8 X 2 SRAM MEMORY DESIGN

EEE5322 - VLSI CIRCUITS & TECHNOLOGY

Team 03

Graduate Team

Team Members :

Yashwanth Katta – (3451 7972)

Avinash Ayalasomayajula - (0699 6946)

Kshitij Raj – (1358 4965)

Under the Guidance of :

Professor Scott Thompson

Kelli Borowski (TA)

Breakdown of Work :

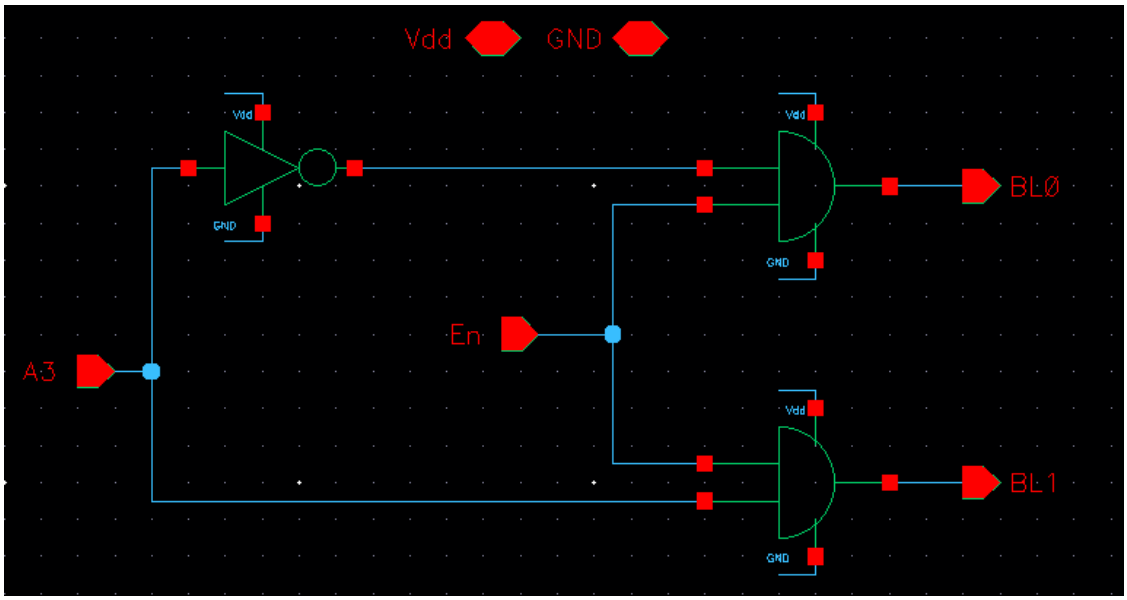
1. **Yashwanth Katta:**
   I was responsible for designing the Row and Column Decoders, the SRAM array and the pre-charge circuitry. I have designed their schematics and layouts. I have tested the working of the individual blocks designed by me to make sure they worked 100%. I connected all the individual block layouts to form the entire SRAM array. When it came to testing, I took part in getting the timing right for the simulation to get a valid output. I have made contribution in completion of the report.

2. **Kshitij Raj :**
   I did the sense amplifier part of the circuit. This included making the sense amplifier schematic, making the layout and extracting the symbol of this block. Then the testing of the sense amplifier was also done by me. I also performed checks on read write circuitry, checking the read simulation for the circuit to make sure that the read circuit is performing at par to give expected outputs when we were trying to read data from the respective SRAM blocks. I am also responsible for partial completion of the report.

3. **Avinash Ayalasomayajula :**
   I made the Read and Write Circuitry, both schematic and layout. I have tested the same to full accuracy. I made the layout considerations for the team to make sure we were designing the layout for minimum measurements. I checked the write simulation for the simulation output and helped in debugging in case of any errors. I also helped in completion of the report.

