Real-World Implementation Of Semiconductors 8x8 SRAM

Using tsmc180nm CMOS technology **Circuit Level

Represented by

- Ahmed Mohamed Ahmed Hussien Mohamed
- Abdelrahman Ahmed Sheded
- Mahmoud Mohamed Abdelhamid Azazi
- Omar Ali Mohamed El Telbany
- Ahmed Mohamed Abd El Montasser
- Ahmed Mohammed Moussa Ali
- Mohamed Reda El Sayed Mohamed Shaheen

SUPERVISORS

DR/ ALSHIMAA NABIL

DR/ HOWIDA ABDELHALYM

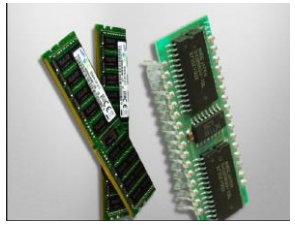
ECE DEPARTMENT - FACULTY OF ENGINEERING

ZAGAZIG UNIVERSITY

Literature Review :

Paper	Technology	Cell Size	Power Dissipation	Propagation Delay	Analysis & Tools	Advantages	Disadvantages
1	0.25u SCMOS	6.5um x 6.5um	0.5mW (read), 0.8mW (write)	1.2ns (read), 1.8ns (write)	HSPICE simulation, Microwind layout tool	High density, low power consumption, high speed	High leakage current, low noise margin
2	65nm CMOS	0.8um x 0.8um	0.4mW (read), 0.6mW (write)	1ns (read), 1.5ns (write)	Cadence Virtuoso tool, Spectre simulator	Compact size, low power consumption, high speed	High leakage current, low noise margin
3	180nm CMOS	3um x 3um	0.3mW (read), 0.4mW (write)	1ns (read), 1.5ns (write)	Tanner EDA tool, SPICE simulator	Low power consumption, high speed, improved stability	Large cell size, high leakage current
[4]	45nm CMOS	Not given	Not given	Not given	HSPICE simulation, Microwind layout tool	Low power consumption, high speed, reduced leakage current	Not given
[5]	180nm cmos	Not given	U.C 131uw (write)	0.47ns(Read) 0.96ns(write)	HSPICE simulation, Microwind layout tool	Low power consumption, high speed, improved stability and reliability	Not given
[6]	Not given	Not given	Not given	Not given	HSPICE simulation, Cadence layout tool	Low power consumption, high speed, improved stability and reliability, variability tolerance and low-voltage operation	Larger cell size than 6T SRAM
[7]	130nm RRAM-CMOS hybrid technology	Not given	Not given	Not given	Cadence Virtuoso tool, Spectre simulator	Non-volatility, low power consumption, high speed, improved stability and reliability	Larger cell size than conventional SRAM
Our work	TSMC 180NM	Not given	120 uW for 8*8 (W)	W : 839 ps , R :691 ps	Cadence Virtuoso simulator	low power, stable read and write conditions	low speed , moderate area

Storage Types :

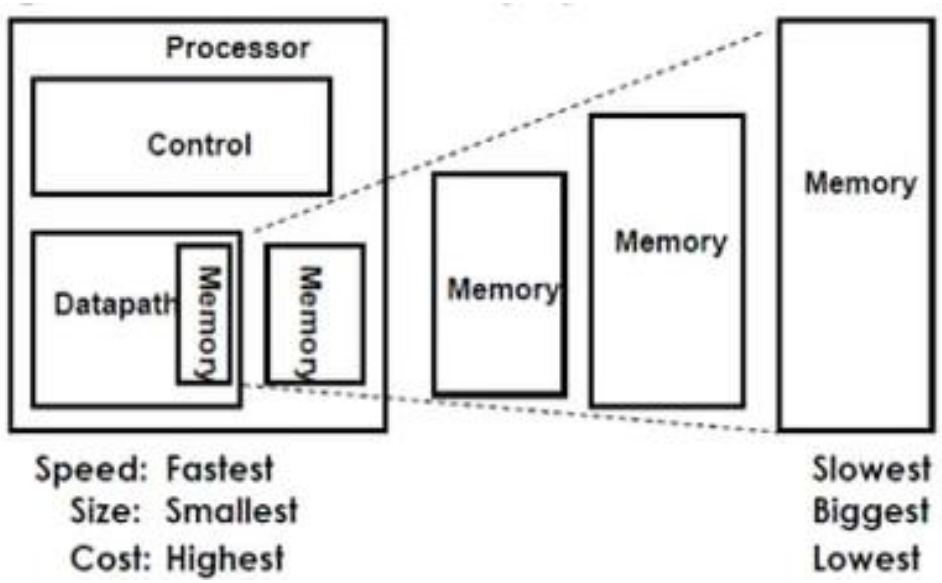


- Volatile → DRAM & SRAM
- Non-volatile → Magnetic Storage(HDD) , solid state device (SSD) and FLASH MEMORY.

Traditional hard disk drive

Solid state hard drive

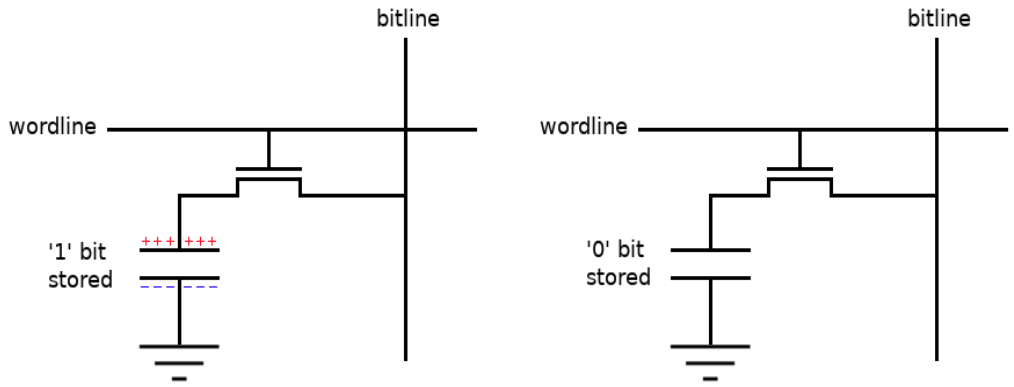
Memory Hierarchy



RAM types :

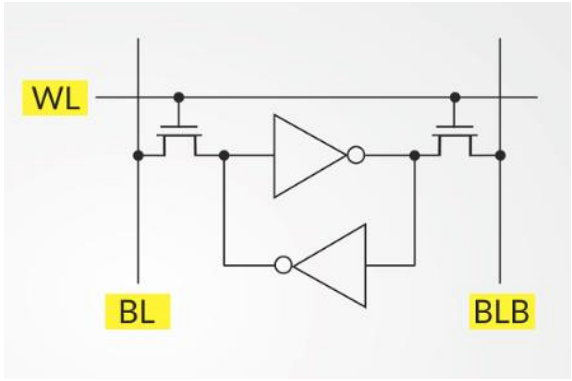
DRAM

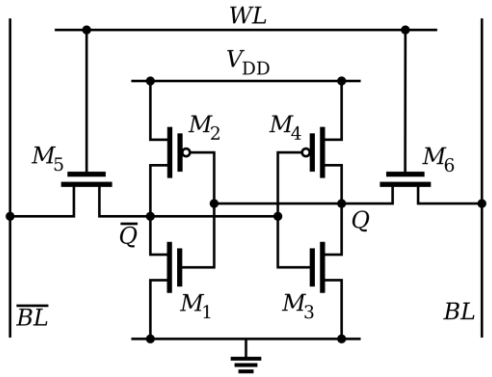
- Charge determine the stored bit (0,1)
- require periodic refresh cycle
  to maintain stored data.



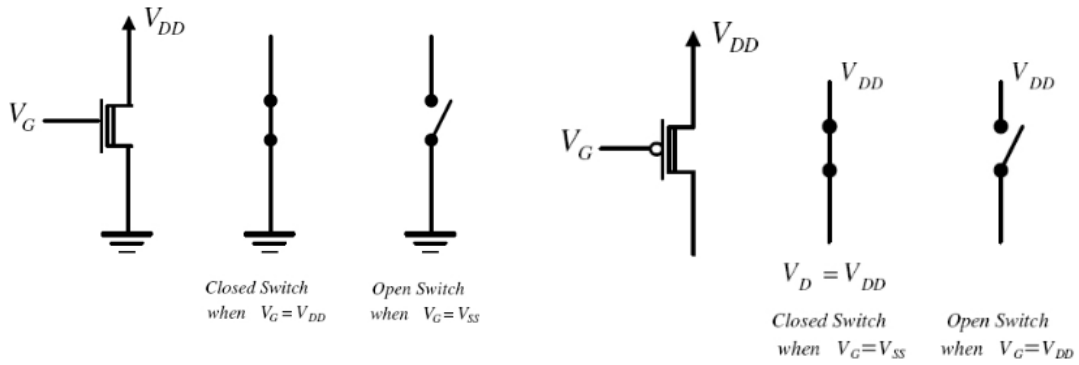
SRAM

- Hold data without external refresh
- Simplicity : don't require external refresh circuitry
- Speed: SRAM is faster than DRAM
- Cost: several times more expensive than DRAMs
- Size: take up much more space than DRAMs
- Power: consume more power than DRAMs
- Usage: level 1 or level 2 cache



