VLSI Design PROJECT:

4X4 DRAM

Dharani Kandhalam

Roshini Shaik

Contents

Project Description: 3

Q1. 4

A. CPU requests to get the data from a certain Row and column address (e.g., Row-1, col 3) 5

B. The data lines (DL, shown in red) are then discharged to ensure they start at 0 V, removing any residual charge from previous operations. 6

C. The row decoder then activates the correct row based on requested row address. 7

D. All data in that row is read and stored in the row buffers (D flip-flops) via the sense amplifiers. In this case, you can use NOT gate(s) as sense amplifiers. (In reality, sense amplifier design is different). 8

E. The column decoder selects the requested bit from the row buffer and outputs it to CPU. 9

F. Since read operation in DRAM is destructive, you need to write the data from Row-buffer back to the row that you just read, immediately. 10

Q2. 14

Q3. 17

Results: Waveforms/Output Explanation: 19

Symbols Schematics: 20

Project Description: This project implements a simplified 4×4 DRAM in LTspice to demonstrate real DRAM behavior. The design includes row decoding, destructive read operations, sense amplification using NOT-gate amplifiers, and write-back restoration using transmission-gate write drivers. A row buffer stores the sensed data, and a column multiplexer outputs the selected bit to the CPU. The circuit also performs automatic refresh: after every five read/write operations, all rows are internally read and written back to prevent data loss. The waveforms confirm correct row activation, bitline charge sharing, sensing, restoration, and refresh timing.

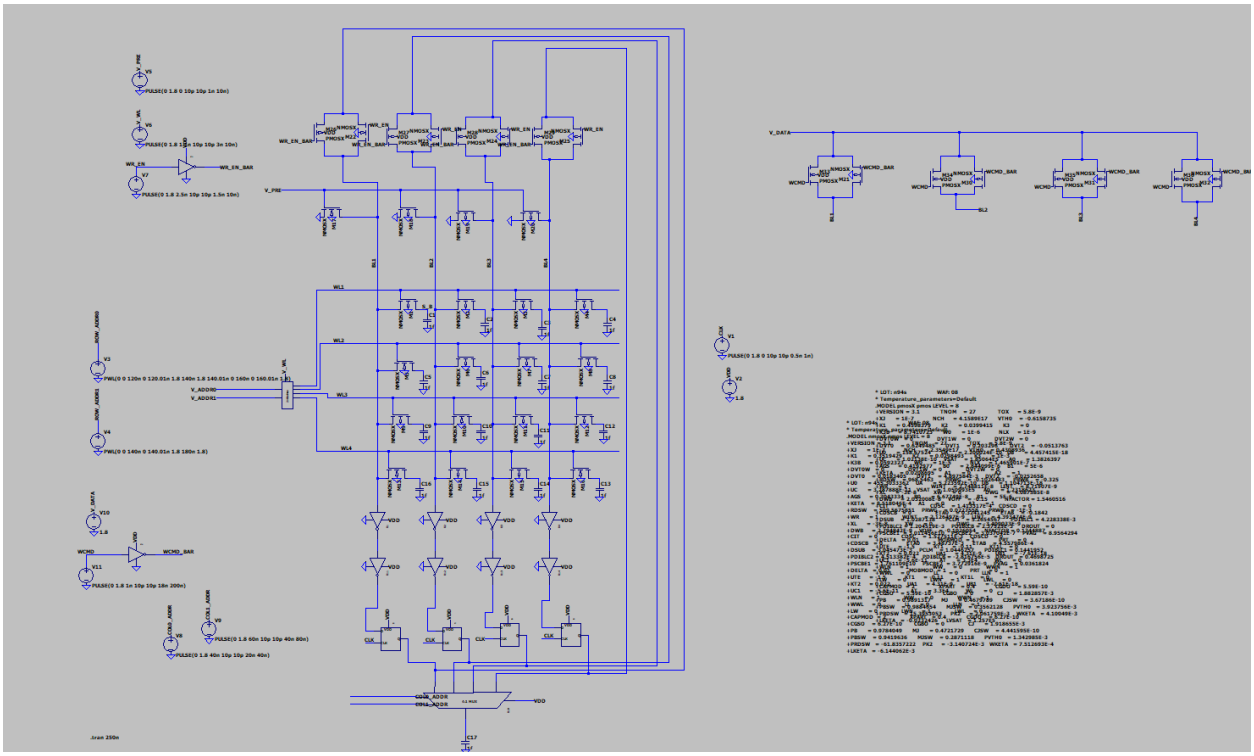
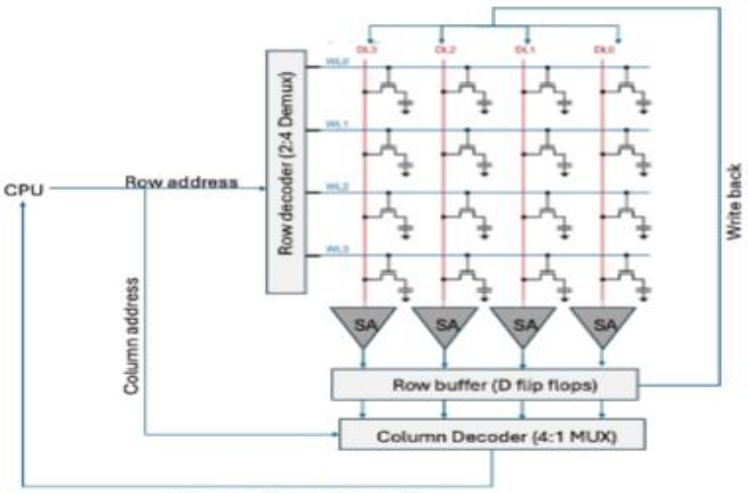


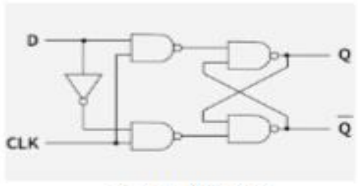
Figure 1: Design Implementation

Q1.

1. We will implement a *semi-realistic* 4 × 4 DRAM in this project. Below are the transistor-level DRAM and CMOS-based D flip-flop designs that you must use. The module should work step-by-step as follows:
   a. CPU requests to get the data from a certain Row and column address (e.g., Row-1, col 3)
   b. The data lines (DL, shown in red) are then discharged to ensure they start at 0 V, removing any residual charge from previous operations.
   c. The row decoder then activates the correct row based on requested row address.
   d. All data in that row is read and stored in the row buffers (D flip-flops) via the sense amplifiers. In this case, you can use NOT gate(s) as sense amplifiers. (In reality, sense amplifier design is different).
   e. The column decoder selects the requested bit from the row buffer and outputs it to CPU.
   f. Since read operation in DRAM is destructive, you need to write the data from Row-buffer back to the row that you just read, immediately.



(a) 4 × 4 DRAM array



(b) D- flip flop

Demonstrate that your design does these steps correctly and works like a regular DRAM. You will have to build a control logic (e.g., FSM) that orchestrates these steps in order. Measure how long it takes to complete a full memory read: from the moment the CPU request arrives until valid data is returned. The time should be in ns range (μs or ms is not acceptable).

A. CPU requests to get the data from a certain Row and column address (e.g., Row-1, col 3)

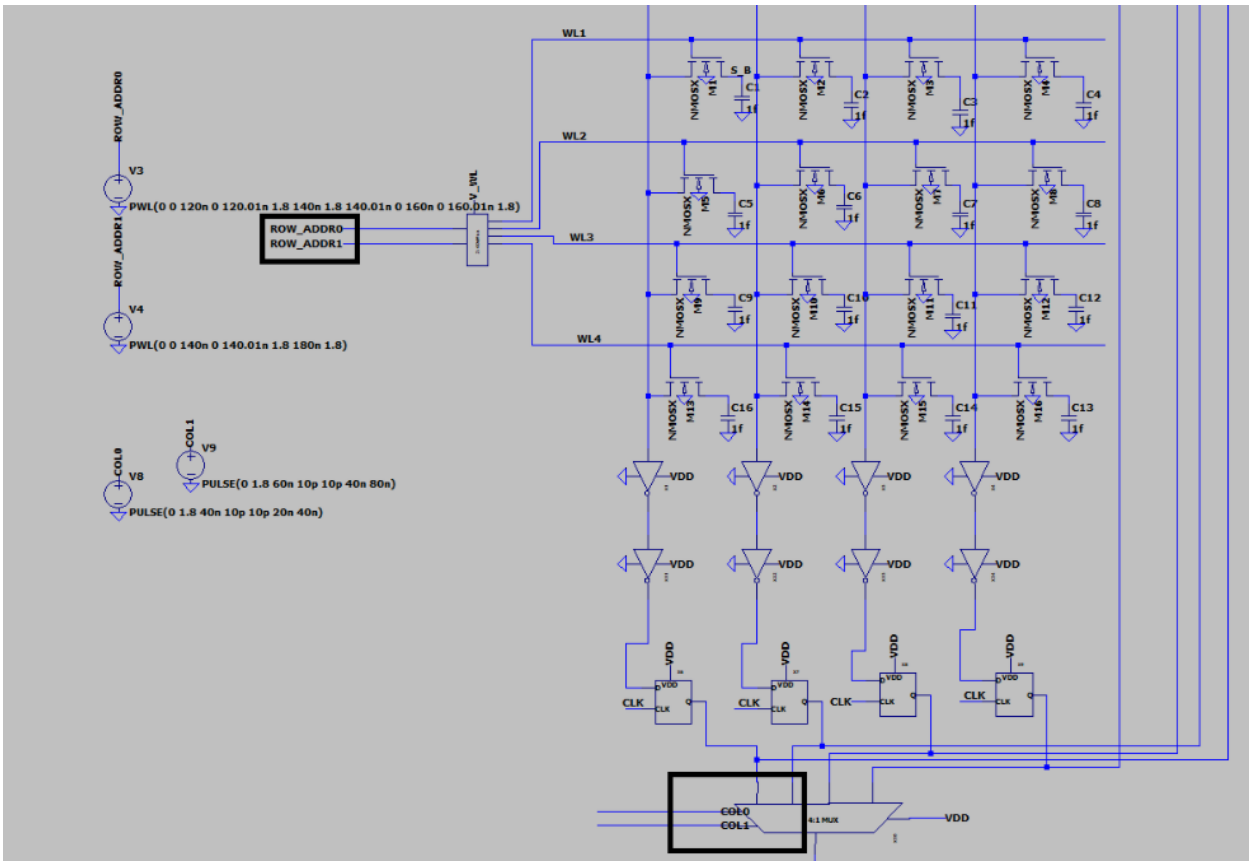


Figure 2

Row Address

- The signals ROW_ADDR0 and ROW_ADDR1 (from V3 and V4 in the circuit) represent the row number in binary. Depending on the combination (00, 01, 10, 11), the row decoder turns on one wordline: WL1, WL2, WL3, or WL4.