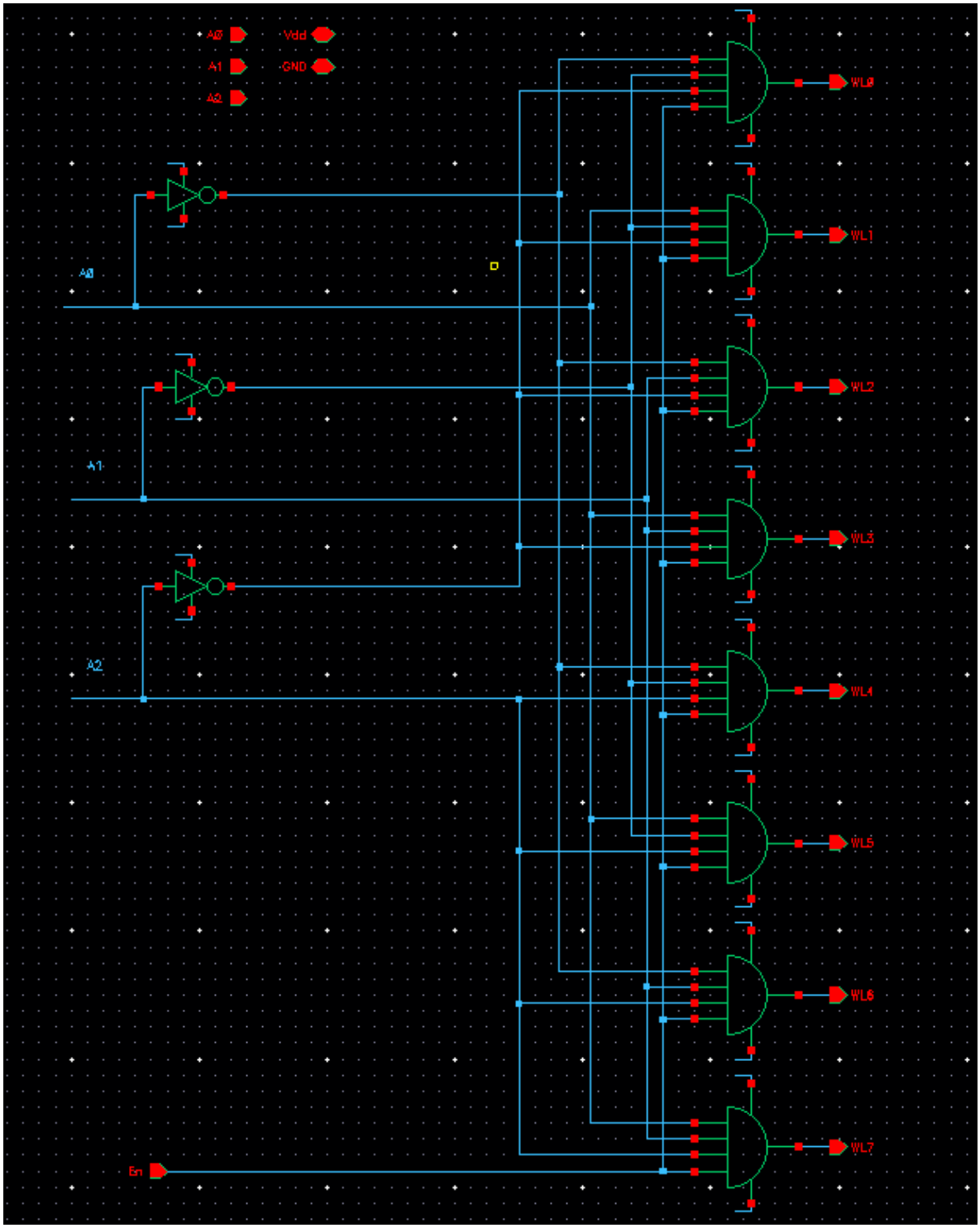
SCHEMATICS

1. COLUMN DECODER :



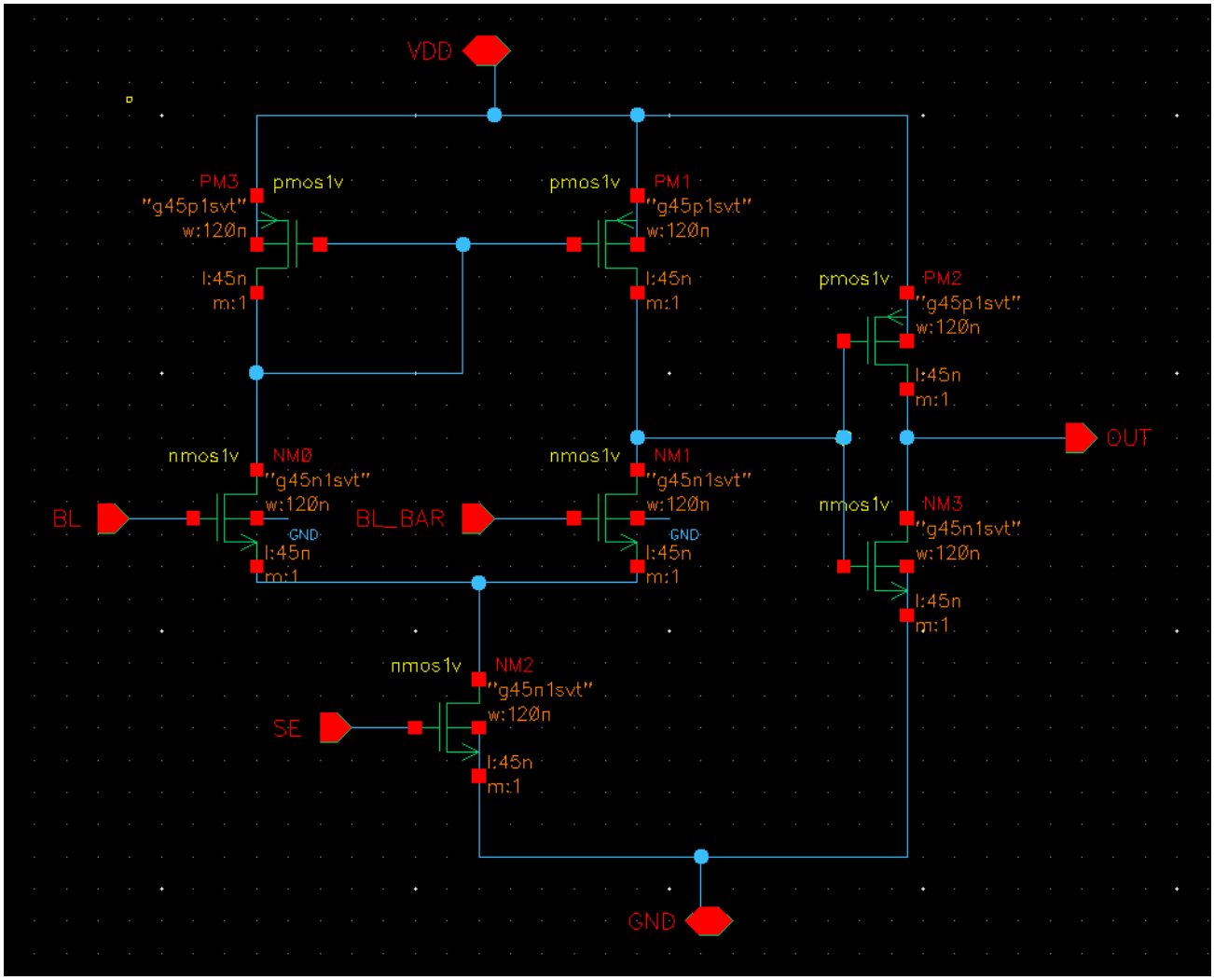
- The enable(En) pin here implicates the CLK1 in the complete circuit.

2. ROW DECODER :



- The enable(En) pin here implicates the CLK1 in the complete circuit.