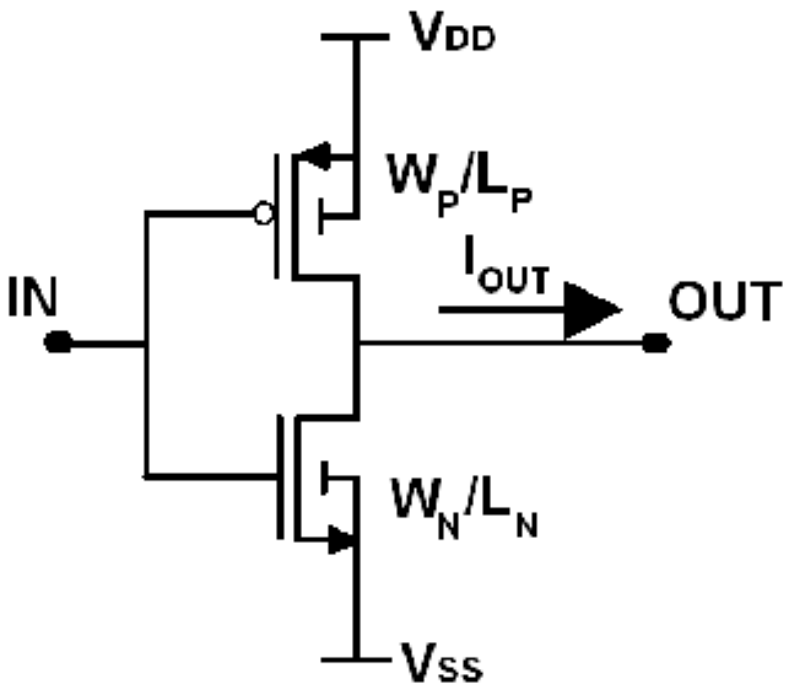


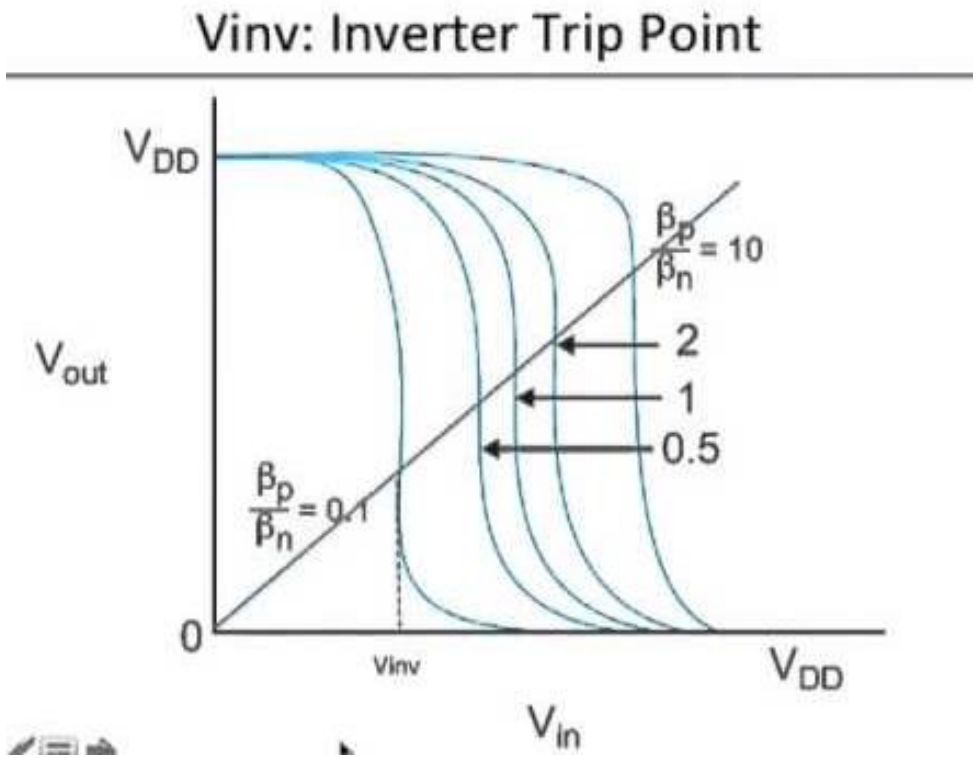
MOSFET as a switch



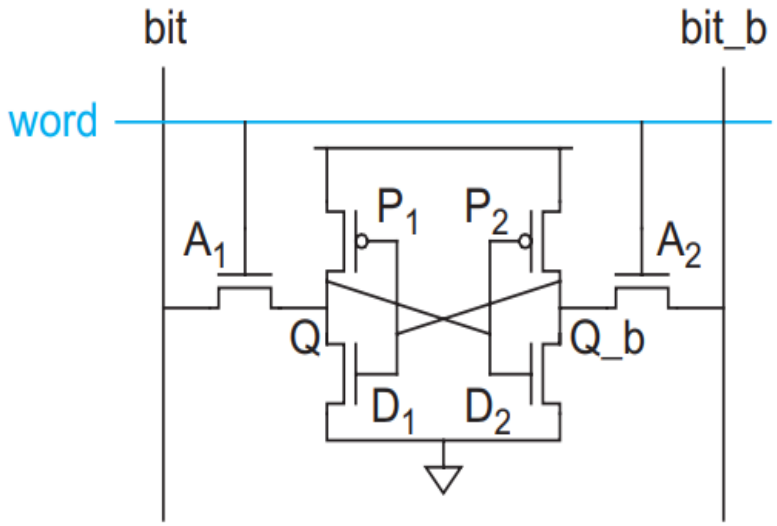
Inverter trip point

Assume WP/WN = Bp/Bn





How to size MOSFETS at SRAM



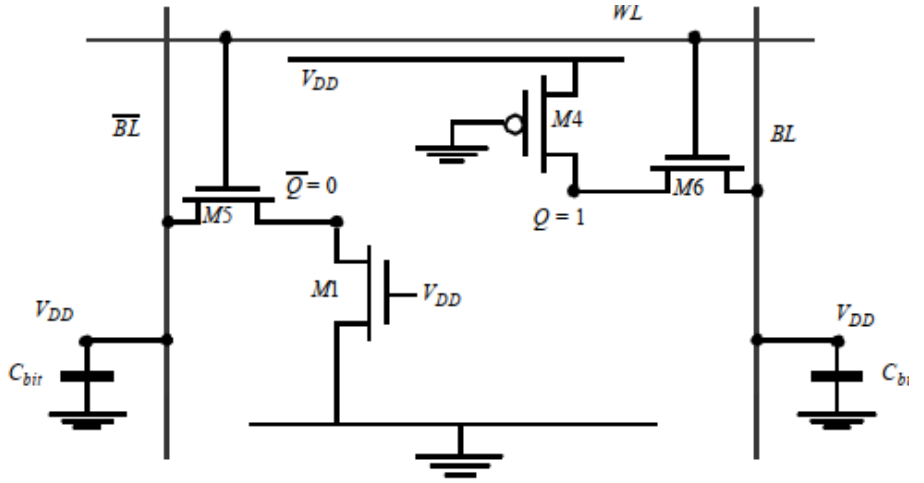
To ensure both read stability and write stability, the transistors must satisfy ratio constraints.

The NMOS pulldown transistors (D1 ,D2) in the cross-coupled inverters must be strongest.

The access transistors (A1,A2) are of intermediate strength, and the PMOS pullup transistors(P1,P2) must be weak.

We will tell why soon.

SRAM Read Operation



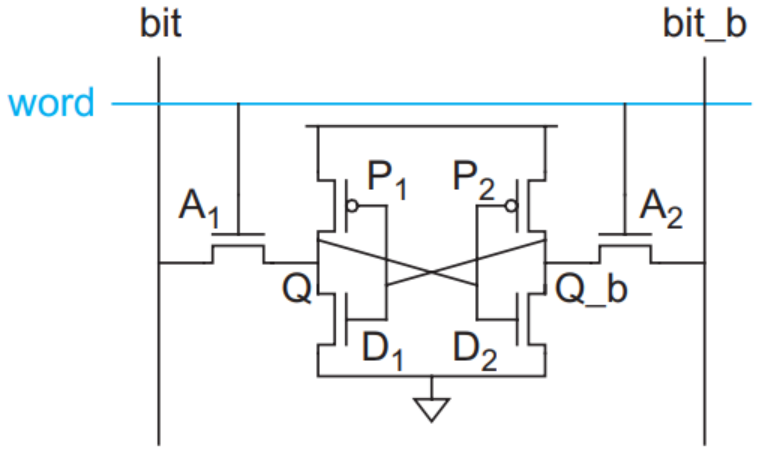
Read Operation:-the value is a l, stored at Q.
The read cycle is started by pre charging both the bitlines to a logical 1, then asserting the word line WL enabling both the access transistors.

- The second step occurs when the values stored in Q and Q' are transferred to the bit lines by leaving BL at its precharged value and discharging BL through M1 and M5 to a logical O.
- On the BL side, the transistors M4 and M6 pull the bit line towards VDD, a logical 1.
- If the content of the memory was a O, the opposite would happen and BL would be pulled toward I and BL toward O.

Write Operation

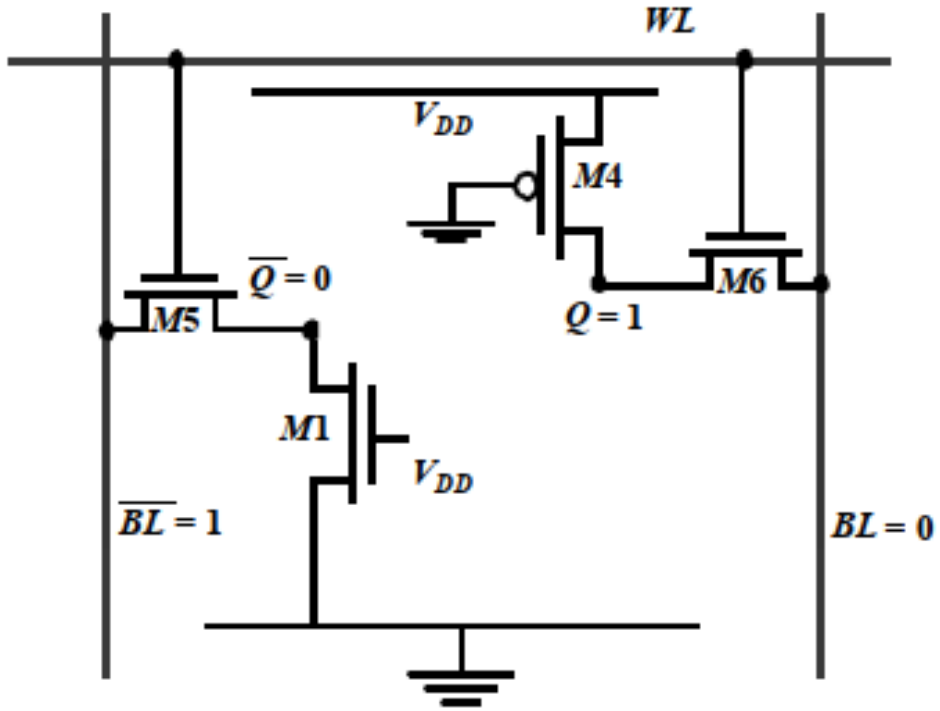
The start of a write cycle begins by applying the value to be written to the bit lines.

- If we wish to write a O, we would apply a O to the bit lines, i.e. setting BL' to 1 and BL to O.