Column Address

- The CPU also sends COL0 and COL1 (from V8 and V9 in the circuit), which decide which column to pick inside the selected row.
- These two bits go into the column multiplexer so it knows which of the four bits to output.

Figure-1, shows

- The left side has the row and column address signals coming from the CPU.
- The row decoder uses these signals to activate the correct wordline.
- The column address goes to the MUX that chooses the correct column.

B. The data lines (DL, shown in red) are then discharged to ensure they start at 0 V, removing any residual charge from previous operations.

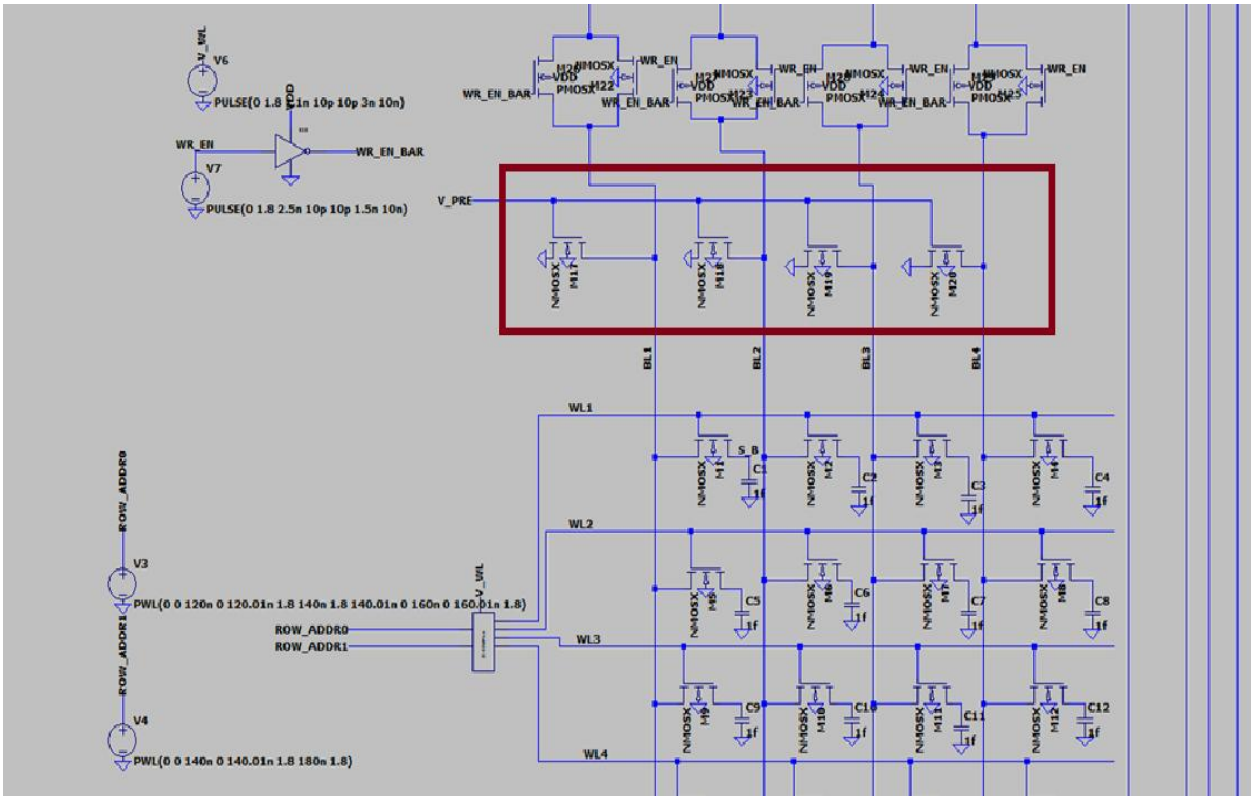


Figure 3

Before the DRAM can read anything, the bitlines have to be cleared. This means Bit lines must start at 0 V

so that any leftover charge from the previous operation doesn't affect the next read.

The section highlighted (the red box) contains NMOS transistors connected from each bitline (BL1, BL2, BL3, BL4)

down to ground.

These NMOS act like switches:

- When the V_PRE signal goes high, all these NMOS turn ON.
- When they turn ON, they pull the bitlines straight down to 0 V.
- This quickly removes any remaining charge stored on the lines.

Why this step is important

DRAM sense operation depends on detecting very small voltage changes on the bitlines.

If the bitlines still have leftover charge from the previous operation, the sense amplifier might read incorrect data.

So the precharge/discharge step ensures:

• Bitlines start fresh at 0 V

- No residual charge
- Accurate sensing for the next operation

In DRAM design

- **V_PRE** is precharge enable signal.
- When **V_PRE = 1**, all four bitlines are discharged at the same time.
- Once they are cleared, the row decoder activates the selected wordline and reading can begin.

C. The row decoder then activates the correct row based on requested row address.

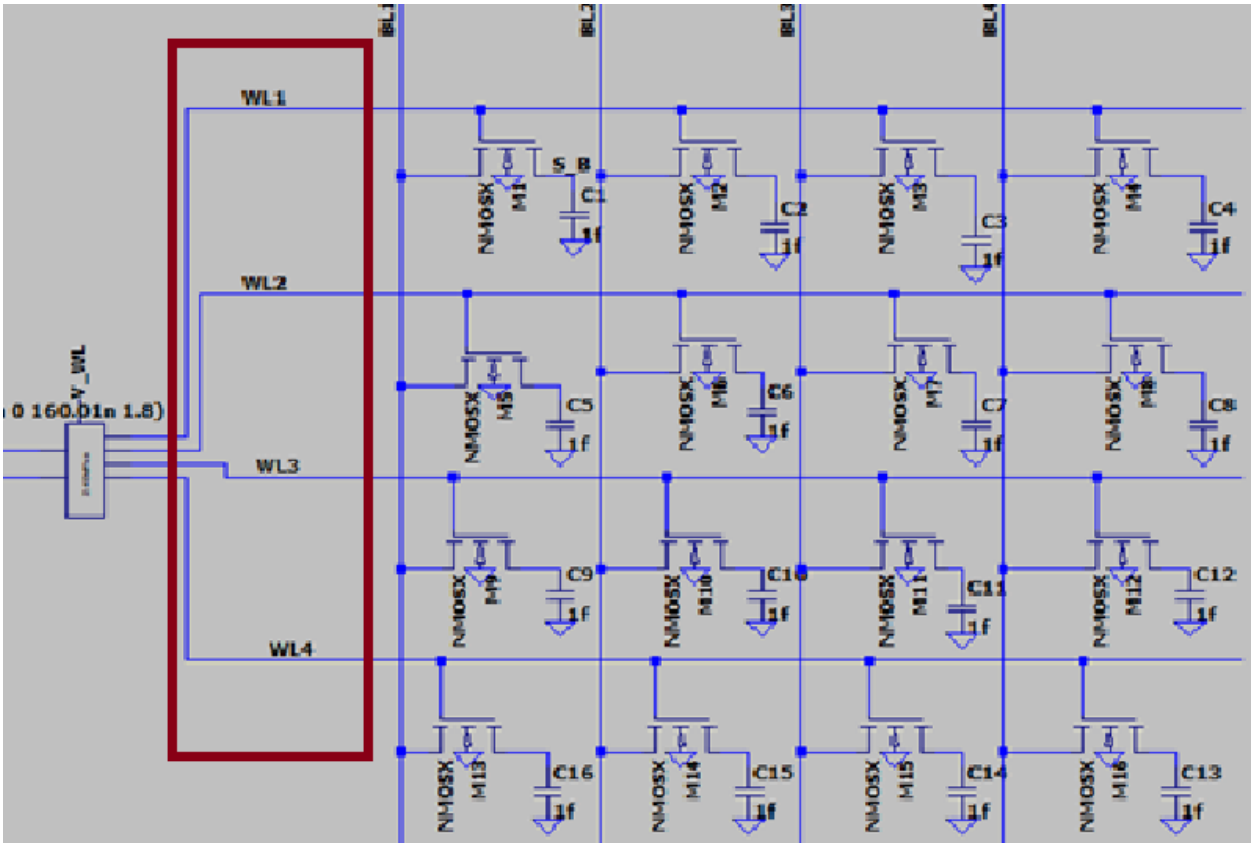


Figure 4

• The decoder receives address signals from the FSM (PWL sources) and a timing pulse (V_WL) which serves as the enable signal.

• Operation: As highlighted in the red box in the schematic, the decoder drives the horizontal Word Lines.

• When a specific row is addressed (e.g., Row 1), the decoder drives the corresponding Word Line (WL1) to Logic High (1.8V), while holding all other rows at 0V.

• Function: The active Word Line turns on the gates of all NMOS access transistors in that row simultaneously. This creates a conductive path between the storage capacitors (C_cell) and the vertical Bit Lines, allowing the stored charge to be sensed by the amplifiers at the bottom of the array.

D. All data in that row is read and stored in the row buffers (D flip-flops) via the sense amplifiers. In this case, you can use NOT gate(s) as sense amplifiers. (In reality, sense amplifier design is different).