3. SENSE AMPLIFIER SCHEMATIC :

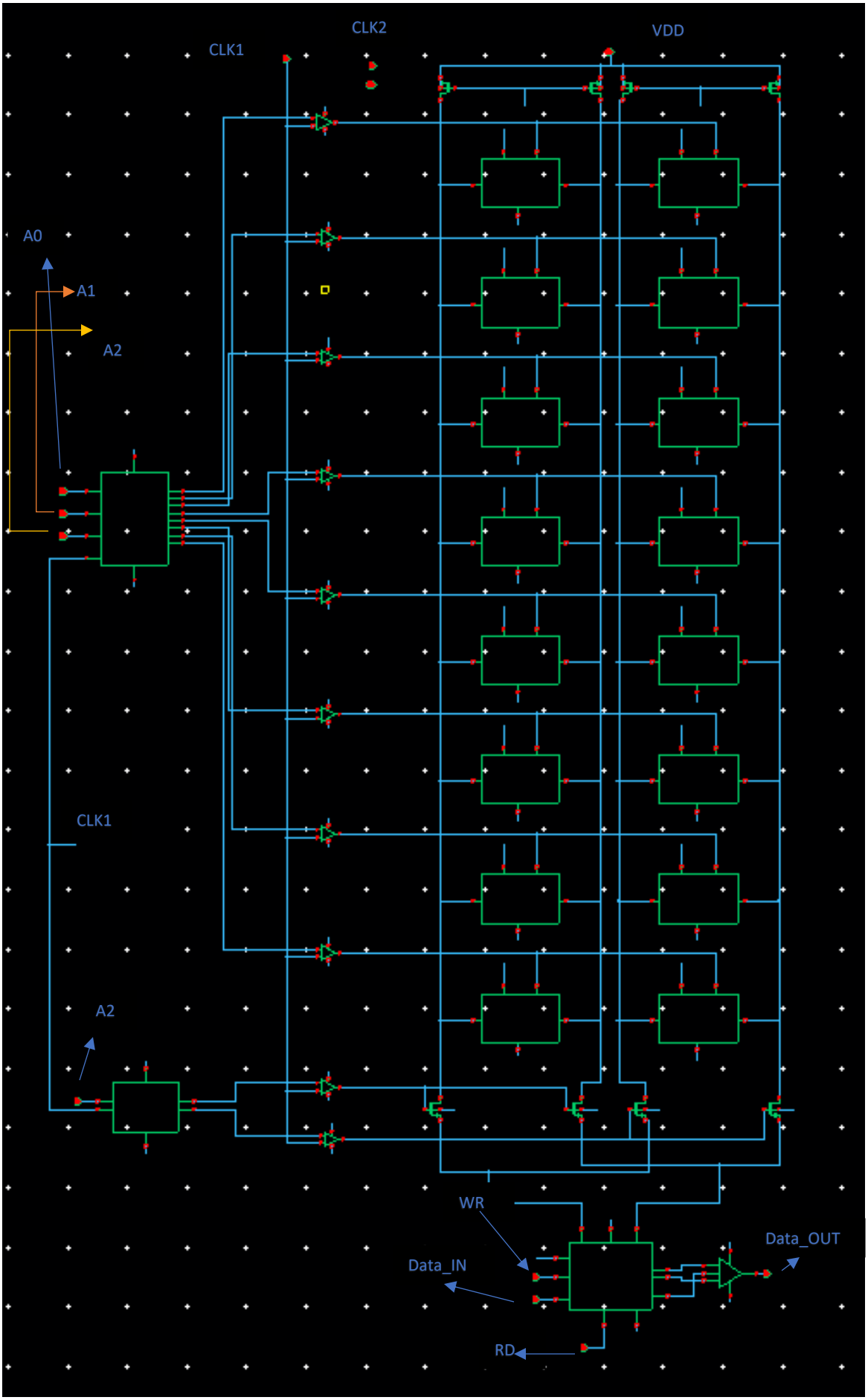


- The Sense Enable(SE) is taken from the Read signal that comes out of the buffer.
- We only want the SE pin to be HIGH when we want to read.

NOTE:

- All the enables are connected to CLK1.
- All blocks have one VDD and one GND connection in the next schematic (the blue lines popping out)

4. 8 X 2 SRAM ARRAY SCHEMATIC :



Schematics for Symbols Used :