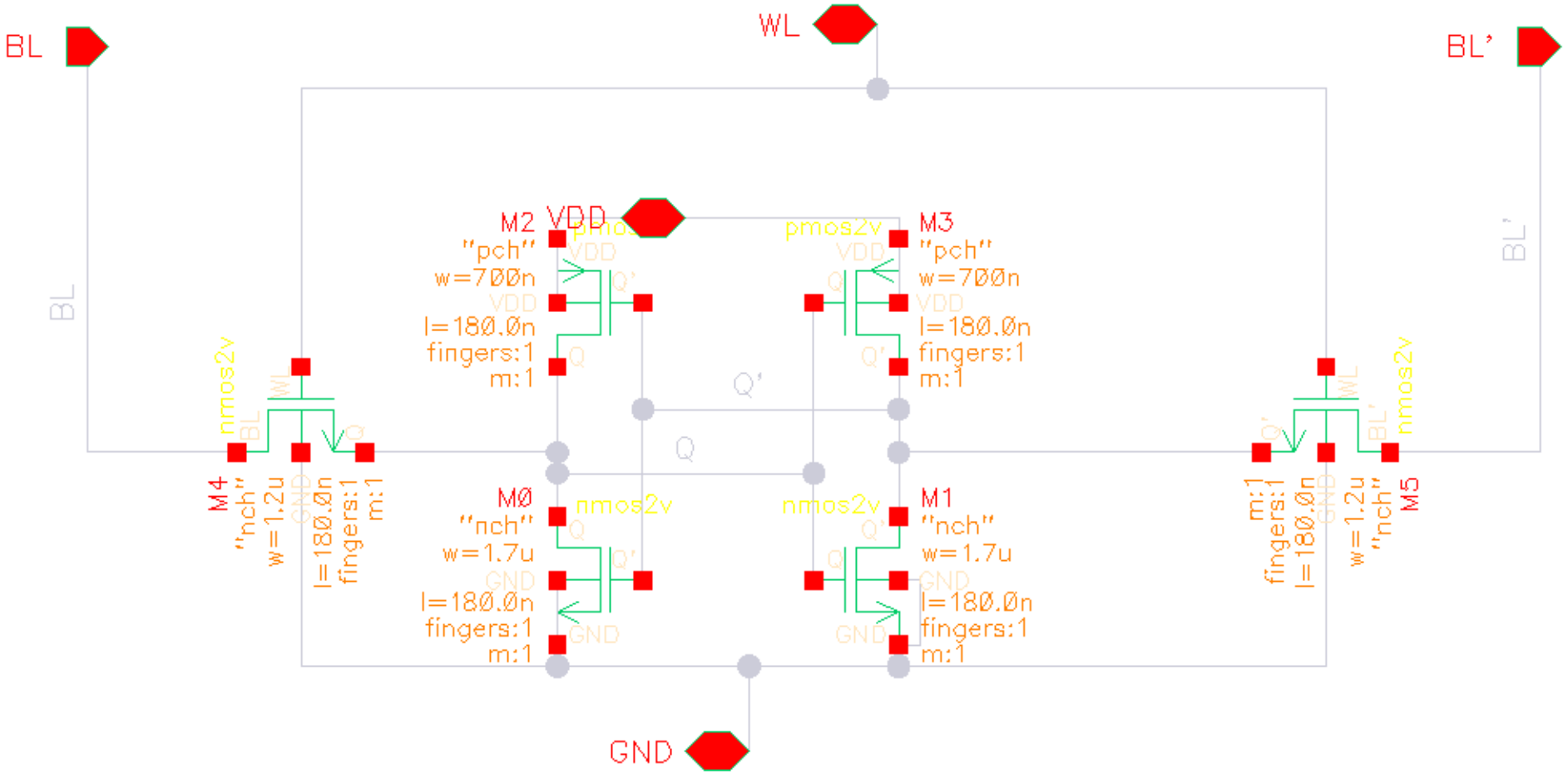
HOLD OPERATION

As long as word line is not activated the SRAM will hold is status no matter what is applied to bit line. No need for a refreshment circuit like DRAM.

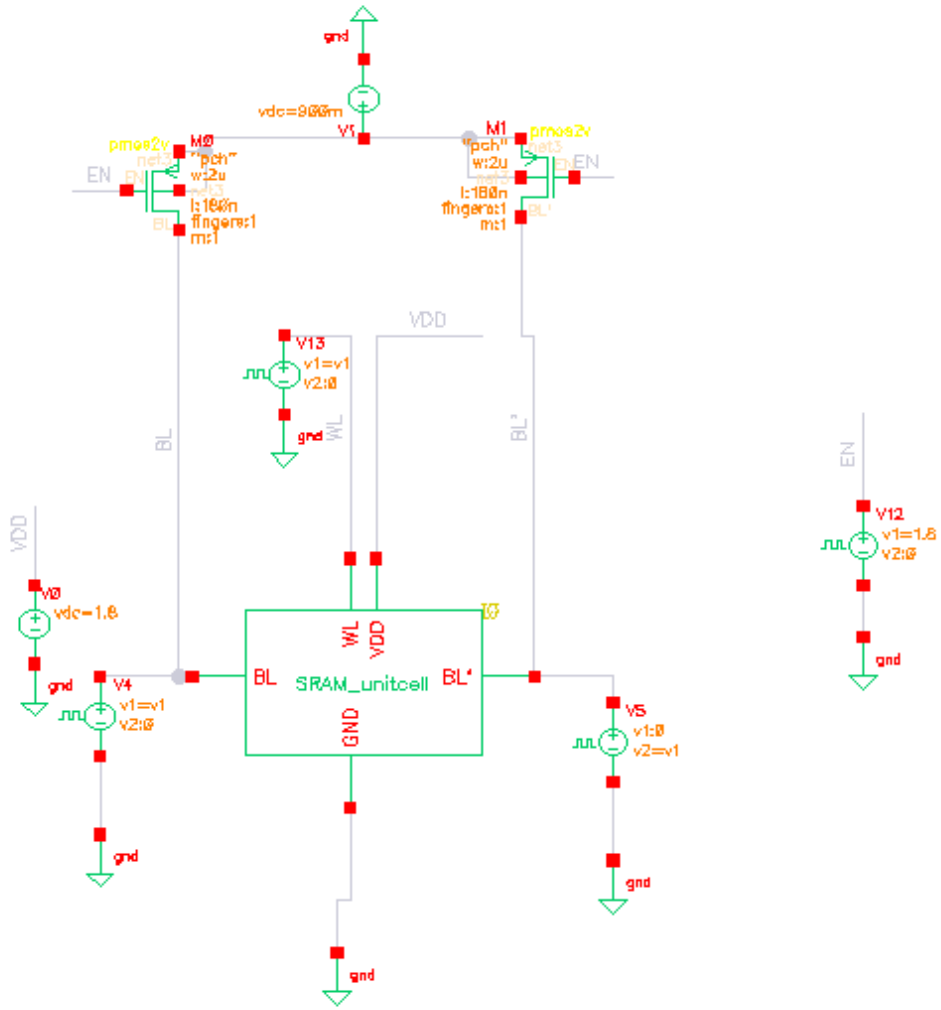


6T SRAM Circuit implementation

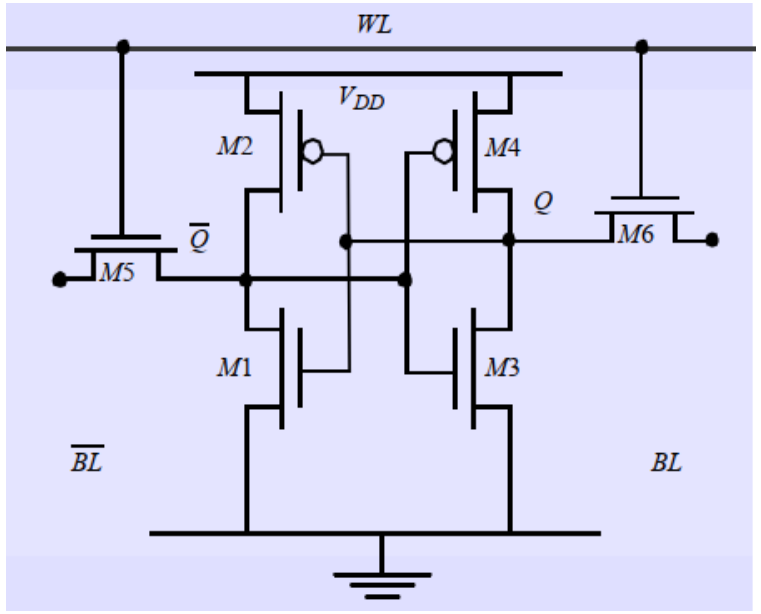
(On circuit simulator tool : CADENCE VIRTUOSO)