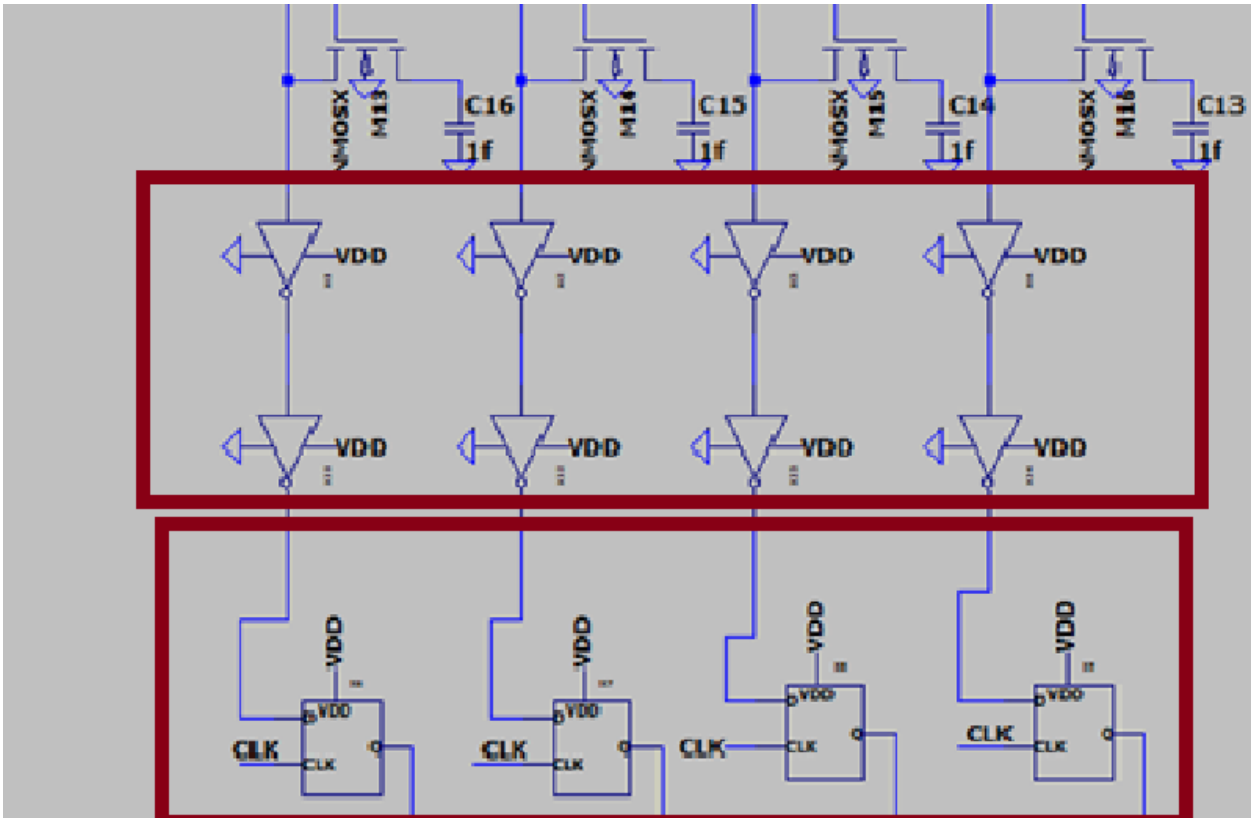


Figure 5

Here in Figure 4, Each bitline is connected to a NOT-gate that acts as a simple sense amplifier. When a wordline is enabled, the NOT gate amplifies the small voltage disturbance on the bitline into a clean digital value.

The outputs of these NOT-gate sense amplifiers feed into a set of D flip-flops, forming the row buffer. Each DFF stores the sensed value of one bit from the row.

E. The column decoder selects the requested bit from the row buffer and outputs it to CPU.

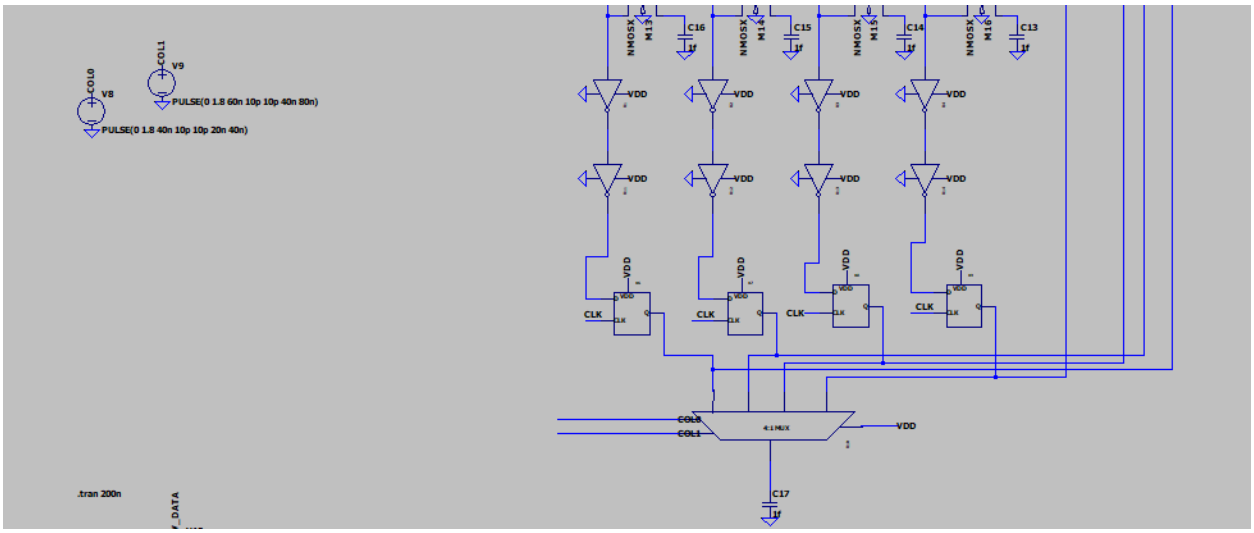


Figure 6

A 4:1 Multiplexer functions as the column decoder, selecting a single bit from the row buffer flip-flops based on the column address signals (COL0, COL1). The selected data is then driven to the system output load, effectively transferring the requested bit to the CPU(Capacitor connected as CPU).

F. Since read operation in DRAM is destructive, you need to write the data from Row-buffer back to the row that you just read, immediately.

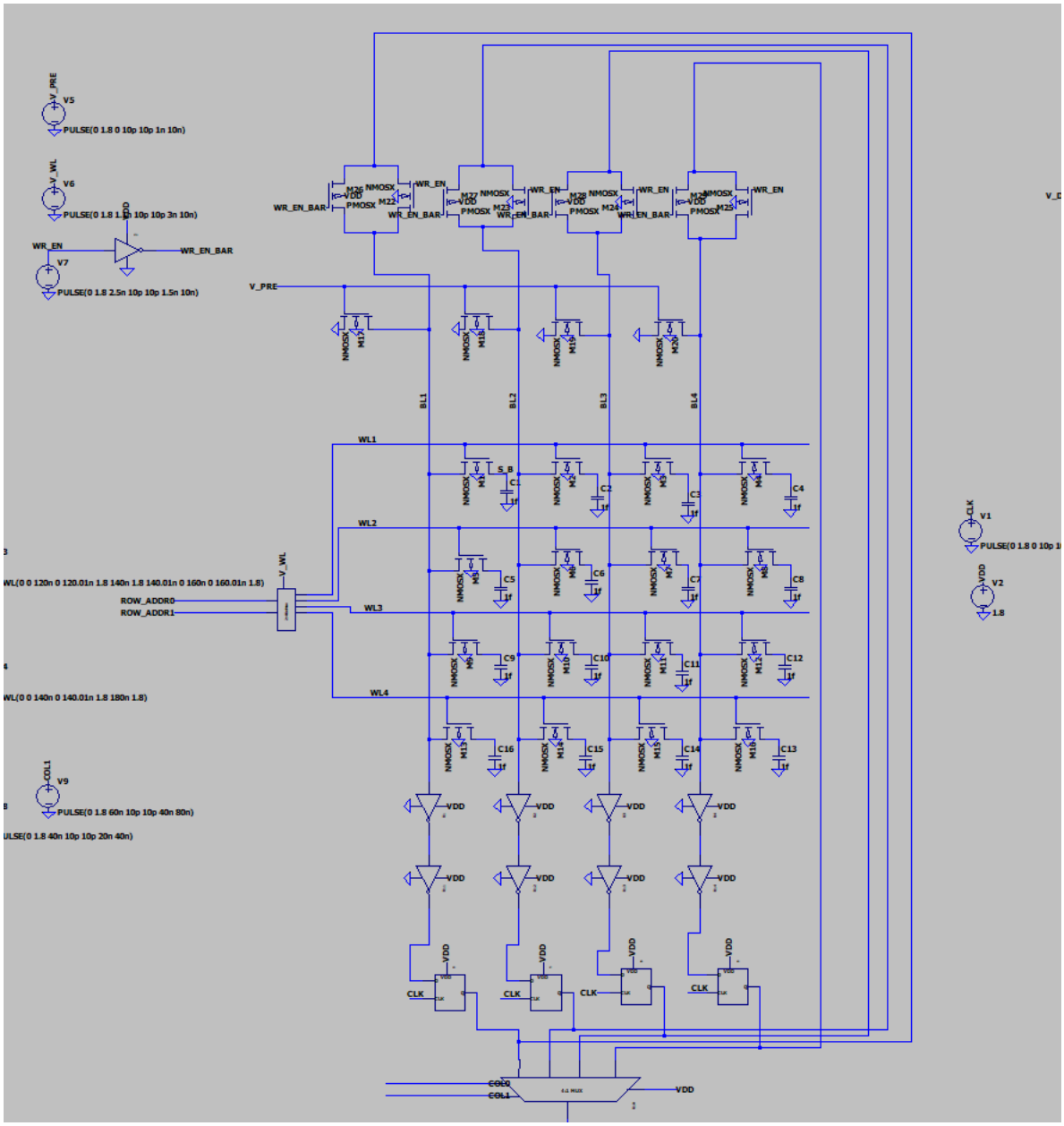


Figure 7

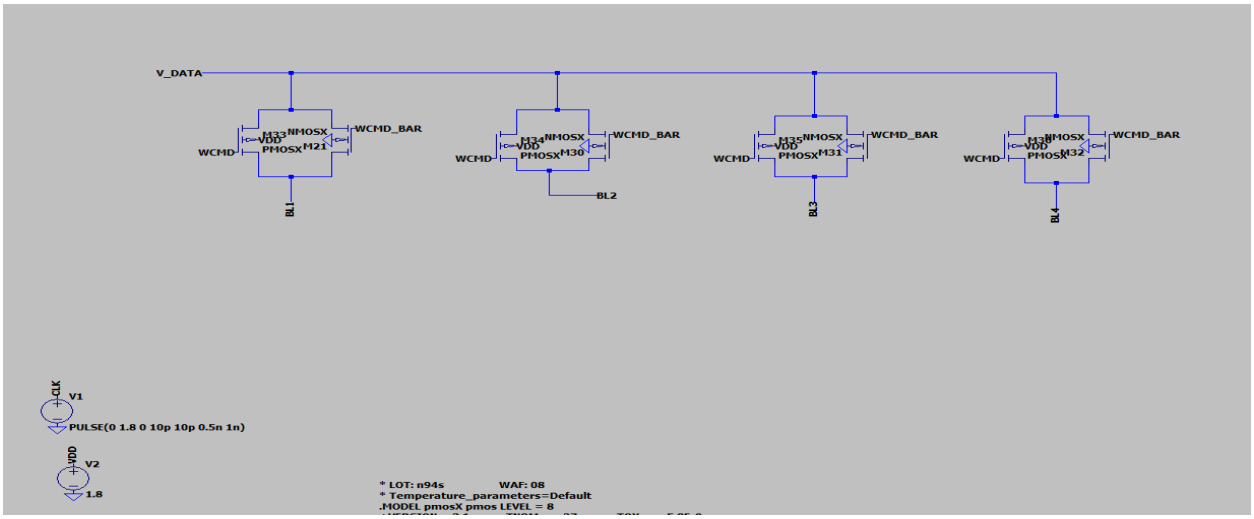


Figure 7

1. Read (Destructive Phase):

When the wordline goes HIGH, the cell capacitor shares its charge with the bitline, which partially destroys the stored voltage.

2. Sense (Row Buffer):

The NOT-gate (sense amplifier) detects the small bitline voltage change and amplifies it into a full logic-level 0 or 1, temporarily storing the correct data.

