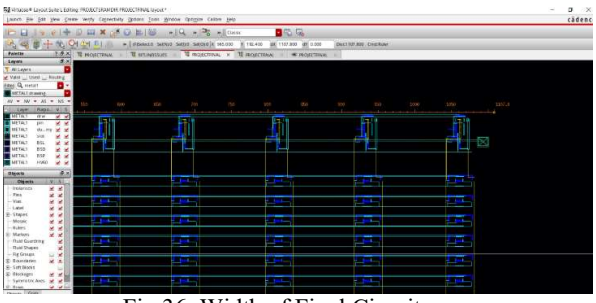

Fig 36: Width of Final Circuit

The total area is calculated by:

$$\text{Total Area} = 339.7 \times 1107.8 = 376319.66 \text{ units}^2$$

VII. Conclusions

This project provided valuable insights into the design and implementation of an 8×8 SRAM array using TSMC 180nm CMOS technology. Here are the key takeaways:

A. Lessons Learned

Through the design and layout process, we gained a deeper understanding of:

- The importance of parasitic effects (capacitances and resistances) in high-performance memory designs.
- The critical role of layout precision and hierarchy in achieving efficient, manufacturable designs.
- Optimizing circuit performance by balancing layout constraints, circuit sizing, and timing.

We also experienced firsthand how pitch-matching wordline and bitline circuitry to the SRAM array grid ensures both uniformity and efficiency in the design process. This provided excellent practice in layout and reinforced the

importance of adhering to standard design practices in memory arrays.

B. Design Decisions

We made several strategic decisions to ensure functionality, efficiency, and simplicity:

- Prioritizing Pitch-Matching: Wordline and bitline circuitry were laid out to be pitch-matched to the SRAM array. This choice was made to simplify routing, maintain layout uniformity, and align with industry practices for memory design.
- Parasitic Mitigation: High-parasitic nodes were identified during extraction, and their impact was minimized by carefully managing wire lengths and spacing in critical areas.
- Time Allocation: Due to time constraints, we focused on completing the layout and ensuring LVS/DRC compliance rather than extensive sizing optimization.

C. Handling Long Wires and Resistive Loads

The 1mm wire in the circuit posed challenges related to parasitic capacitances and resistive loading:

- Mitigation Approach: Parasitic effects were analyzed, and wire resistance was factored into simulations to ensure timing and signal integrity.
- Wire Design: The wire was carefully routed and optimized within the layout to minimize resistance and capacitance while maintaining layout constraints.

D. Sizing Choices

- SRAM Cell Sizing: The standard 6T SRAM cell design was used with transistor sizes chosen to balance speed, stability, and power consumption. The sizing met functional requirements during simulations without requiring further tuning.
- Peripheral Circuitry Sizing: Components like the wordline decoder, sense amplifiers, and bitline drivers were sized to ensure proper signal propagation and accurate read/write operations.
- Focus on Stability: The sizing decisions prioritized stability and robustness over aggressive optimization to ensure reliable operation.

E. Layout Choices

- Pitch-Matching: The wordline and bitline circuitry were laid out to align with the SRAM array pitch. This approach simplified routing, reduced area waste, and ensured alignment with memory cell rows and columns, enhancing manufacturability.
- Hierarchical Design: The design followed a hierarchical approach, where sub-blocks like decoders, sense amplifiers, and bitline drivers were individually laid out and integrated into the final array.
- Minimizing Parasitics: Layout decisions emphasized minimizing parasitics by reducing interconnect lengths and using efficient routing strategies in critical paths.

In conclusion, this project provided valuable experience in SRAM design, from schematic-level decisions to layout and parasitic analysis. The use of pitch-matched layout for wordline and bitline circuitry not only improved integration but also gave us practical experience in precision layout design. Despite challenges like handling the 1mm wire, the final design met functional requirements, demonstrating a balance between performance, layout efficiency, and manufacturability.

REFERENCES

[1] *S. S. Kadam and P. Bhatasana, "Design of 16-bit Low Power SRAM in 180nm CMOS Technology," Journal of Current Research in Engineering and Science, vol. 2, no. 1, pp. 55-60, 2022.*

[2] R. K. Sah, I. Hussain, and M. Kumar, "Performance Analysis of a 6T SRAM Cell in 180nm CMOS Technology," *IOSR Journal of VLSI and Signal Processing*, vol. 5, no. 2, pp. 20-22, 2015.

[3] P. Athe and S. Dasgupta, "A Comparative Study of 6T, 8T and 9T Decanano SRAM Cell," in *Proceedings of the IEEE Symposium on Industrial Electronics and Applications*, Kuala Lumpur, Malaysia, 2009, pp. 889-894.

[4] F. Moradi and J. K. Madsen, "Improved Read and Write Margins Using a Novel 8T-SRAM Cell," in *Proceedings of the IEEE/IFIP International Conference on VLSI and System-on-Chip*, Santa Cruz, CA, USA, 2012, pp. 1-4.

[5] D. Aggarwal, P. Kaushik, and N. Gujran, "A Comparative Study of 6T, 8T and 9T SRAM Cell," *International Journal of Latest Trends in Engineering and Technology*, vol. 1, no. 1, pp. 44-52, 2012.