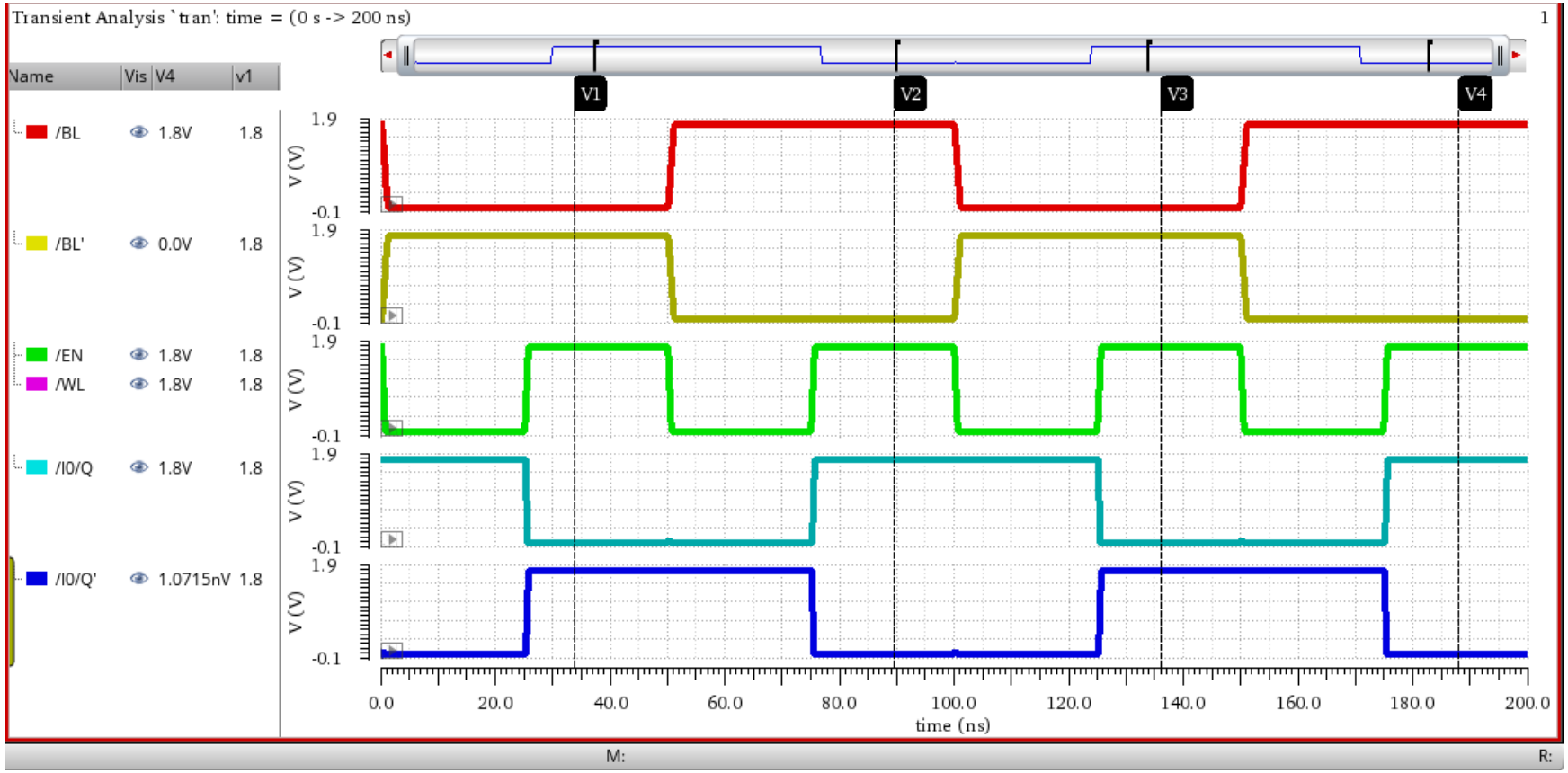


Unit cell testbench write mode

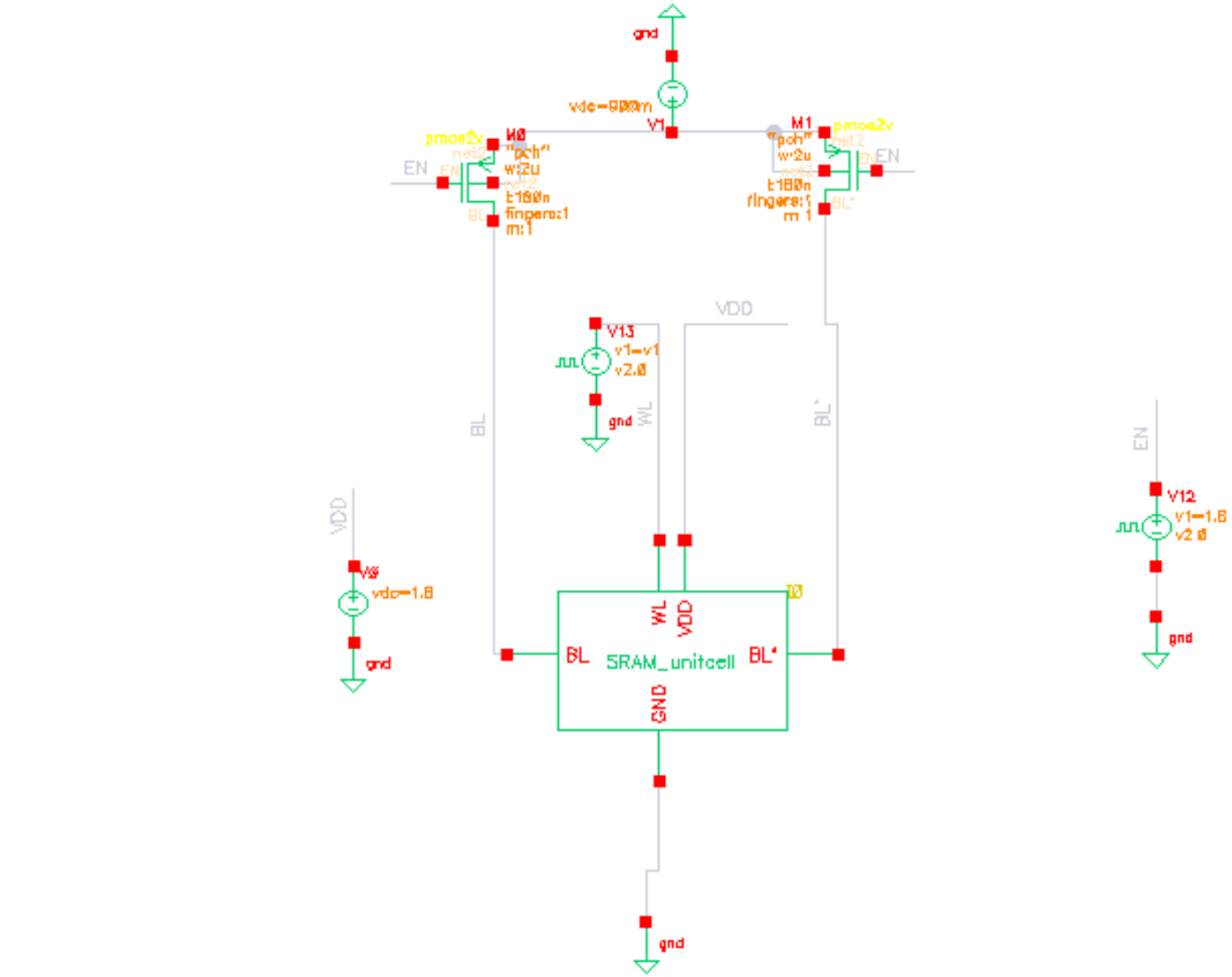


Transient analysis (write mode)

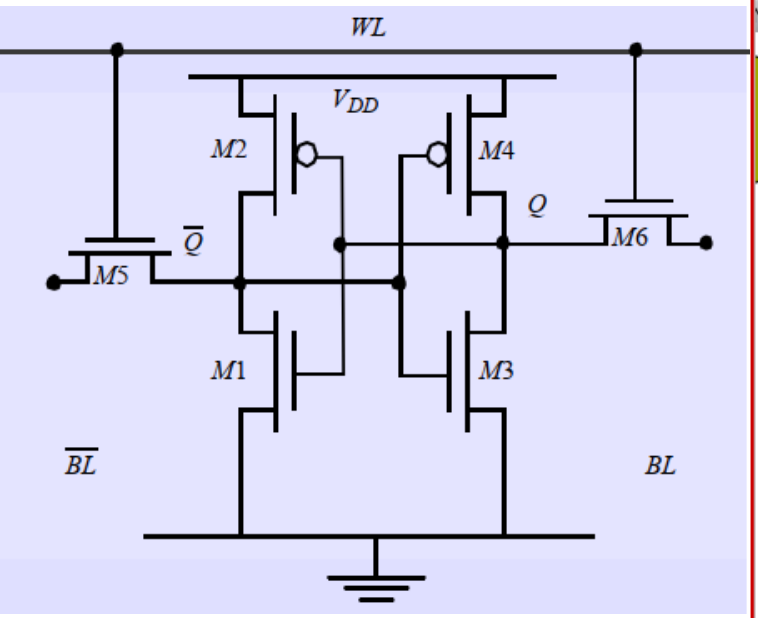


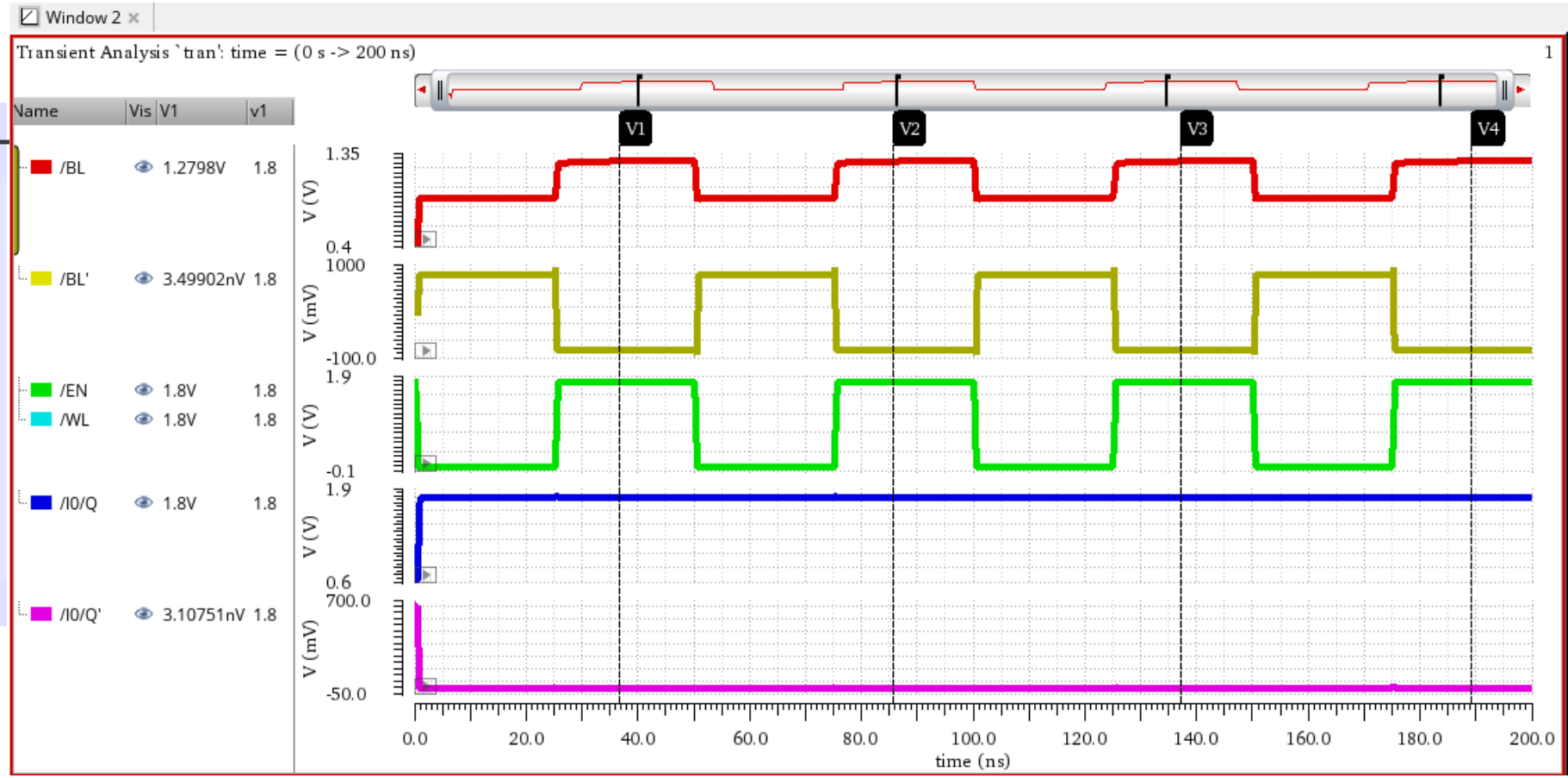


Unit cell testbench read mode



Transient analysis (read mode)