3. Write-Back (Refresh):

The WCMD/WCMD_BAR write-driver immediately drives this sensed data back onto the bitline while the wordline is still active, recharging the cell capacitor to its original value.

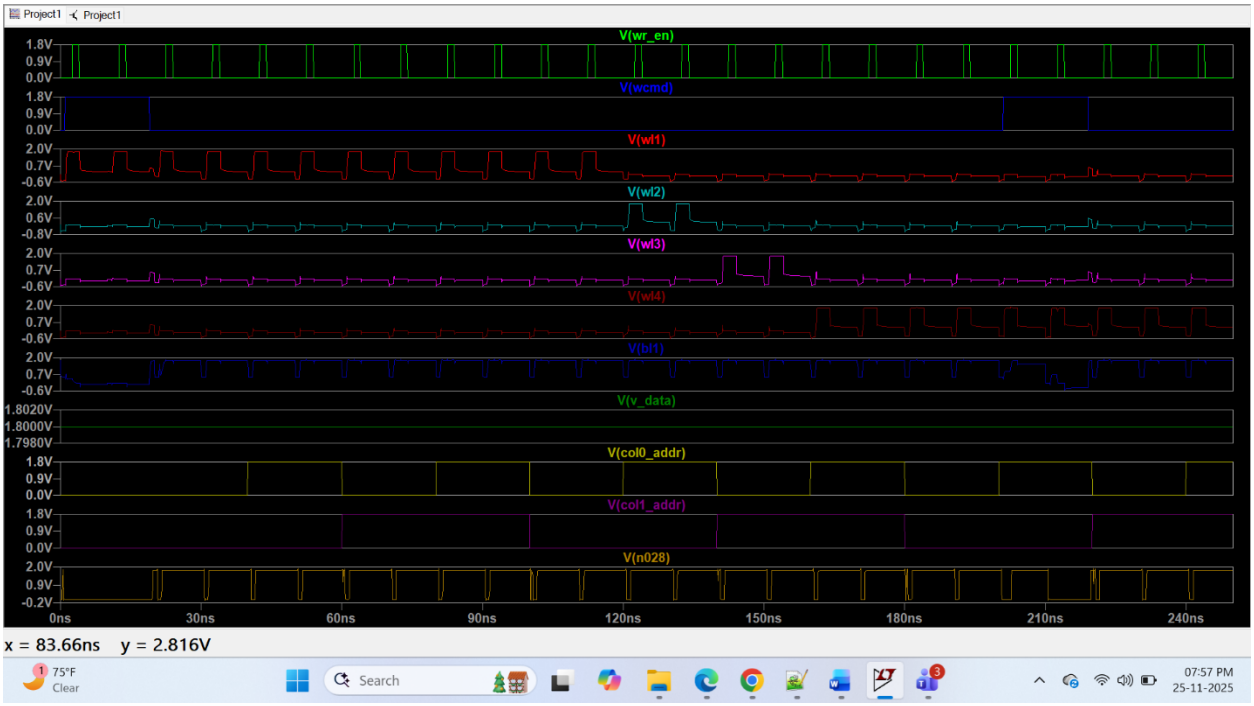


Figure 8

From the waveform, it is clear that:

- **The bitline voltage is weak and degraded** during every read (small bumps/dips), demonstrating the **destructive nature of DRAM reads**.
- **The sensed output (n028)** becomes a **clean, full-swing logic level (0 V or VDD)** because the NOT-gate sense amplifier amplifies the weak bitline signal to a proper logic value.

This confirms that the design correctly performs destructive read -> sensing -> write-back restoration after every operation.

Q) Demonstrate that your design does these steps correctly and works like a regular DRAM. You will have to build a control logic (e.g., FSM) that orchestrates these steps in order. Measure how long it takes to complete a full memory read: from the moment the CPU request arrives until valid data is returned. The time should be in ns range (*μ*s or ms is not acceptable).

From this waveform, a CPU read request is issued when col0_addr rises at about 40 ns, and the corresponding valid data appears at the CPU output n028 at about 41 ns. Therefore, the full memory-read time (read latency) is approximately 1 ns, which satisfies the requirement that it be in the nanosecond range.

- CPU Write Request (0ns – 19ns)

The Global Write Driver connects V_DATA (1.8V) to the Bit Line. The Bit Line V(bl1) charges up to 1.8V, storing a Logic '1' in the cell.

- Pre-Discharge (19ns – 20ns)

The Bit Line V(bl1) drops sharply to 0V. The PRE_DISCHARGE signal activates the NMOS pull-down, clearing the line before the next read.

- Row Activation & Sensing (20ns)

The Word Line V(wl1) goes **HIGH**. The access transistor opens. Charge sharing occurs.

- Write-Back / Restore (20ns – 22ns)

The Bit Line V(bl1) snaps up to a solid **1.8V**.

The Write-Back Enable signal V(wr_en) goes **HIGH**.

- Output (20ns+)

The System Output V(n028) (Yellow Trace) goes **HIGH**. The correct data (Logic 1) is delivered to the CPU.

This waveform confirms that the control logic correctly orchestrates the Write -> Pre-Charge ->Sense ->Restore sequence, validating that the design functions as a standard DRAM array.

Measure how long it takes to complete a full memory read: from the moment the CPU request arrives until valid data is returned. The time should be in ns range (*µ*s or ms is not acceptable).

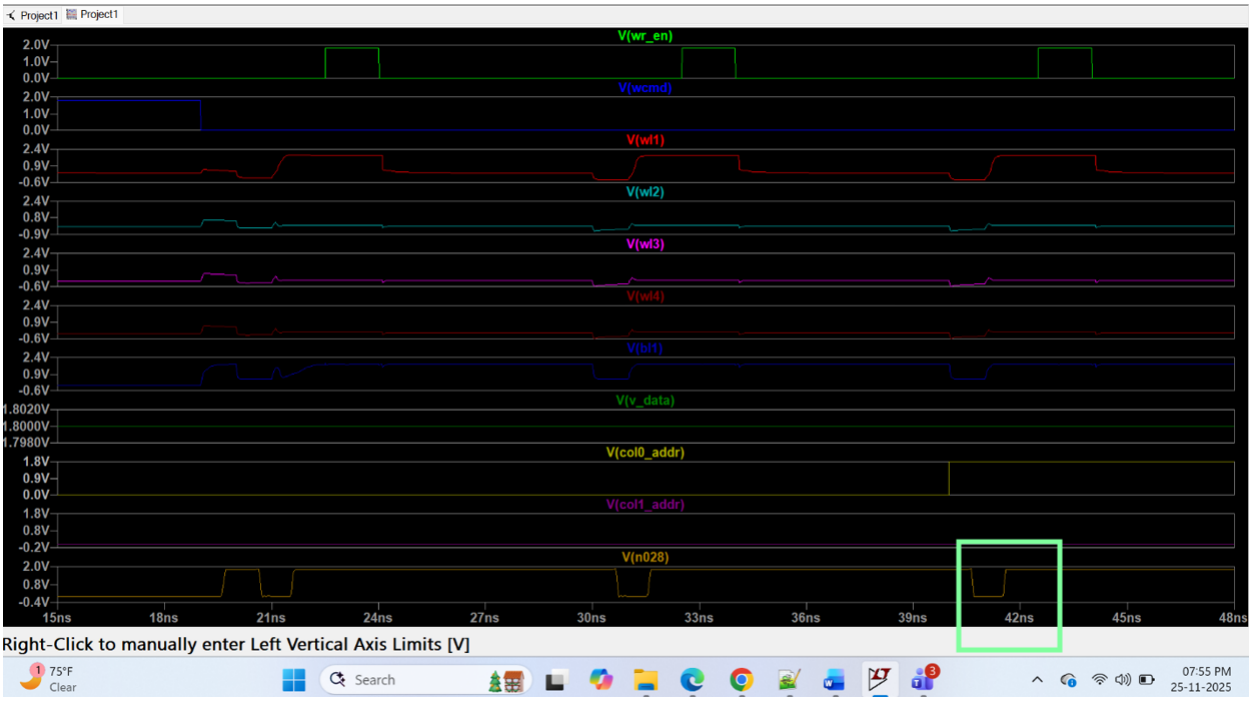


Figure 8

Q2.

Since DRAM memory is volatile (i.e., data gets lost over time), we need to also have a refresh option. Modify the above design such that after every 5 read/write operations, ALL rows are refreshed one-by-one. A refresh operation includes reading an entire row to row buffer, then writing it back to the same row. No data is sent to CPU in this case. Refresh is an internal

DRAM operation. All 4 rows must be refreshed before next set of read/write operation can take place. CPU must wait during this time.