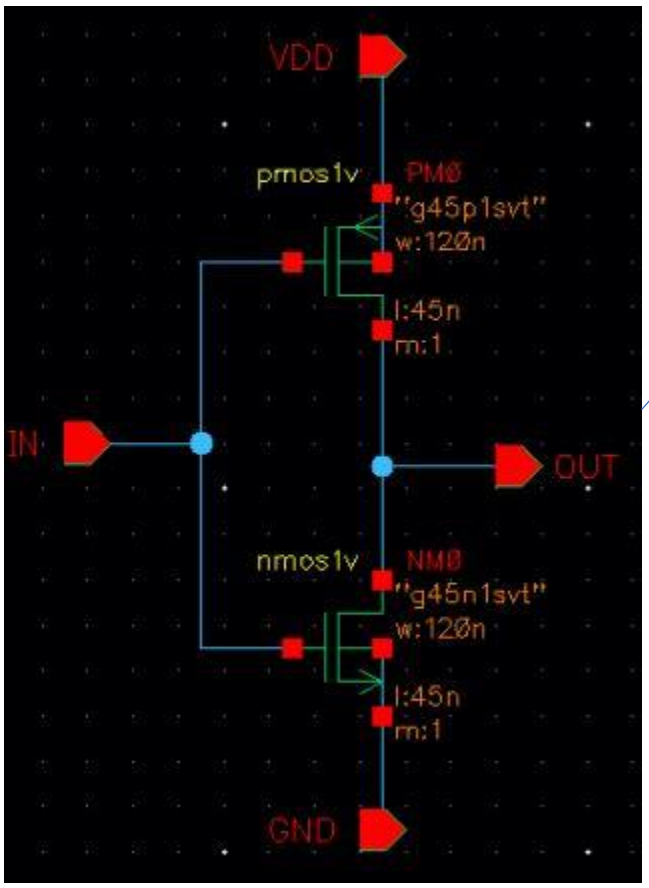


FIG : Inverter

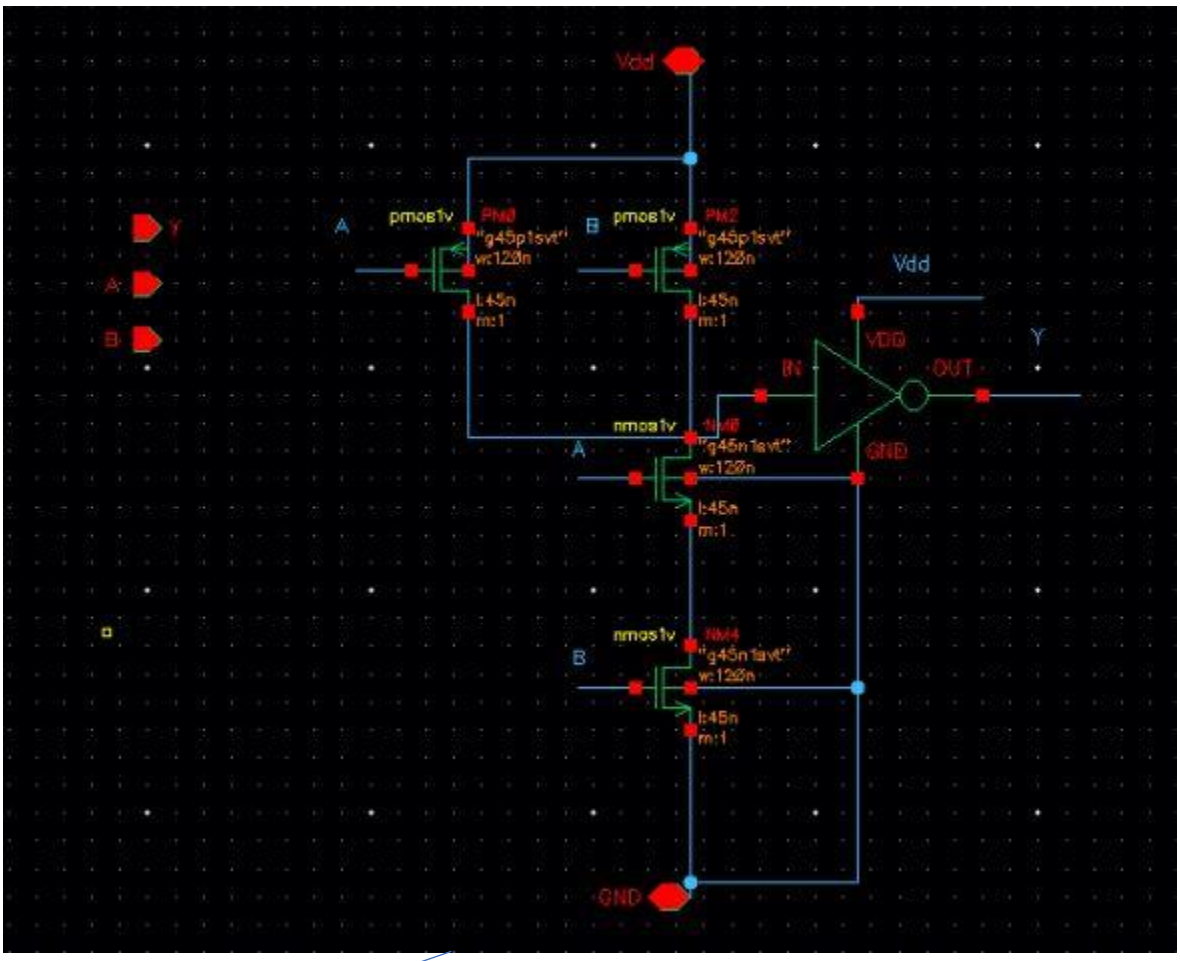


FIG : 2 IN AND GATE

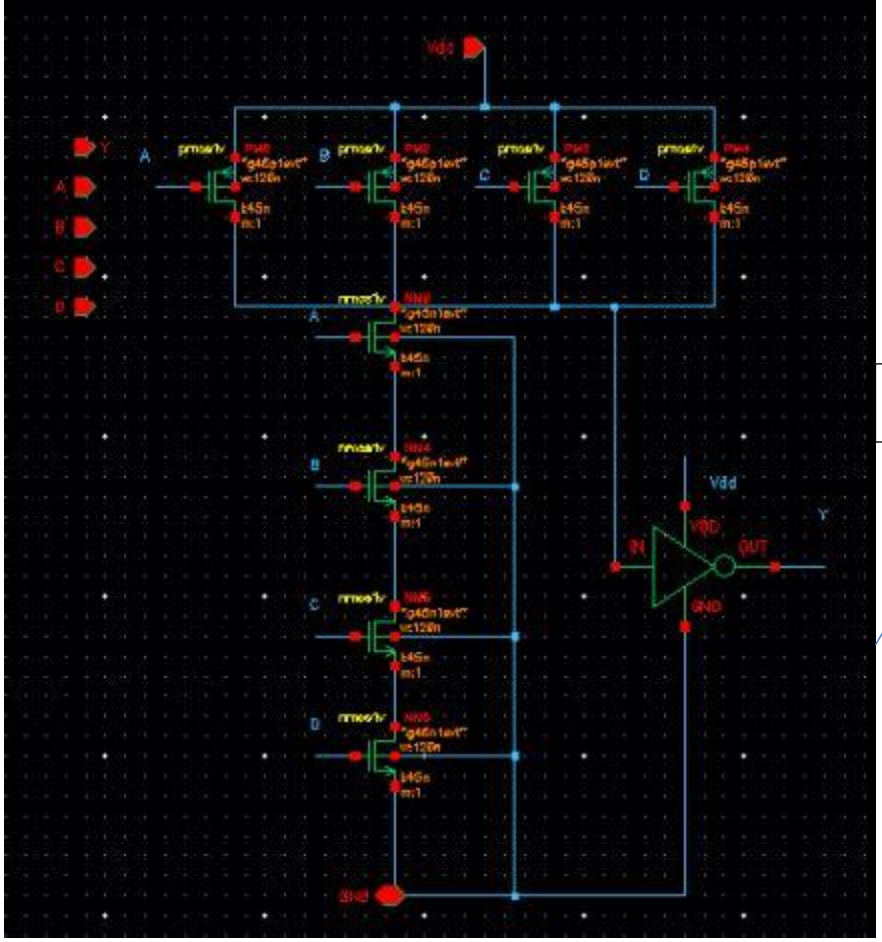


FIG : 4 IN AND GATE

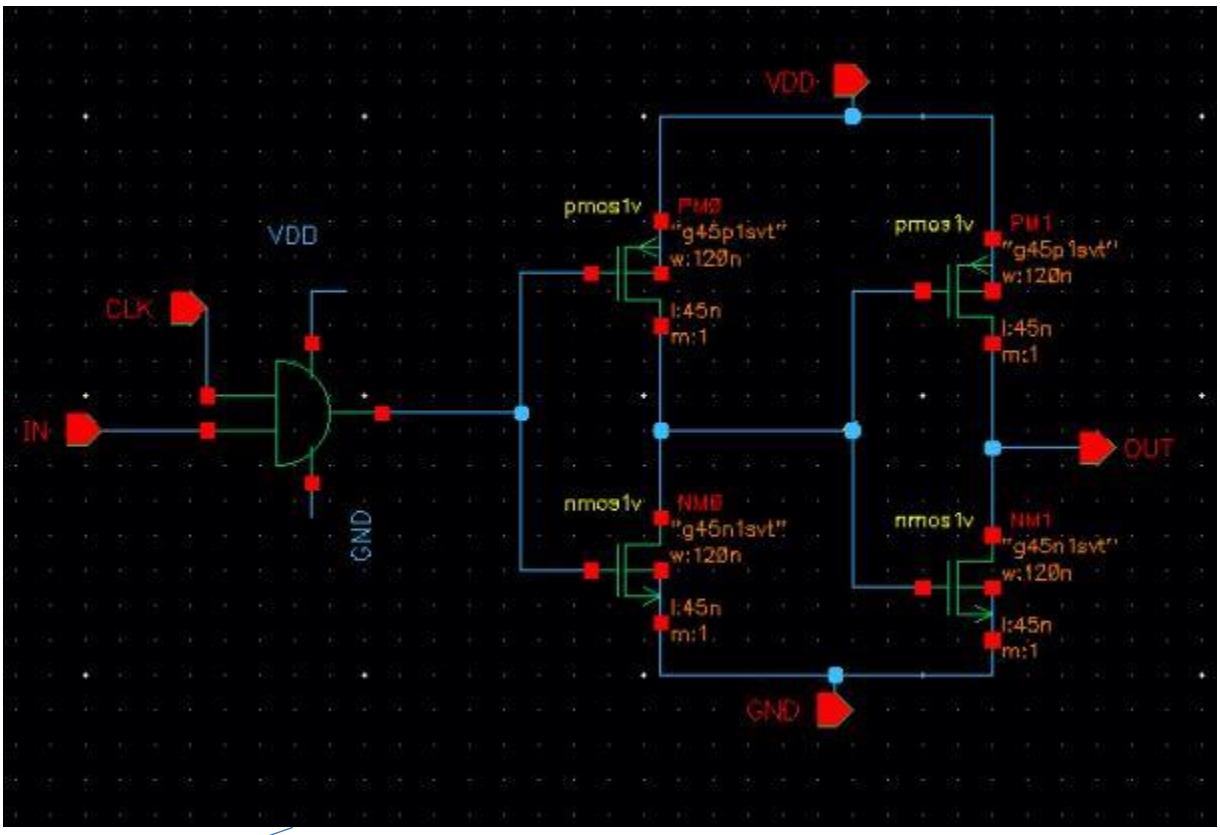


FIG : Buffer

SIMULATION SETUP :

- Testing the individual blocks was really challenging.
- Both CLK1 and CLK2 are operating at 100ns, i.e., around 10MHz frequency with 40% and 60% duty cycles.
  CLK1 is given an initial delay of 10ns, to make sure that both the clocks aren't ON at the same time.
- CLK1 is active HIGH operative, i.e., when it is HIGH then the buffers are enabled.
- CLK2 is active LOW operative, i.e., when it is LOW the Bit-lines are precharged.
- Following the clocks, we had to setup the Write, Read, Data_IN, A0, A1, A2 (Row Addresses) and A3 (Column Address Line).
- We are writing to and reading from four different cells. These cells are activated by wordlines – 0, 1, 4 and 5.
  So, A1 = 0 for all the times. A0 and A2 are operated at 5MHz and around 621KHz respectively with 50% duty cycles each.
- We are always using the cells in the first column for reading and writing. So, we are accessing Bitline 0 all the time. So, input A3 = 0 for all the cases.
- Now we are giving an input. Data_IN is operated at 25% duty cycle with a frequency of 5MHz with a delay of 10ns at the beginning.
- Write and Read signals are operated at 1.25MHz but are opposite to each other to make sure we are not writing and reading at the same time.