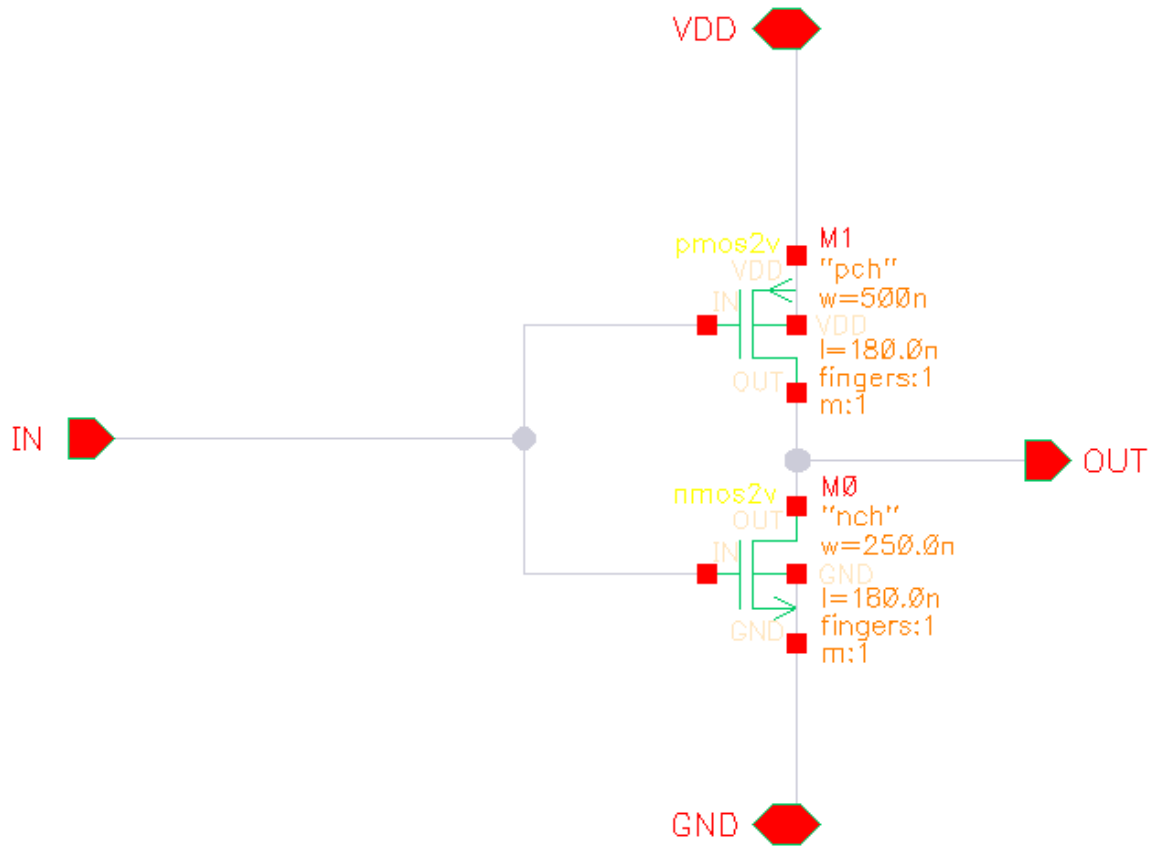
Decoder truth table

Enable	INPUTS			Outputs							
E	A_2	A_1	A_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

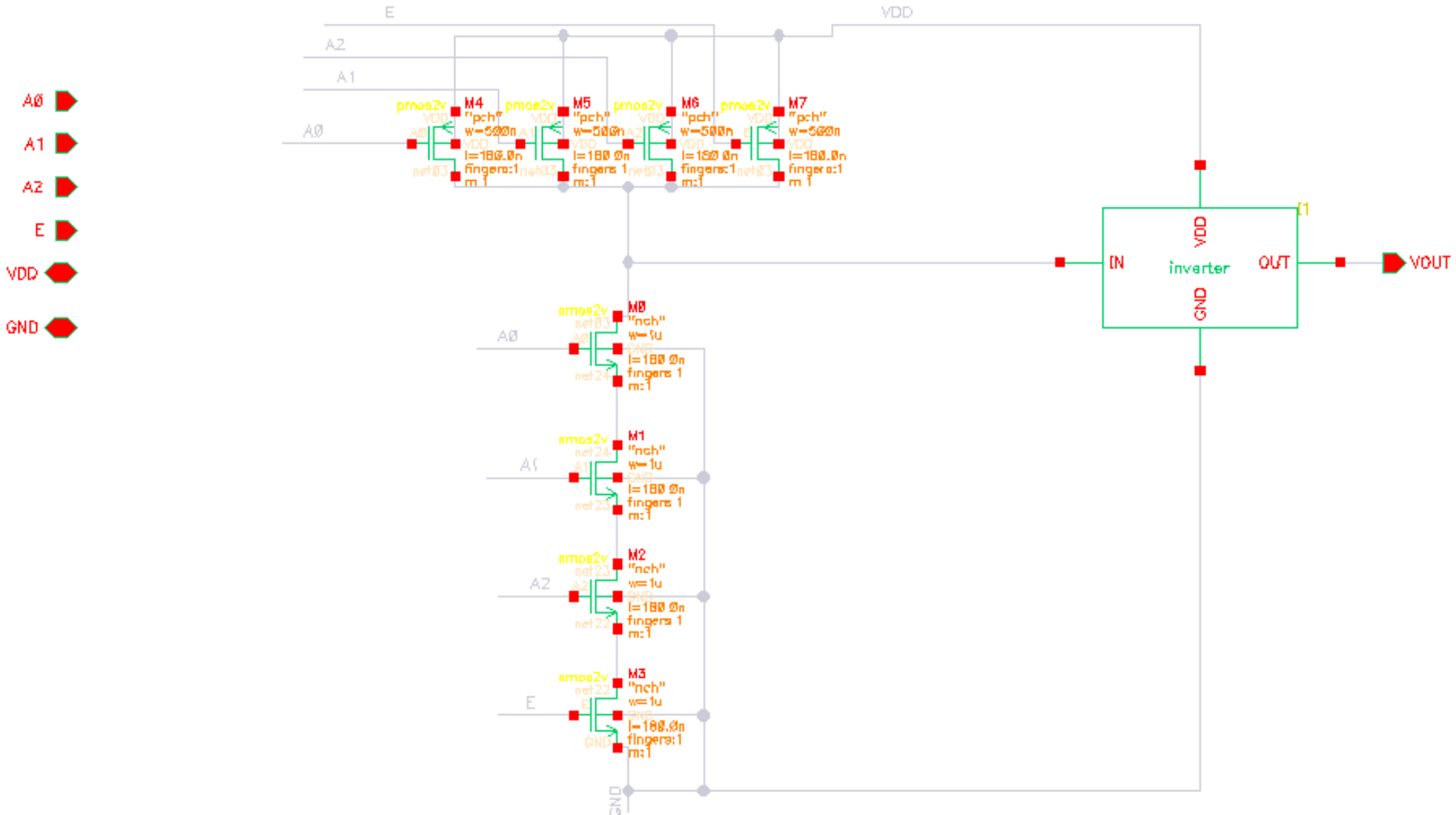
The logic expression

$$Y_0 = EN.A_0'.A_1'.A_2'$$
$$Y_1 = EN.A_0.A_1'.A_2'$$
$$Y_2 = EN.A_0'.A_1.A_2'$$
$$Y_3 = EN.A_0.A_1.A_2'$$
$$Y_4 = EN.A_0'.A_1'.A_2$$
$$Y_5 = EN.A_0.A_1'.A_2$$
$$Y_6 = EN.A_0'.A_1.A_2$$
$$Y_7 = EN.A_0.A_1.A_2$$