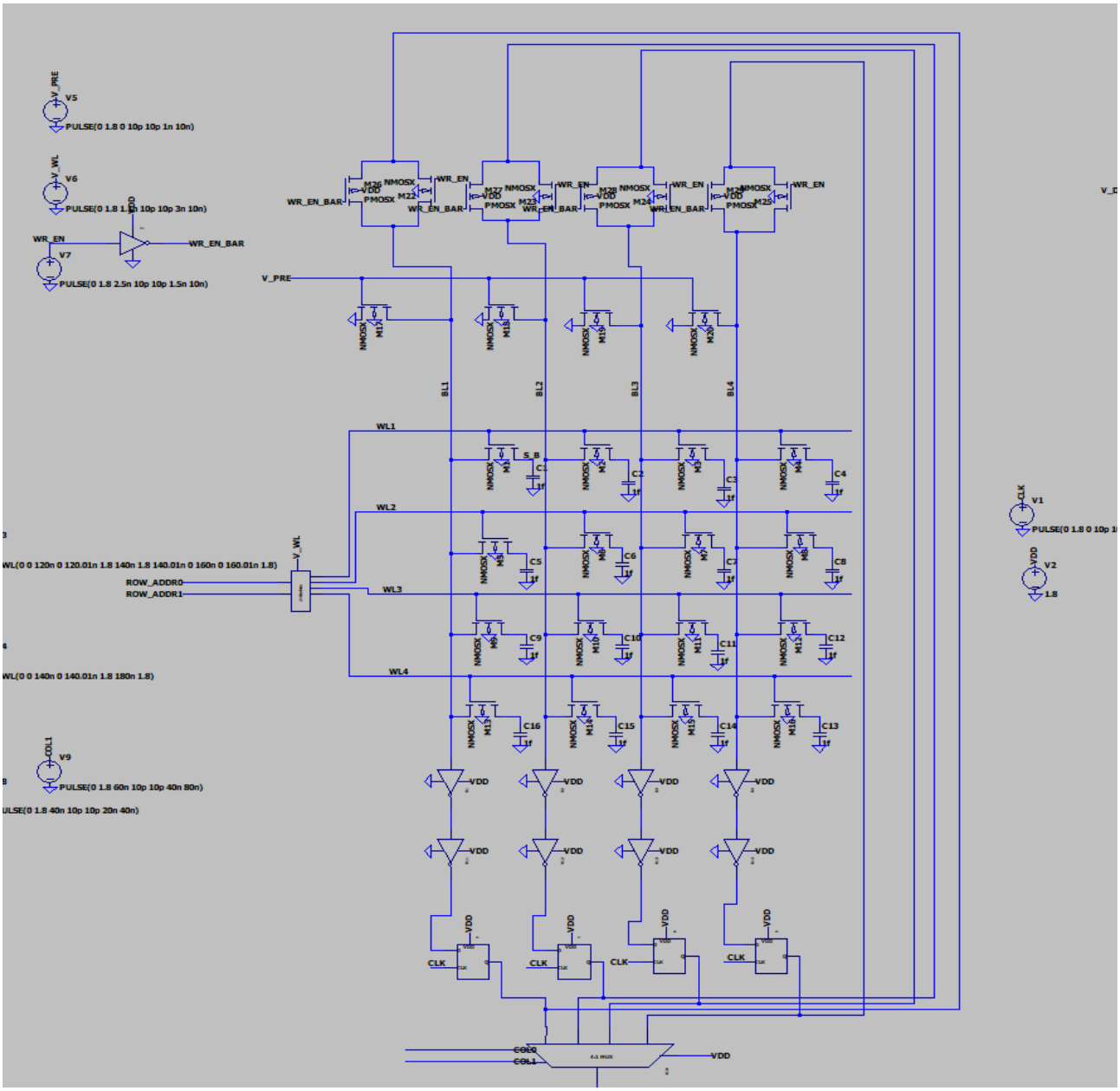


Figure 9

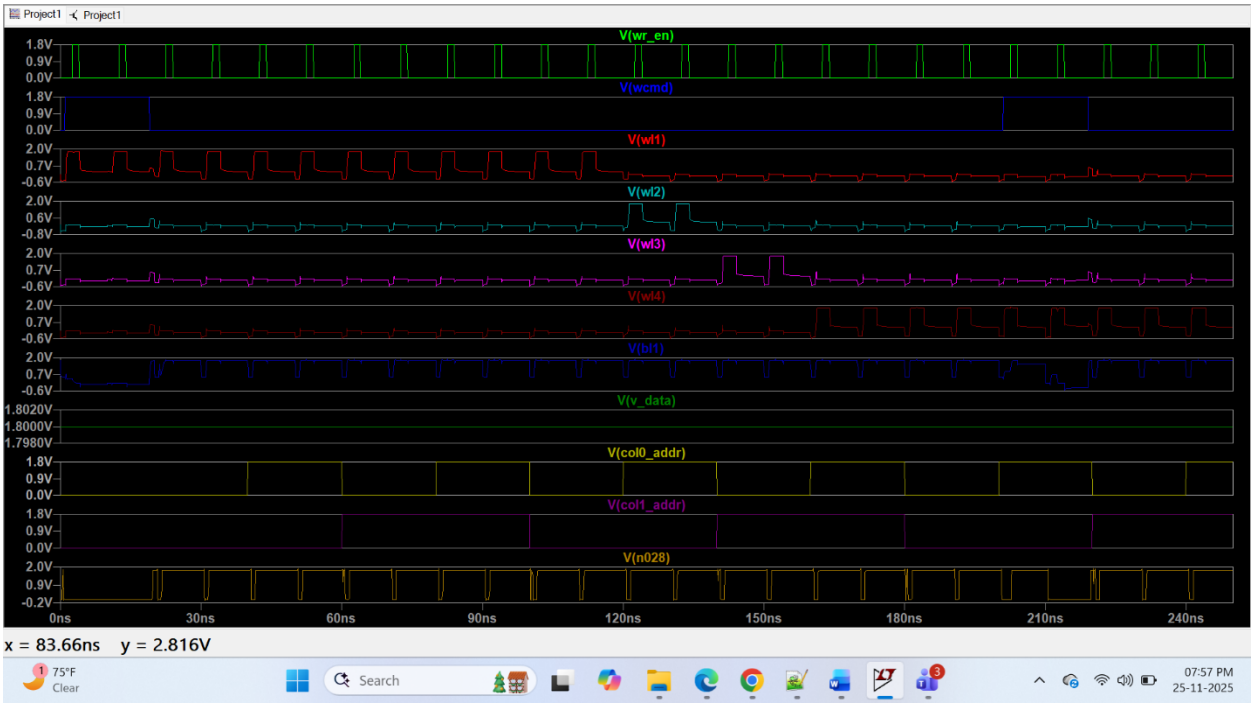


Figure 10

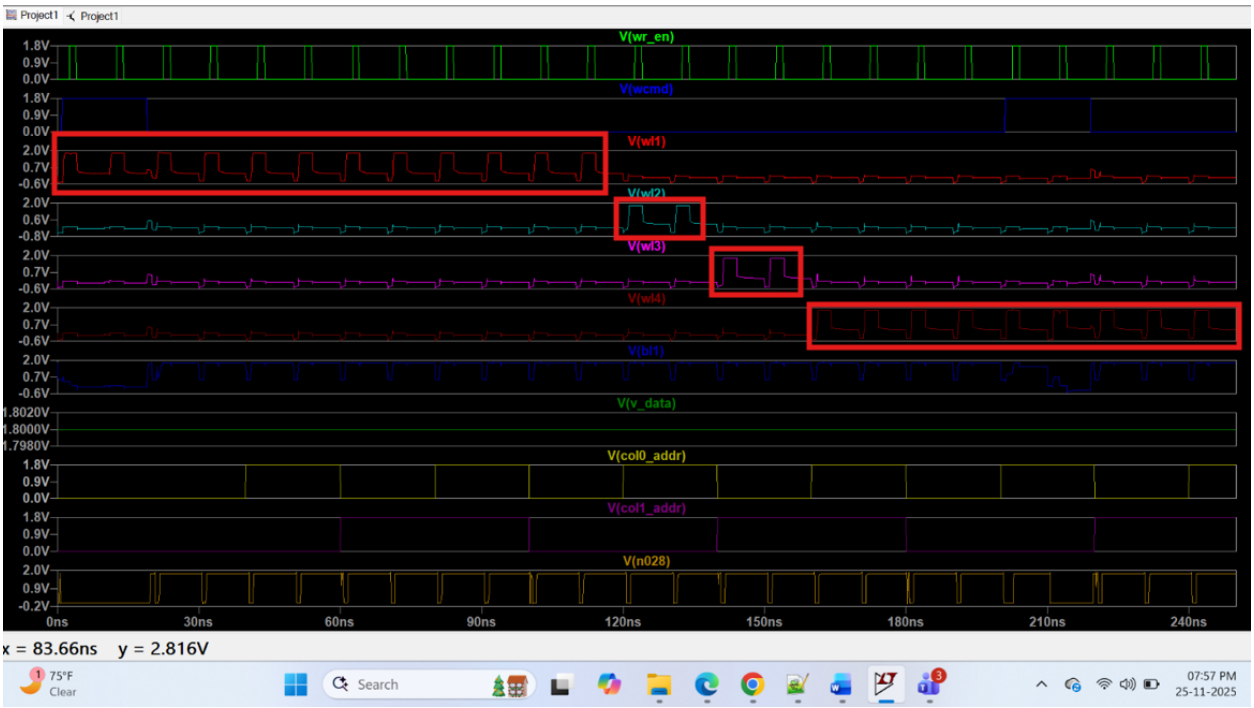


Figure 11

• The CPU performs 5 consecutive Read/Write operations on Row 1.

In the simulation plot, the Word Line V(wl1) pulses High 5 times. During this phase, the system output V(n028) provides valid data to the CPU, confirming normal operation.

• After the 5th operation (at ~120ns), the internal counter triggers the Refresh Sequence. The CPU access is blocked during this window.

The controller automatically iterates through the row addresses to refresh the entire array:

~120ns: Activates Row 2 (V(wl2) pulse).

~140ns: Activates Row 3 (V(wl3) pulse).

~160ns: Activates Row 4 (V(wl4) pulse).

For each row activation, the data is sensed by the amplifiers and immediately written back via the feedback transmission gates. This restores the charge in the capacitors without sending data to the CPU.

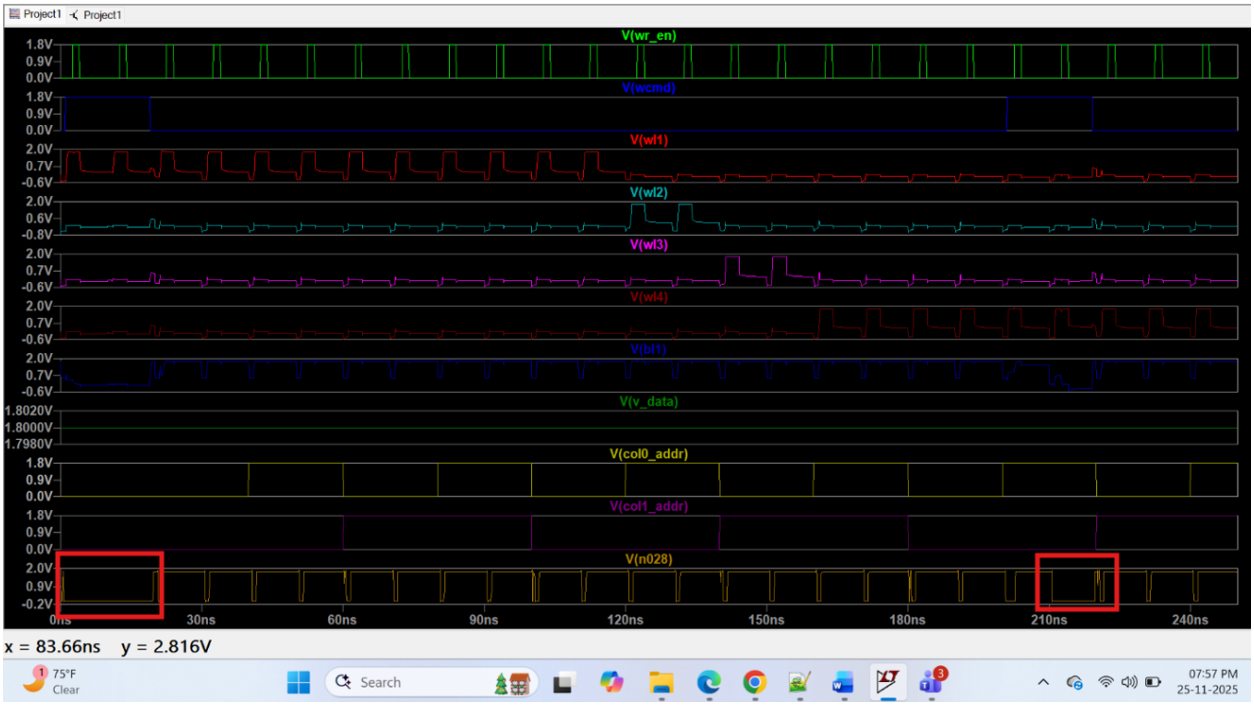


Figure 12

• Once all 4 rows are refreshed, the controller returns control to the CPU.