These summarize the simulation setup we have followed throughout this project for testing out SRAM array.

Below is a simulation output showing the simulated output for the above given values.

NOTE : Nets 128, 127, 124 and 123 are WL0, WL1, WL4 and WL5 respectively and net 118 is the BL0. Apparently, the node output of the SRAM units are quite fluctuating. So, instead we use BL0 for observing the write access times.

The simulation output shows that we are using SRAM cells – (0,0), (1,0), (4,0) and (5,0). The output proves our simulation to be successful because we are able to read a value after writing it to an SRAM.

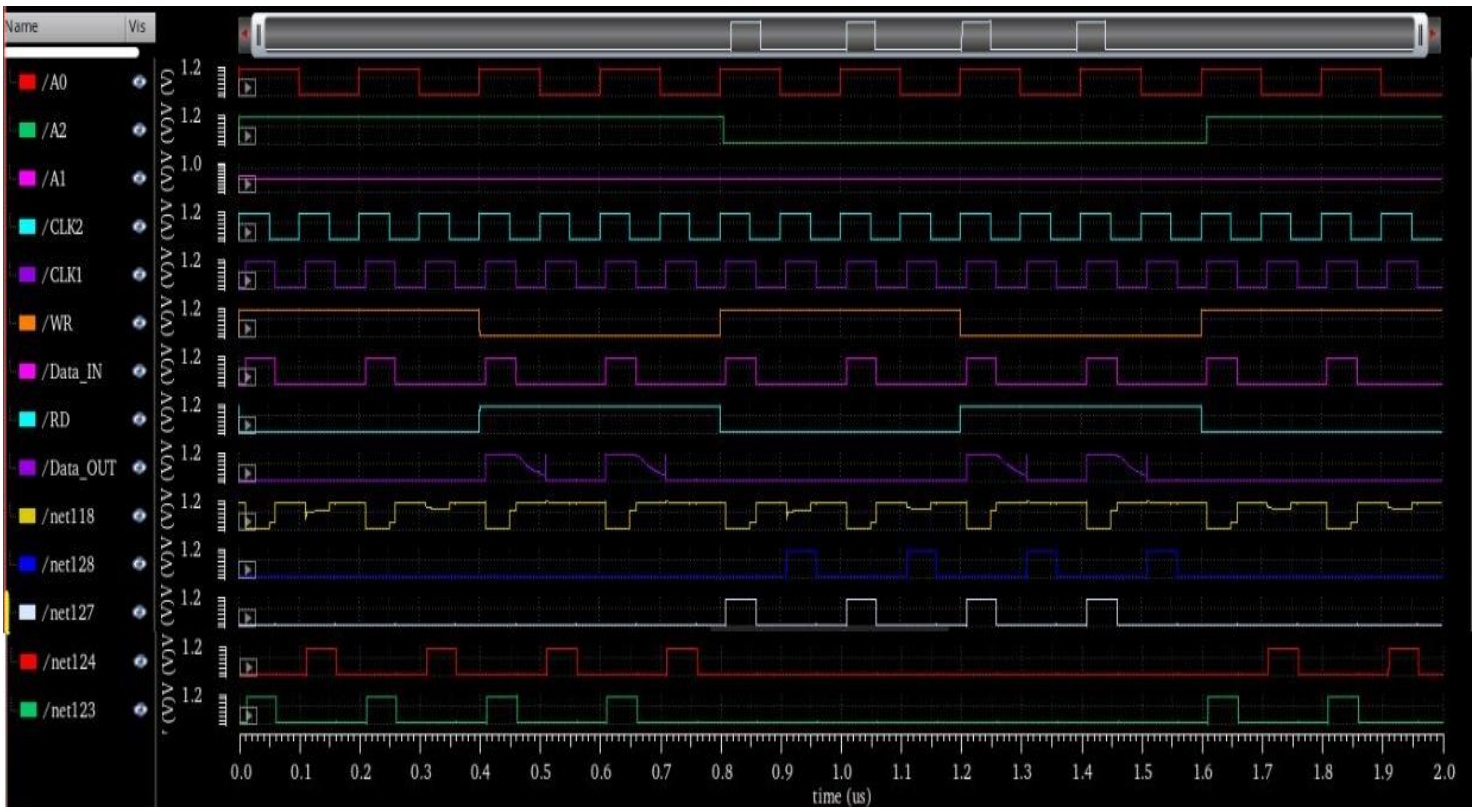


Fig: Simulation Output for the SRAM Array.