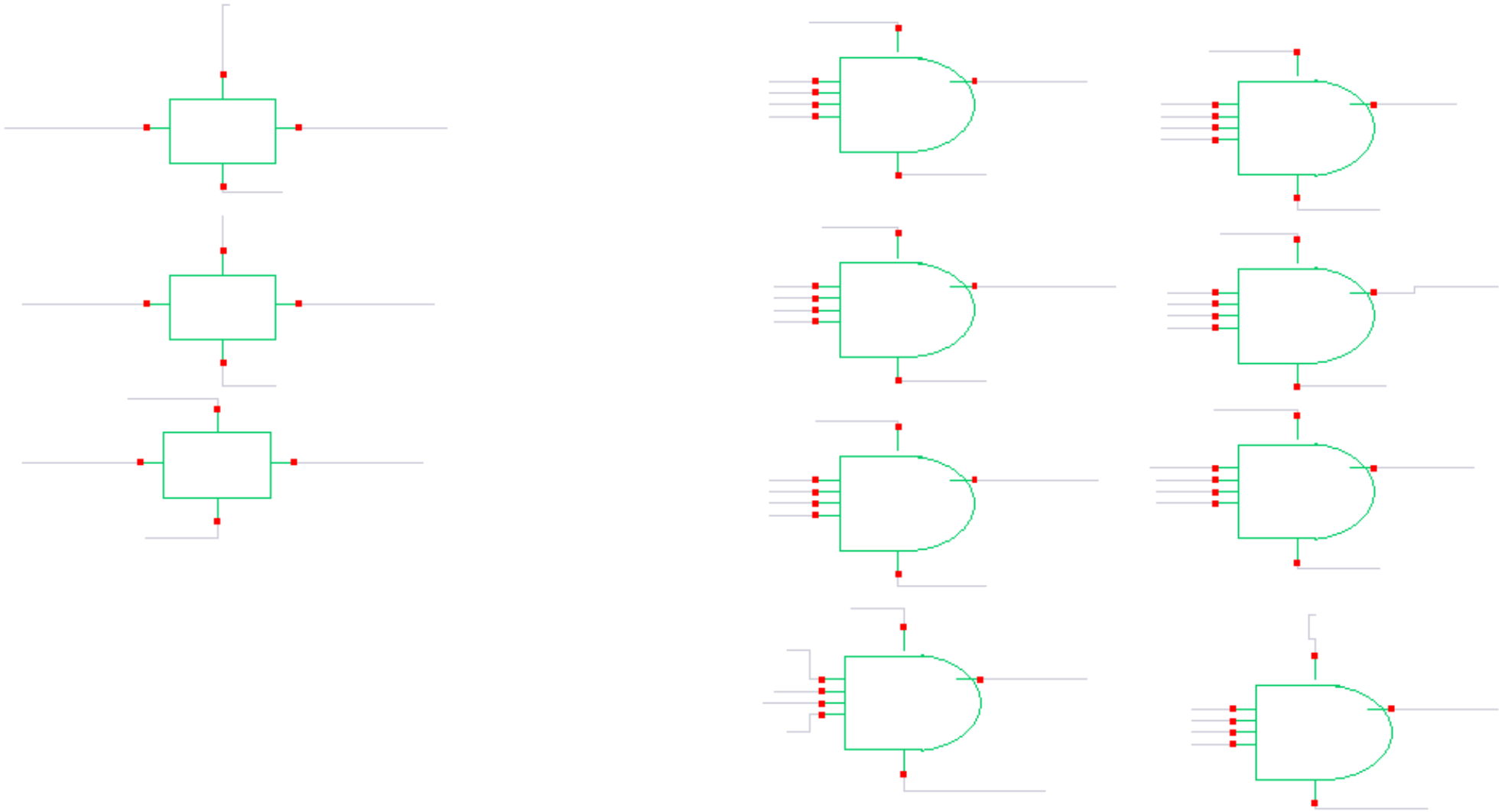
INVERTER circuit diagram



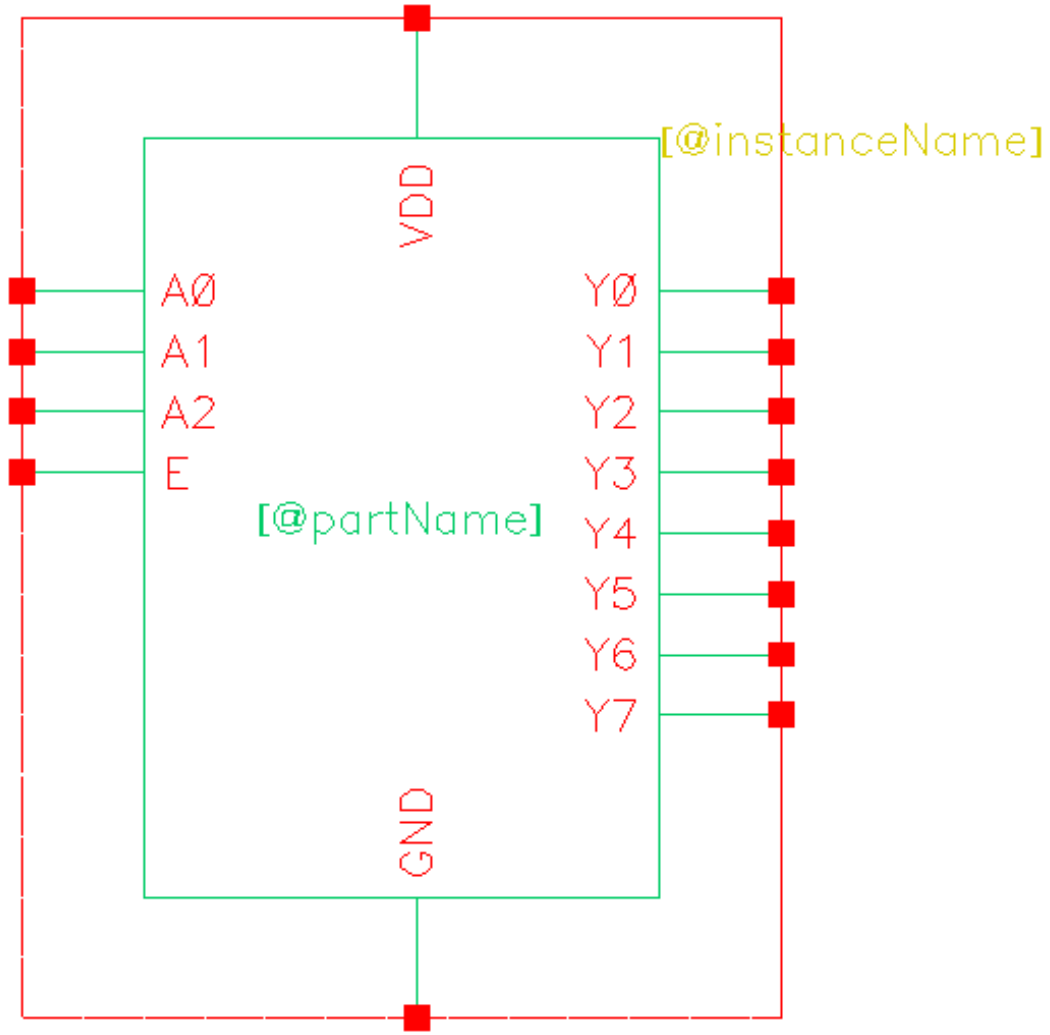
4-input AND circuit



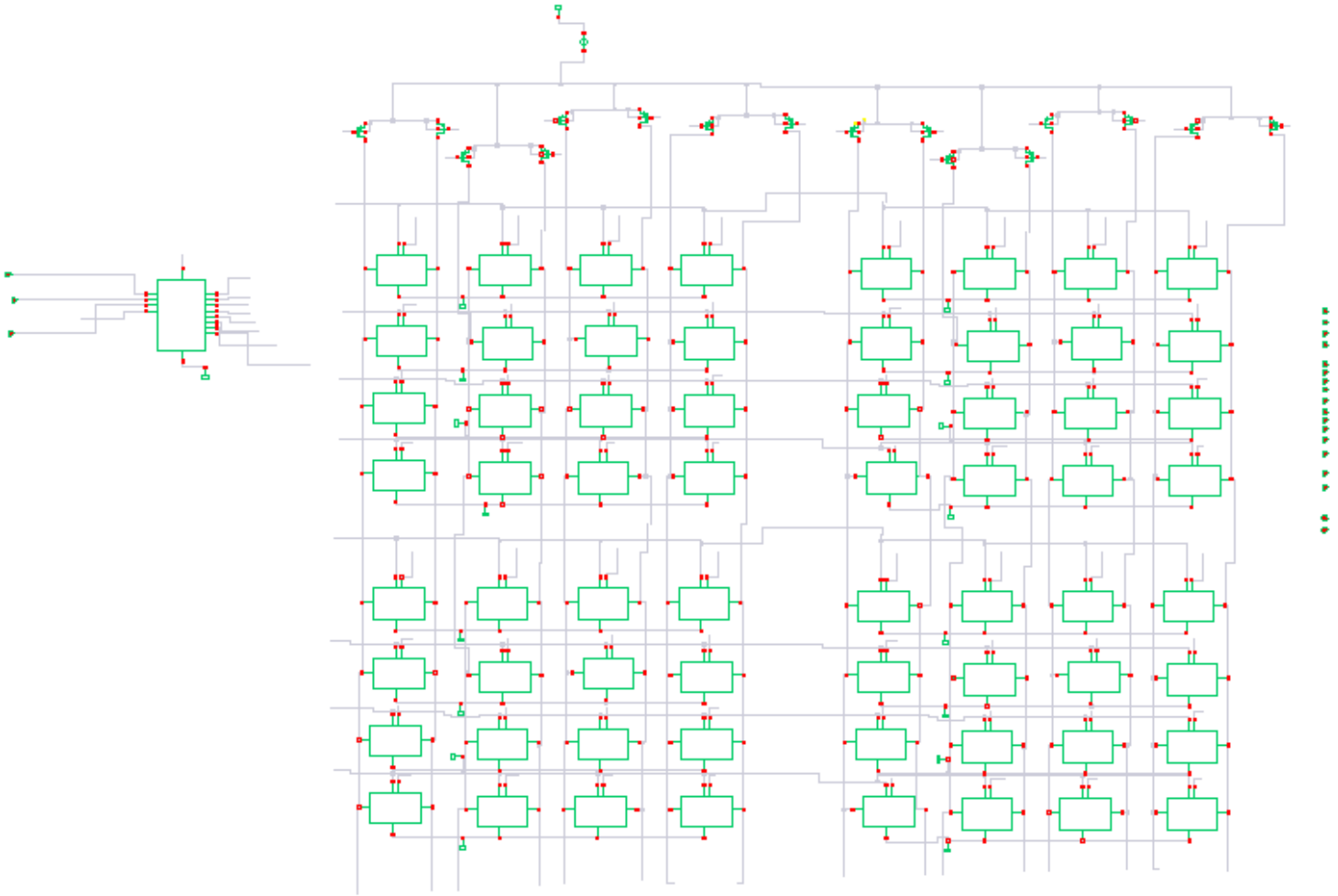
Decoder design implementation



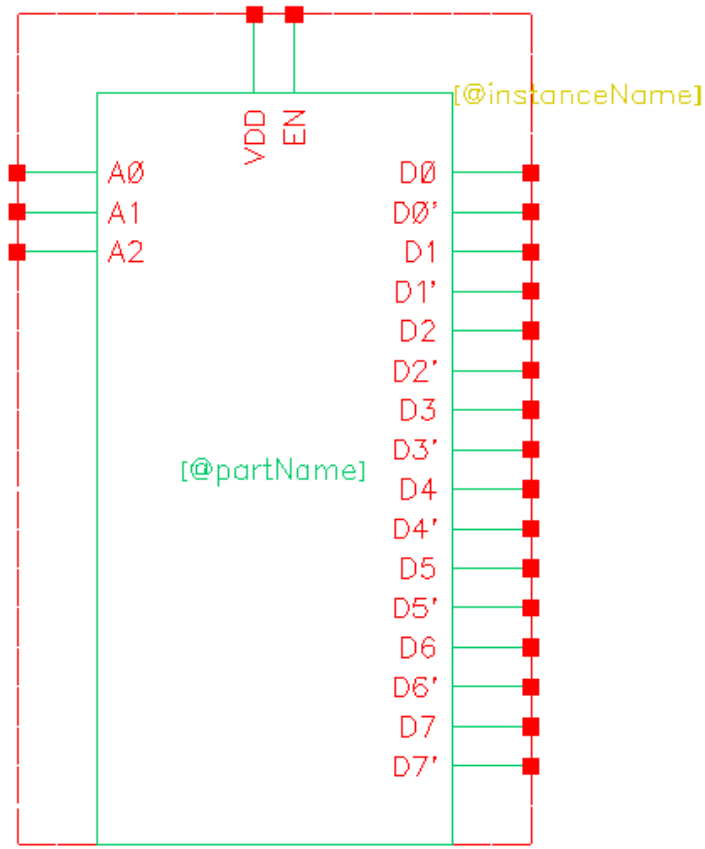
DECODER symbol



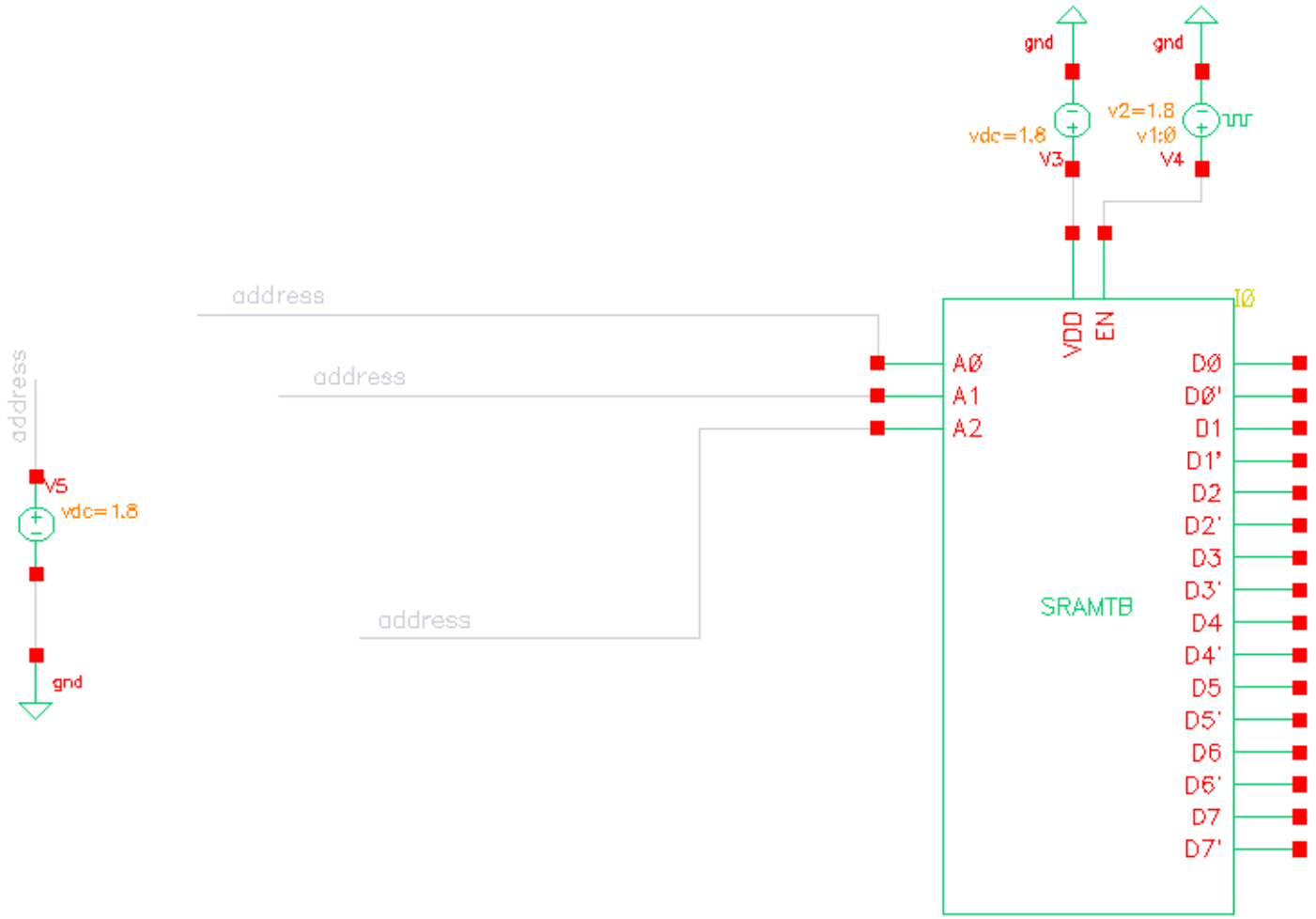
8x8 SRAM as a one unit



8X8 SRAM SYMBOL (as a unit chip IC)