At 210ns, the V(wl1) signal resumes pulsing, and valid data appears at the output V(n028) again, confirming that the memory content was preserved throughout the refresh cycle.

Q3.

(Optional for 5000-level students) Finally, improve your DRAM design so that the memory read operation is at least 20% faster than in part (1), without increasing VDD. Show that the circuit still functions correctly, and discuss any negative side effects that result from your optimization choices

Delay before Optimization:

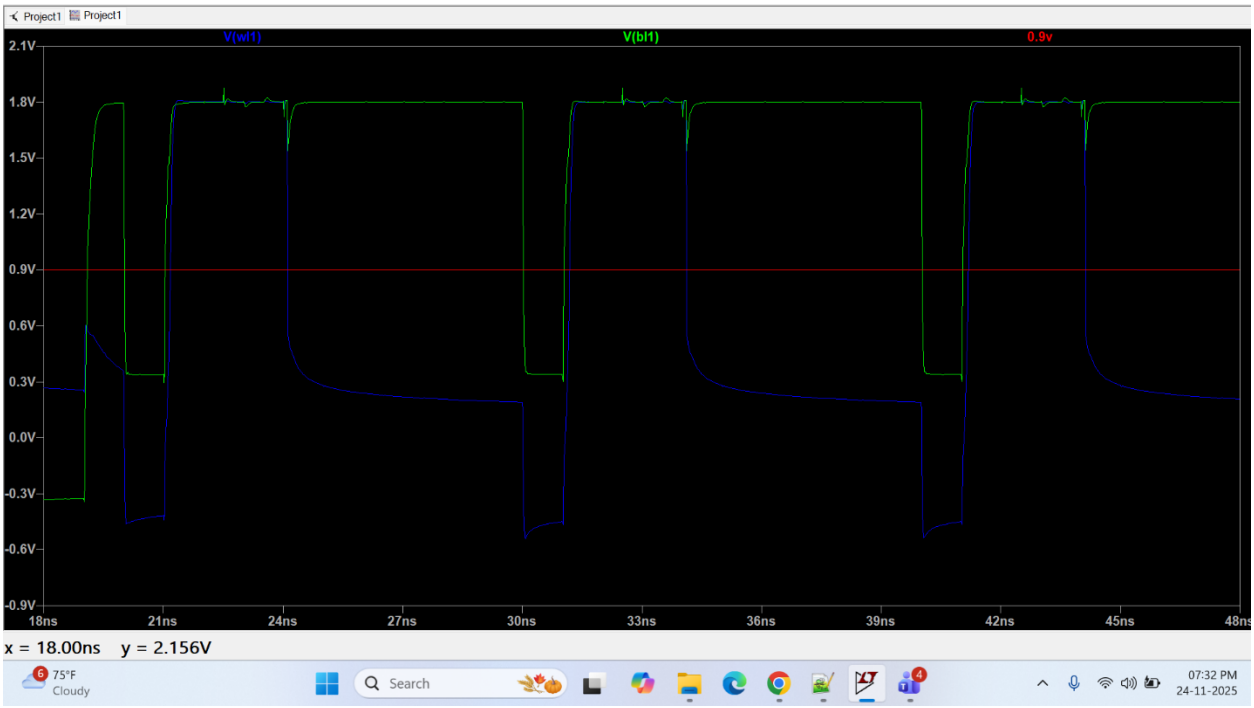


Figure 13

tplh = 31.60ns - 24.15ns = 7.45ns

tphl = 30.65 - 21.17ns = 9.48ns

tp = 8.465ns

To make 20% faster:

Faster Read Access Time

- (C_cell)= 50f
- By increasing the storage capacitance and reducing parasitic load, the bit line voltage rises significantly faster during charge sharing.

Reduced Pre-Discharge Power:

- Nmos = 400nm
- Downsizing the Pre-Discharge transistors reduces the gate capacitance that the control logic must drive. This slightly lowers the dynamic power consumption of the pre-discharge control signal during every clock cycle.

Delay After optimization:

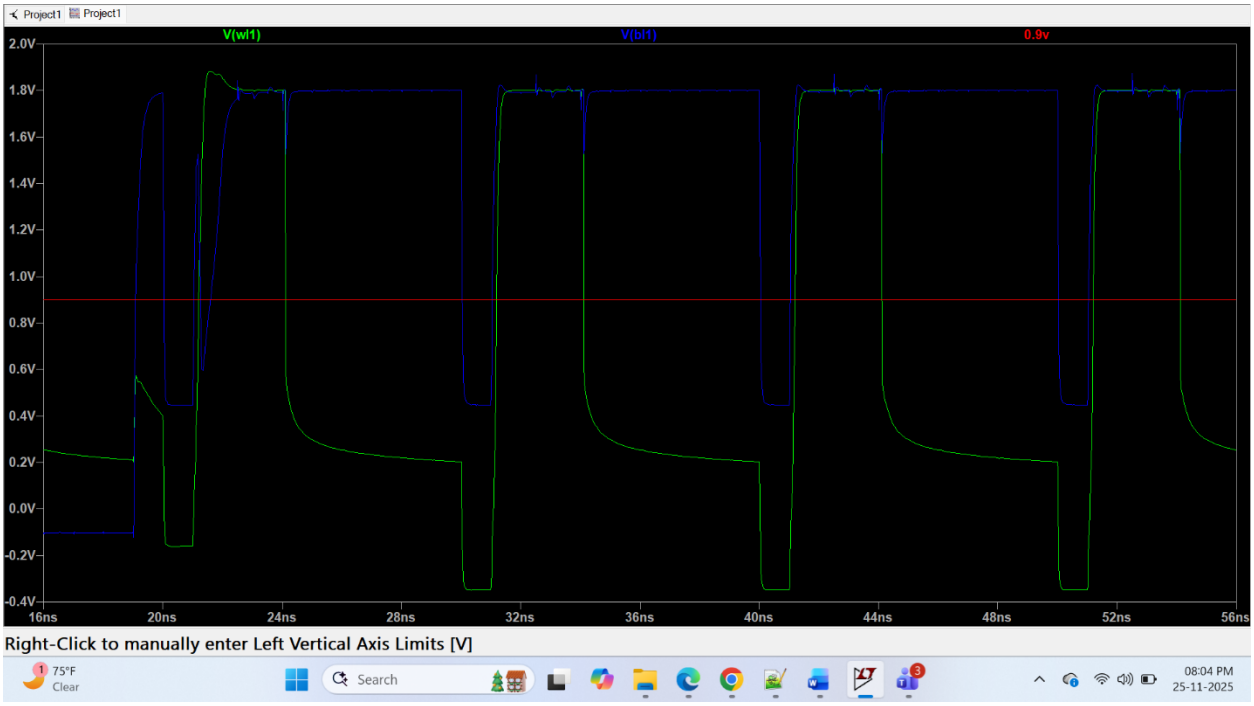


Figure 14

tplh = 30.62ns – 24.34ns = 6.28ns

tphl = 29.79ns - 22.08ns = 7.68ns

tp = 6.98 ns

Drawbacks:

Reduced Memory Density (Area Penality):

Increasing the storage capacitor. This directly increases the area required per bit, reducing the overall memory density (bits per mm2) and increasing manufacturing costs.

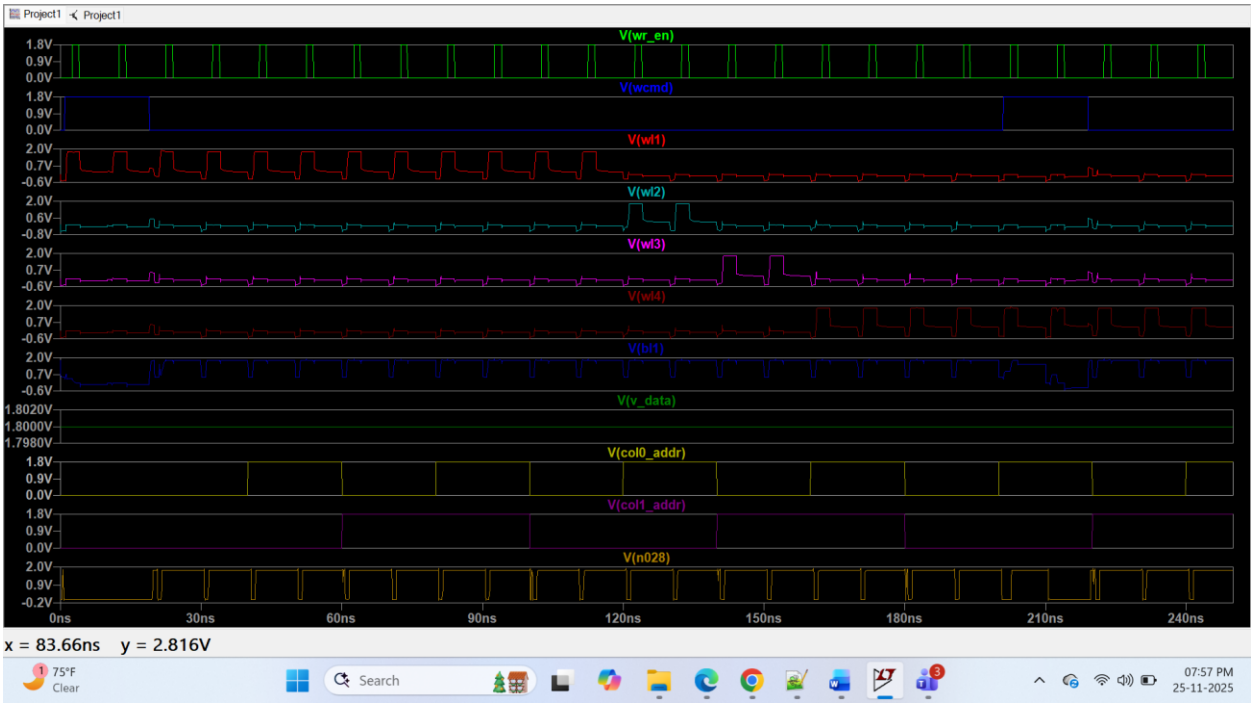
Increased Dynamic Power Consumption:

A larger capacitor stores more energy, this increases the dynamic power dissipation of the memory array, which is critical for battery-powered devices.

Slower Write Access Time:

While a larger capacitor improves Read speed (by dumping charge faster), it hurts Write speed because it takes longer to fill up through the access transistor's resistance (RC time constant increases). The CPU must drive the bit lines for a longer duration to ensure a strong '1' is stored.

Results: Waveforms/Output Explanation:



WR_EN (green)

- These short pulses represent **CPU read/write requests**.
- Each pulse starts one memory operation.

WCMD (blue)

- This signal controls the **write-back (refresh) driver**.
- When WCMD goes high, the sensed data is written back into the selected cell.