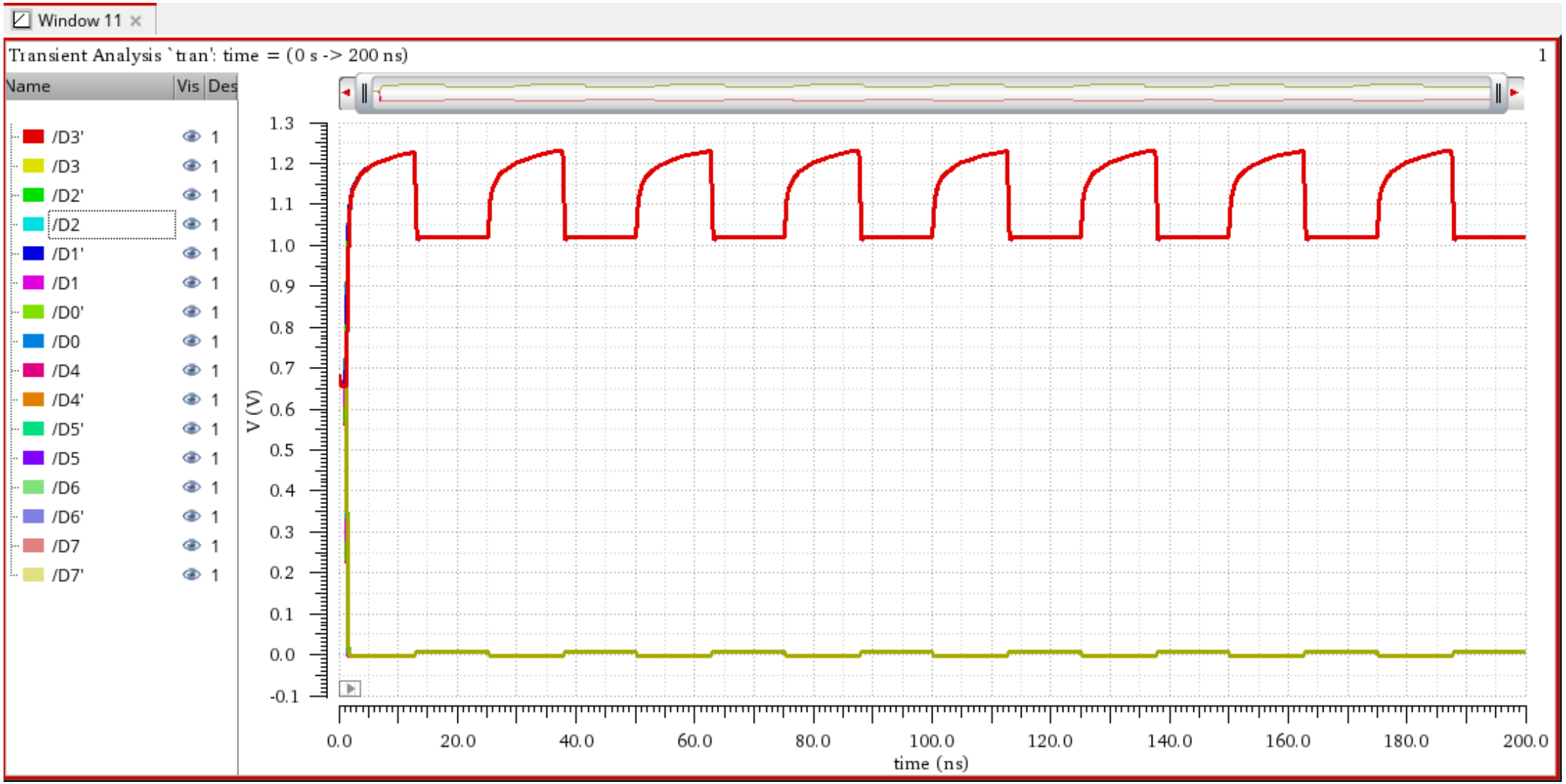


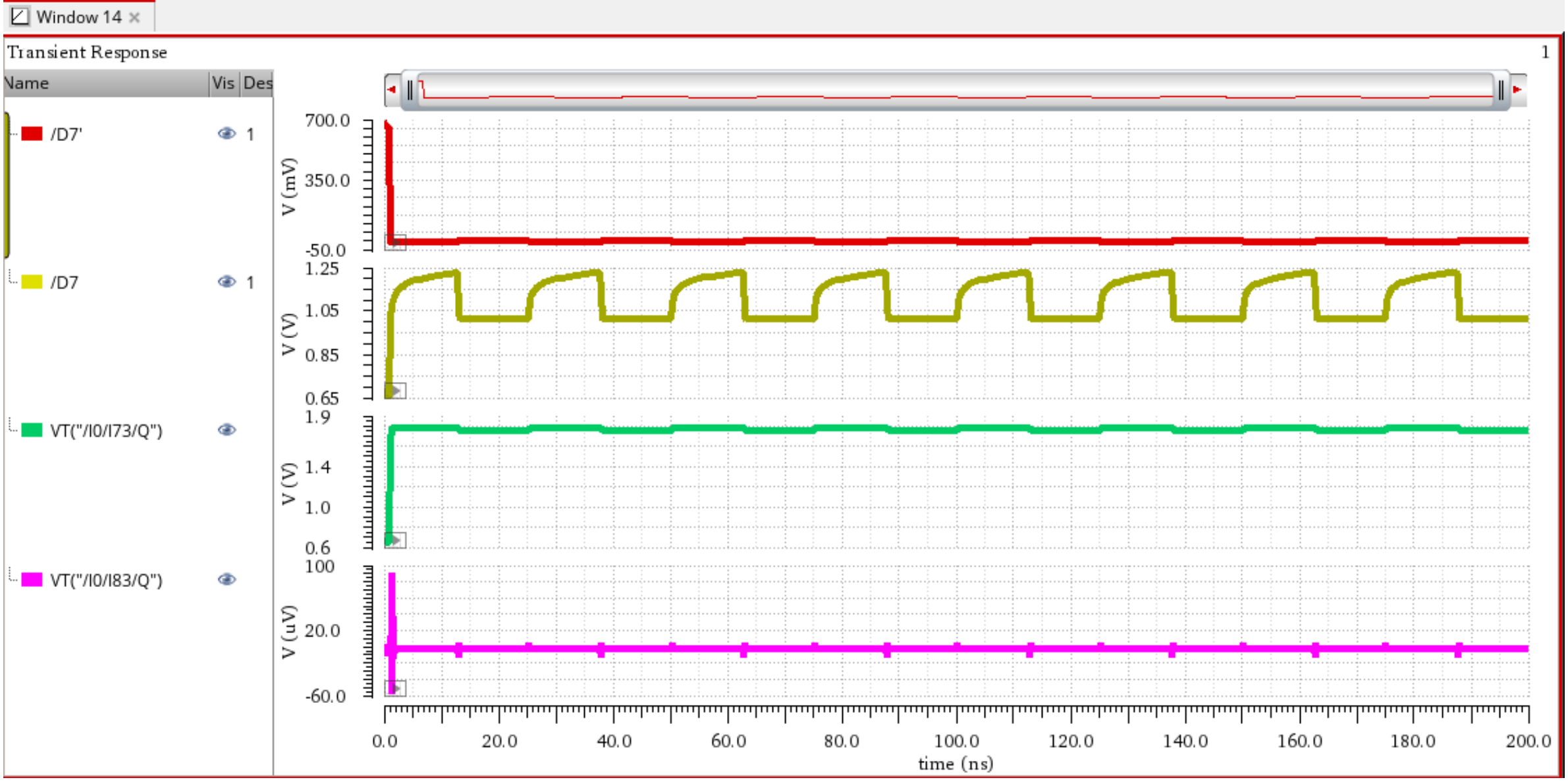
8X8 SRAM testbench



Decoder o/p (word line)

Read operation of 8x8 SRAM

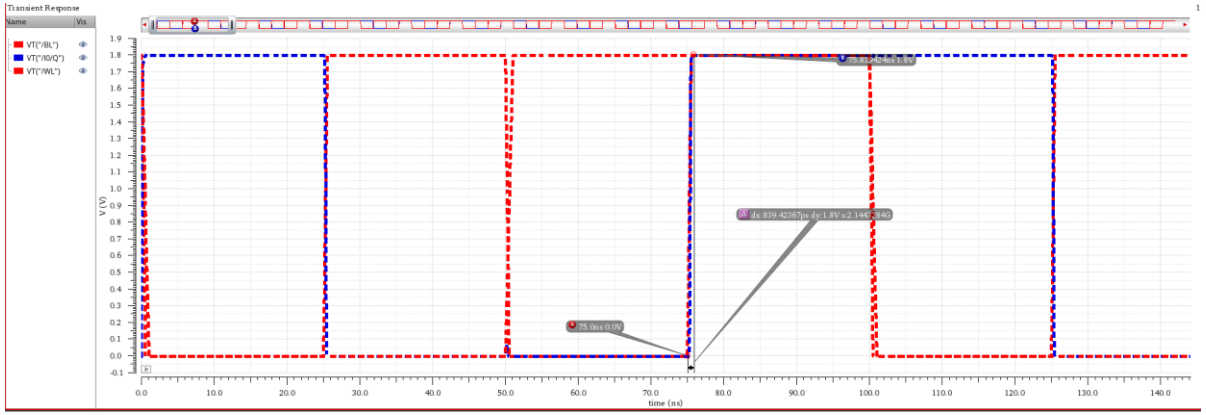




**we set cell 6 to be Q=0 to make sure that the read value is corre
and not read from cell 6.

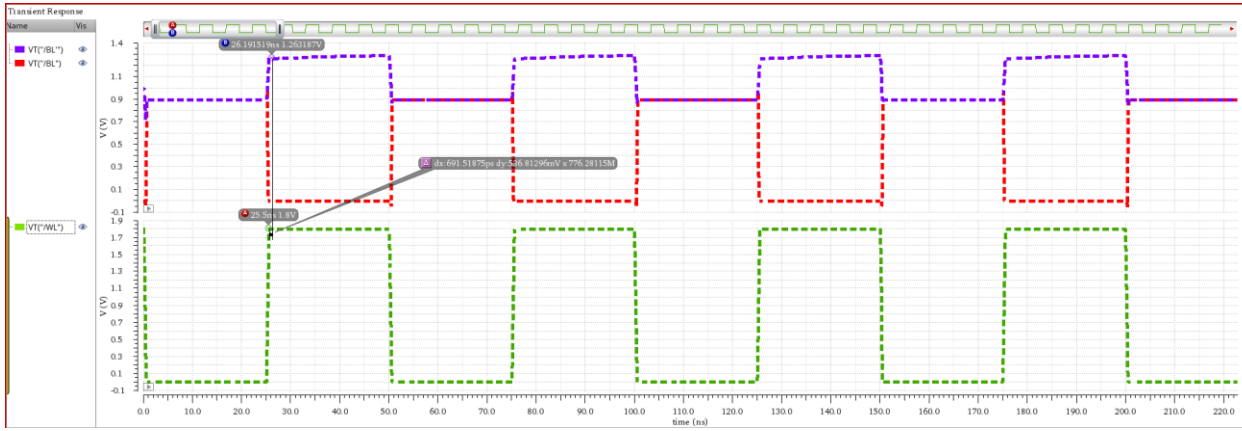
Performance check

- Write delay : It is the delay between the applications of the word line WL signal and the time at which the data is actually written.