Wordlines WL1–WL4 (red, cyan, purple, brown)

- Only **one wordline rises at a time**, meaning only one **row** is active per operation.
- This shows correct **row decoding**.

Bitline BL1 (dark blue)

- Shows **small dips/bumps** instead of clean logic levels.
- This is the **destructive read behavior** of DRAM, where the cell slightly disturbs the bitline.

V_DATA (green line)

- Constant VDD → represents the **input data** used during write-back.

Column addresses COL0_ADDR (yellow) and COL1_ADDR (purple)

- These signals change every operation.
- They determine **which bit of the row buffer is sent to the CPU**.

n028 (yellow-gold)

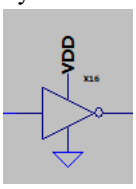
- This is the **final DRAM output**.

- It shows **clean 0 V / VDD levels**, proving that the sense amplifier restored the weak bitline signal correctly.

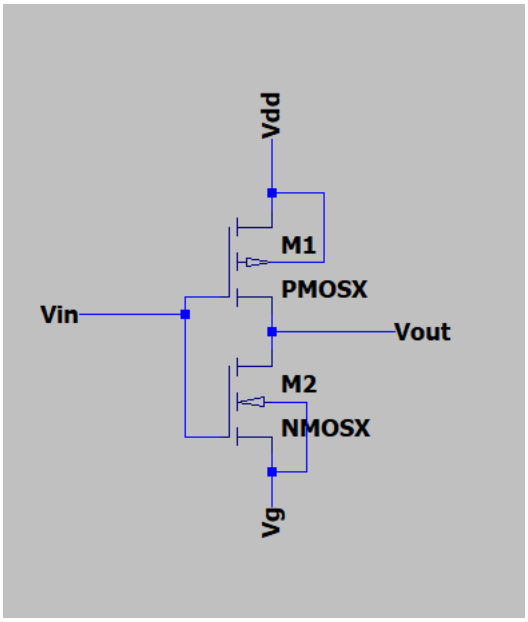
Symbols Schematics:

1. NOT Gate:

Symbol:

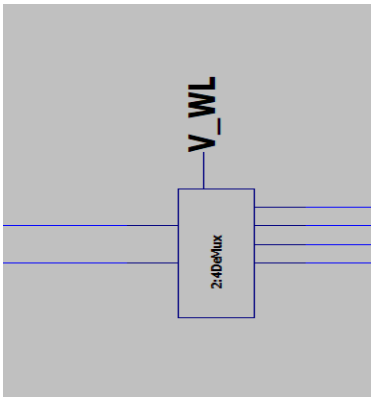


Schematic:

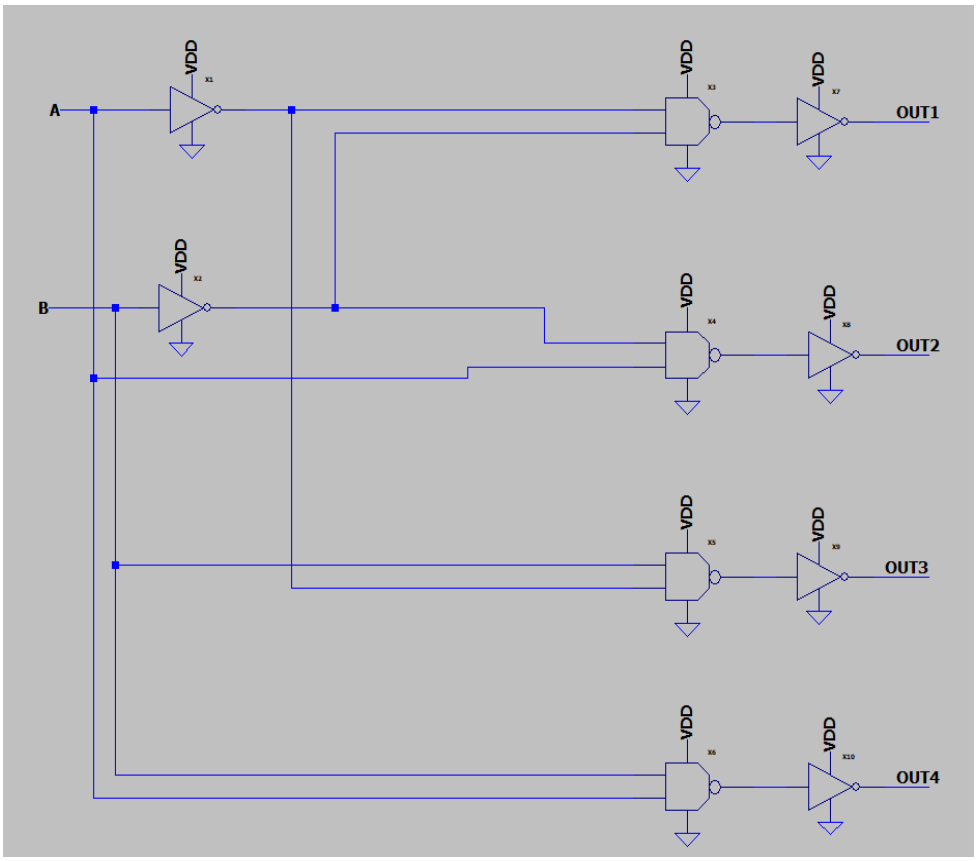


2. 2:4 DeMUX

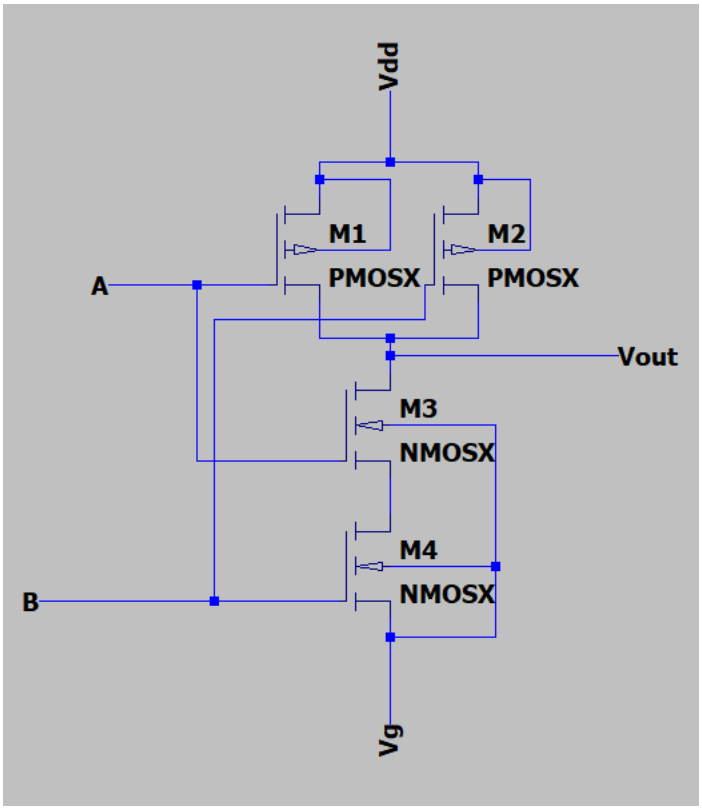
Symbol:



Schematic:

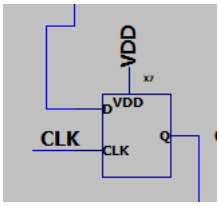

A
B
VDD
OUT1
OUT2
OUT3
OUT4

Nand Gate:

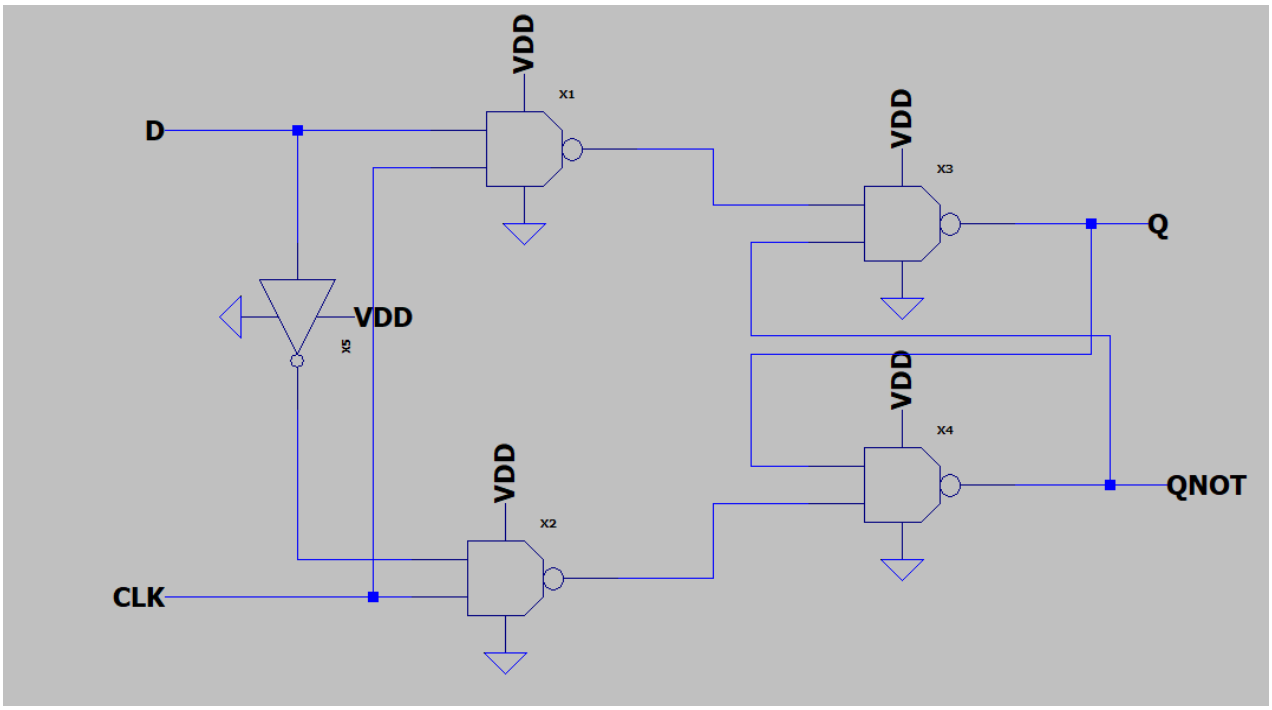

Vdd
M1
PMOSX
M2
PMOSX
A
Vout
M3
NMOSX
M4
NMOSX
B
Vg

3. DFlipFlop:

Symbol:

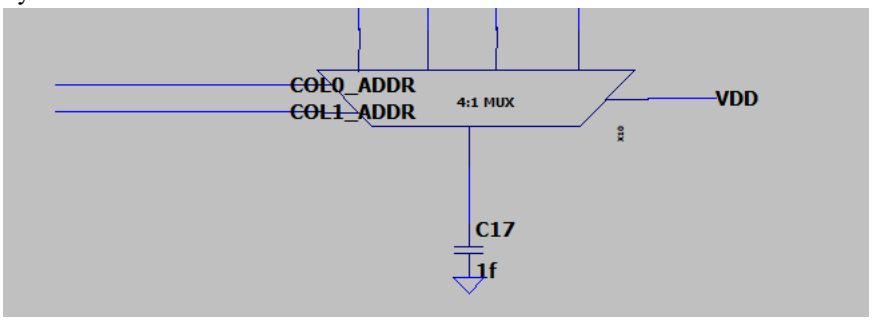


Schematic:

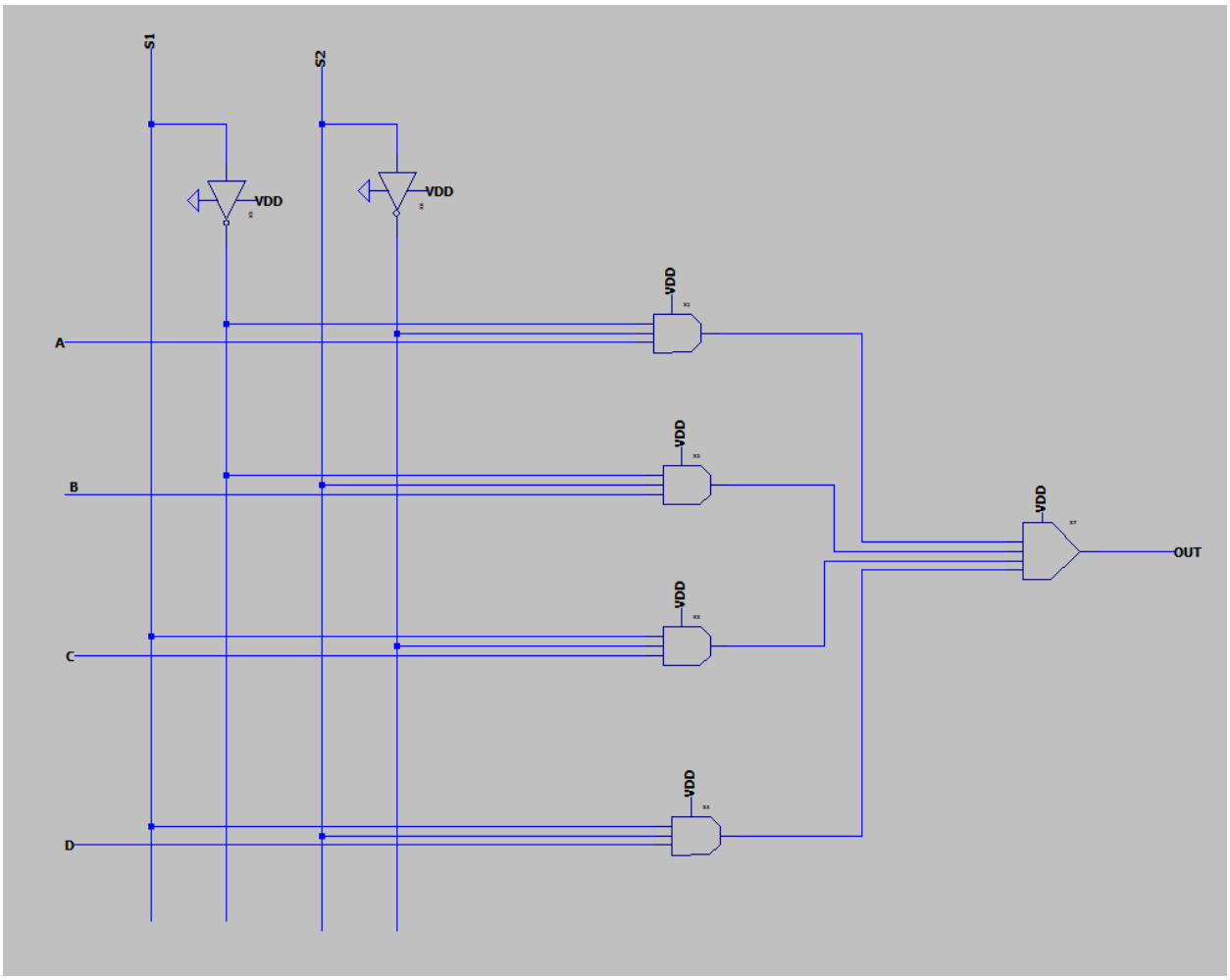


4. 4:1Mux

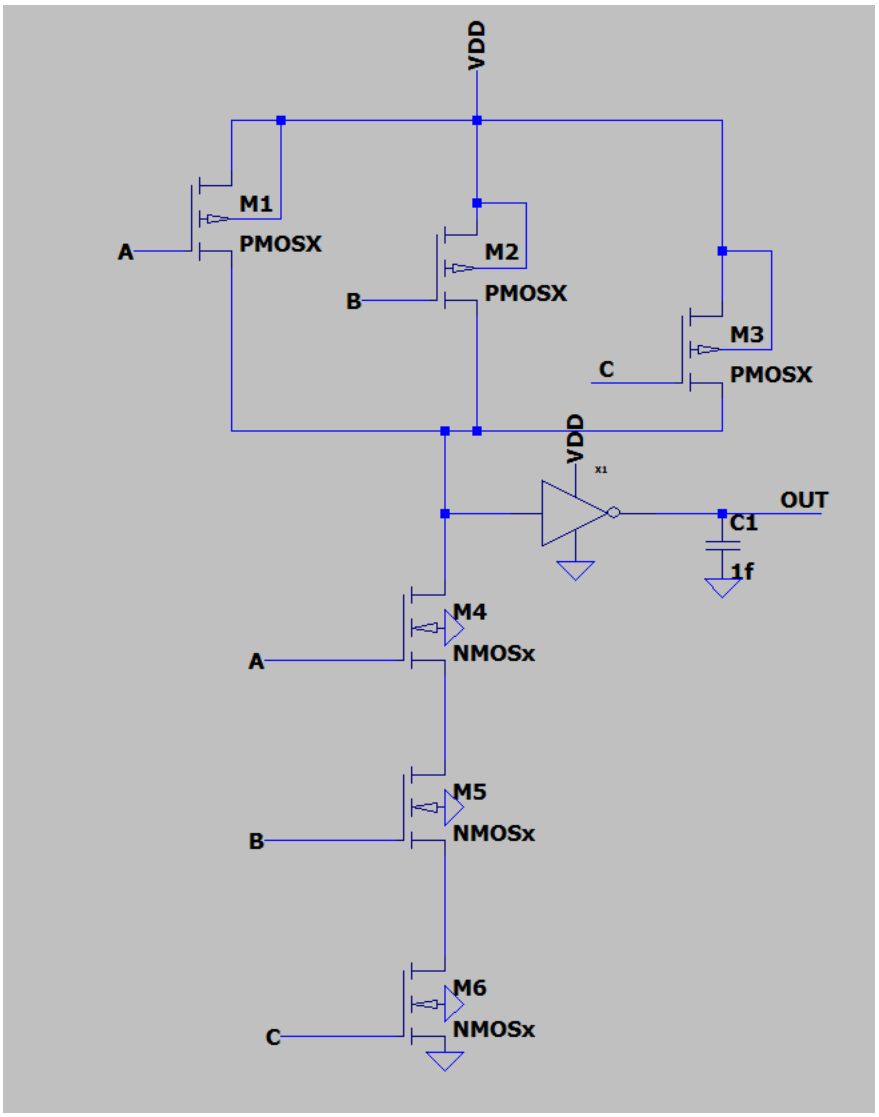
Symbol:



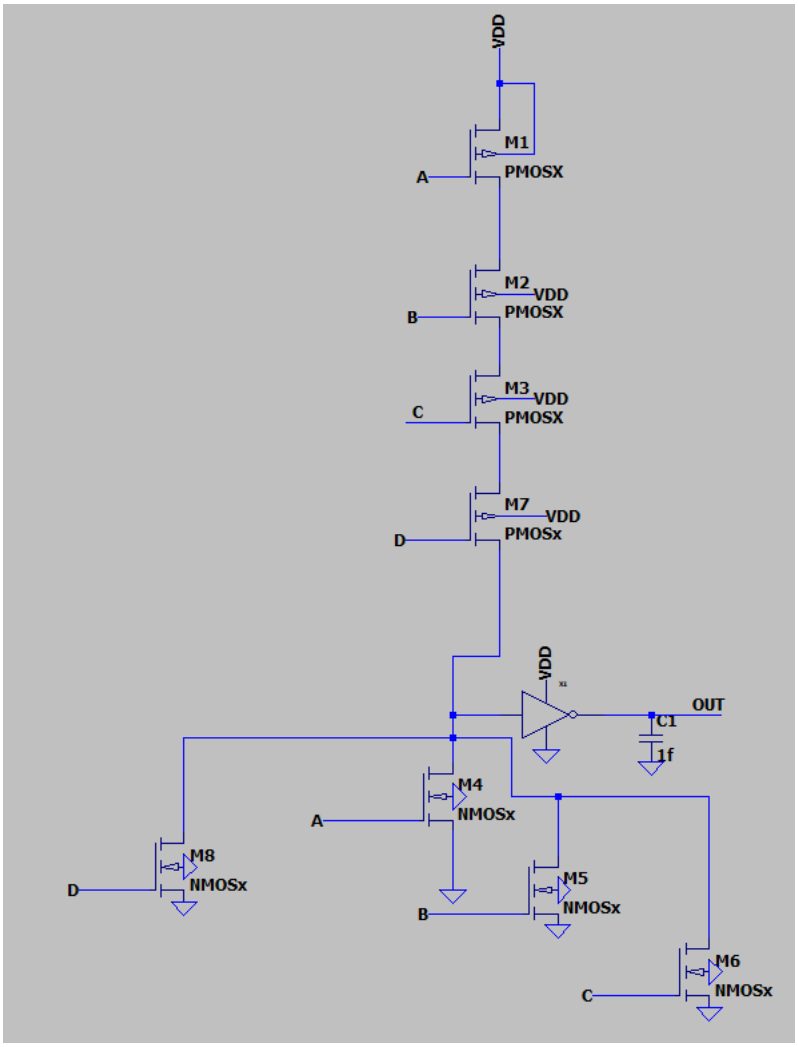
Schematic:



AND3Input:


VDD
M1
A
PMOSX
M2
B
PMOSX
M3
C
PMOSX
VDD
x1
OUT
C1
1f
M4
A
NMOSx
M5
B
NMOSx
M6
C
NMOSx

OR4Input:


VDD
M1
PMOSX
A
M2
VDD
PMOSX
B
M3
VDD
PMOSX
C
M7
VDD
PMOSx
D
VDD
OUT
C1
1f
M4
NMOSx
A
M8
NMOSx
D
M5
NMOSx
B
M6
NMOSx
C