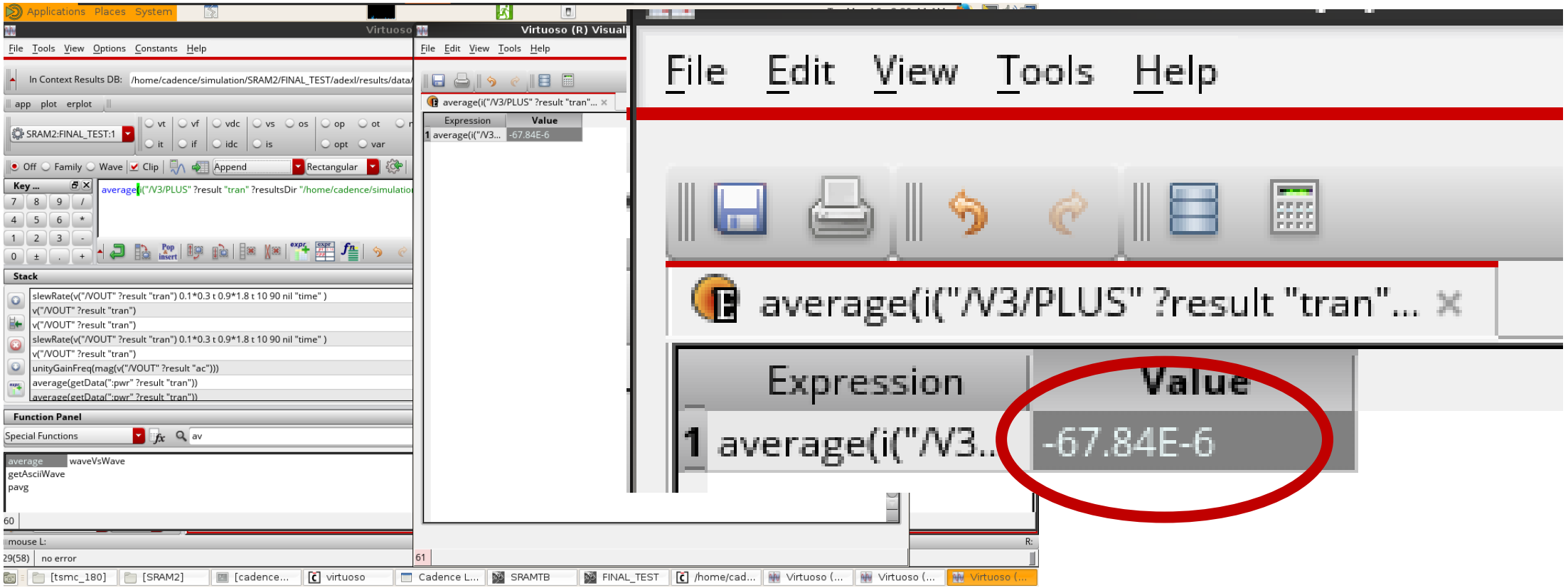
We can see the delay is actually **839.4ps**

- Read delay : It is the delay between the applications of the word line WL signal and the time at which the data is actually Read.



Read delay is about **691.6ps**

Total power dissipation
I = 67.84uA , P = 120uW , @f=20Mhz



System verification through MATLAB code:

```
% Initialize the SRAM
sram = zeros(8, 8);
% Define the decoder table
% Loop until the user chooses to exit
while true
% Prompt the user to choose a read or write operation
    operation = input('Enter "read" or "write" (or "exit" to quit): ', 's');

    % If the user chooses to exit, break out of the loop
    if strcmpi(operation, 'exit')
        break;
    end
```

```
% Prompt the user to enter a binary address
    address = input('Enter a 3-bit binary address (e.g. [0 1 0]): ');
% Check that the address is 3 bits long
    if numel(address) ~= 3
        error('Address must be 3 bits long.');
    end
% Convert the binary address to a row index
    row_index = bi2de(fliplr(address)) + 1;
% Perform the chosen operation
    switch lower(operation)
        case 'read'
```

Cont’d verification code:

```
% Prompt the user to enter a binary address
    address = input('Enter a 3-bit binary address (e.g. [0 1 0]): ');
% Check that the address is 3 bits long
    if numel(address) ~= 3
        error('Address must be 3 bits long.');
    end
% Convert the binary address to a row index
    row_index = bi2de(fliplr(address)) + 1;
% Perform the chosen operation
    switch lower(operation)
        case 'read'
% Read the data from the SRAM
            output_data = sram(row_index, :);

            % Display the output data from the read operation
            disp('Output data from read operation:');
            disp(output_data);
        case 'write'
```

Cont’d verification code

```
% Prompt the user to enter data to write to the SRAM
            data = input('Enter 8 bits of data to write to the SRAM (e.g. [1 0 1 0 1 0 1 0]): ');

            % Check that the data is 8 bits long
            if numel(data) ~= 8
                error('Data must be 8 bits long.');
            end
% Write the data from the user to the SRAM
            sram(row_index, :) = data;
% Display the SRAM contents after the write operation
            disp('SRAM contents after write operation:');
            disp(sram);
        otherwise
            error('Invalid operation specified.');
    end
end
```

Execution of the code

- the Sram is set to zero as initially now we will give the Sram some words as an input (write operation)

```
Enter "read" or "write" (or "exit" to quit): write
Enter a 3-bit binary address (e.g. [0 1 0]): [0 1 0]
Enter 8 bits of data to write to the SRAM (e.g. [1 0 1 0 1 0 1 0]): [1 1 0 0 1 1 0 0]
SRAM contents after write operation:
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     1     1     0     0     1     1     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
```

- We can see that we select the third row from the address and we write the data using the 8_bit data line

- After making write operations to set the Sram with different values

```
SRAM contents after write operation:
     1     1     1     1     1     1     1     1
     0     1     0     1     0     1     0     1
     1     0     1     0     1     0     1     0
     1     1     0     0     1     1     0     0
     1     1     1     1     0     0     0     0
     0     0     0     0     1     1     1     1
     0     0     1     1     0     0     1     1
     1     1     1     1     1     1     1     1
```

Cont’d Execution of the code

- Now if we do read operation we will select the address which will access the row and we will get 8_bit data

```
SRAM contents after write operation:
      1      1      1      1      1      1      1      1
      0      1      0      1      0      1      0      1
      1      0      1      0      1      0      1      0
      1      1      0      0      1      1      0      0
      1      1      1      1      0      0      0      0
      0      0      0      0      1      1      1      1
      0      0      1      1      0      0      1      1
      1      1      1      1      1      1      1      1

Enter "read" or "write" (or "exit" to quit): read
Enter a 3-bit binary address (e.g. [0 1 0]): [0 1 1]
Output data from read operation:
      1      1      0      0      1      1      0      0
```

- If we write once again in the same address the data of the sram will updated to the new data

```
Enter "read" or "write" (or "exit" to quit): write
Enter a 3-bit binary address (e.g. [0 1 0]): [0 1 1]
Enter 8 bits of data to write to the SRAM (e.g. [1 0 1 0 1 0 1 0]): [0 0 0 0 0 0 0 0]
SRAM contents after write operation:
     1     1     1     1     1     1     1     1
     0     1     0     1     0     1     0     1
     1     0     1     0     1     0     1     0
     0     0     0     0     0     0     0     0
     1     1     1     1     0     0     0     0
     0     0     0     0     1     1     1     1
     0     0     1     1     0     0     1     1
     1     1     1     1     1     1     1     1
```

FUTURE WORK !

- As we all know that we can make a larger MEMORY from smaller ones so we may built 256k more of less using this 8x8 SRAM.

- Attention!

- extending the memory size adding more the capacitance on the bit line, so we will need a **sense amplifier** to speed up the operation.

REFERENCE :

[1] V. K. Enumula, S. Gupta, R. Manchukonda, and N. Upadhyay, "Design and implementation of 16X8 SRAM in 0.25u SCMOS technology," [Online]. Available: http://plaza.ufl.edu/nirupamau/SRAM.pdf

[2] "A 8x8 compact SRAM in 65nm standard CMOS technology," [Online]. Available: https://www.researchgate.net/publication/267748232_A_8x8_compact_SRAM_in_65nm_standard_CMOS_technology

[3] "Novel design and implementation of 8 X 8 SRAM cell array for low power applications," [Online]. Available: https://www.researchgate.net/publication/267748232_Novel_design_and_implementation_of_8_X_8_SRAM_cell_array_for_low_power_applications

[4] K. Dhanumjaya, M. N. Giri Prasad, K. Padmaraju, and M. Raja Reddy, "Design of low power SRAM in 45 nm CMOS technology," International Journal of Engineering Research and Applications, vol. 1, no. 4, pp. 2040-2045, 2011.

[5] A. Agal, Pardeep, and B. Krishan, "6T SRAM cell: Design and analysis," [Online]. Available: https://www.ijert.org/research/6t-sram-cell-design-and-analysis-IJERTV2IS120136.pdf

[6] "An 8T-SRAM for variability tolerance and low-voltage operation in high-performance caches," [Online]. Available: https://www.researchgate.net/publication/224164300_An_8T-SRAM_for_Variability_Tolerance_and_Low-Voltage_Operation_in_High-Performance_Caches

[7] "Design of a novel hybrid CMOS non-volatile SRAM memory in 130nm RRAM technology," [Online]. Available: https://www.researchgate.net/publication/328792542_Design_of_a_Novel_Hybrid_CMOS_Non-Volatile_SRAM_Memory_in_130nm_RRAM_Technology

Thanks