- We are first switching on WL4 and WL5. We are writing '1' to SRAM(4,0) and '0' to SRAM(5,0). These happen when we have Write = '1', CLK1 = HIGH and Data_IN is active.
- The same thing happens for SRAM(0,0) and SRAM(1,0).

TIMING RESULTS :

1. Pre-charge Time :

Pre-charge time is the time taken for the bit-line to rise to a certain potential when clock 2 is enabled LOW. Initially when the circuit starts, we make sure that CLK2 goes LOW and charges the Bitlines lines to 1.1V. This enables us to use the Sense Amplifier for getting the output. The precharge circuitry is being operated by PMOSs, so for them to be ON the input, CLK2, has to be LOW.
That's the reason we have CLK2 as active LOW input. So, we calculate the precharge time when CLK2 goes LOW. From the graph we measure the precharge time.
For CLK2 HIGH would implicate 0V because it is active LOW. So, basically we take the values when both the components are at their final value.

Fig: Waveform for pre-charge time

Precharge Time = 9.08ns

2. Write Simulation :

For a write to happen, the Write signal should go HIGH when CLK1 goes HIGH because, when CLK1 is HIGH buffer is enabled.
Below is the figure for Write 0 case :

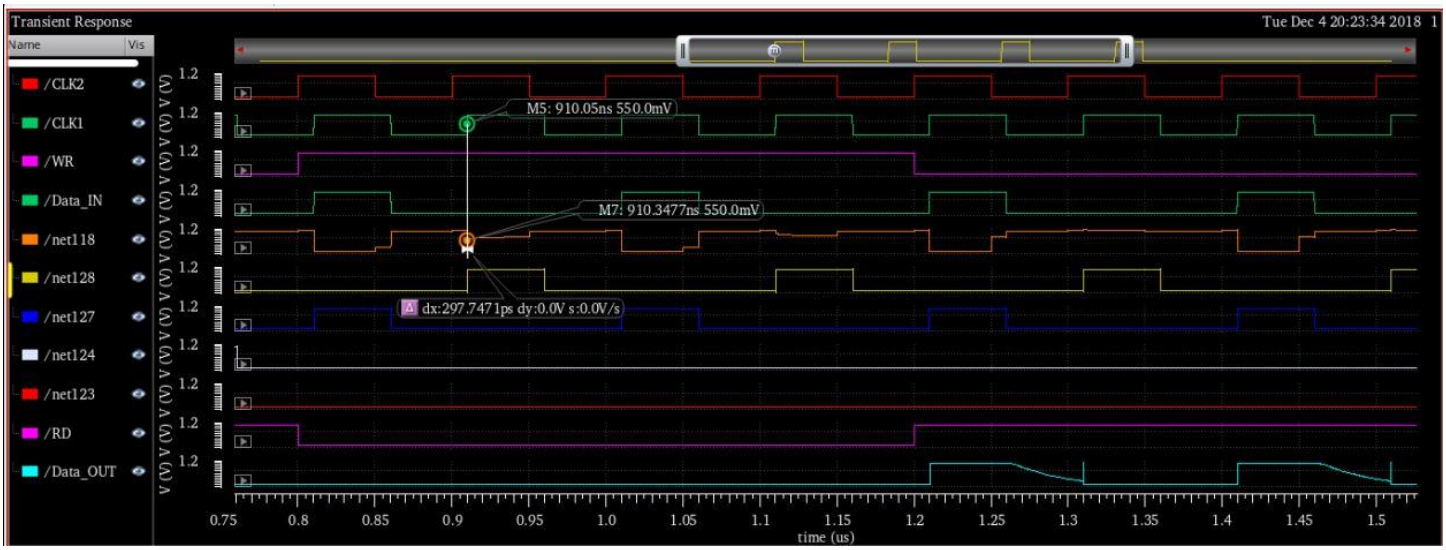


Figure : Waveform for Write 0

As we see from the graph, we are writing '0' to SRAM(0,0) through net128 (WL0). So, as we see that when Write is HIGH and CLK1 is HIGH, Data_IN is '0' is being written to the SRAM(0,0). The value of the Bitline drops to an enough value that gives it a logic 0. This ensures that the value being stored in the SRAM to be '0'.
For measuring the write time for storing '0', we must measure when CLK1 goes to 50% of its final value and Bitline drops to 50% of its final value.

Write time for 0 = 297.75ps

Now let us look at how we are going to write '1' to the SRAM(1,0). Below graph shows the simulation for writing'1'. The measurement is done when CLK1 and Bitline reach 50% of their final value, when Data_IN is HIGH and Write is HIGH. This is now checked for net127 (WL1) being active and the cell which we are writing to is SRAM(1,0).

In the below simulation, the value of the Bitline reaches it's 50% final value quite late compared to when a '0' is being written. But the value is written before the Data_IN value changes which ensures that the value is being written to the SRAM.

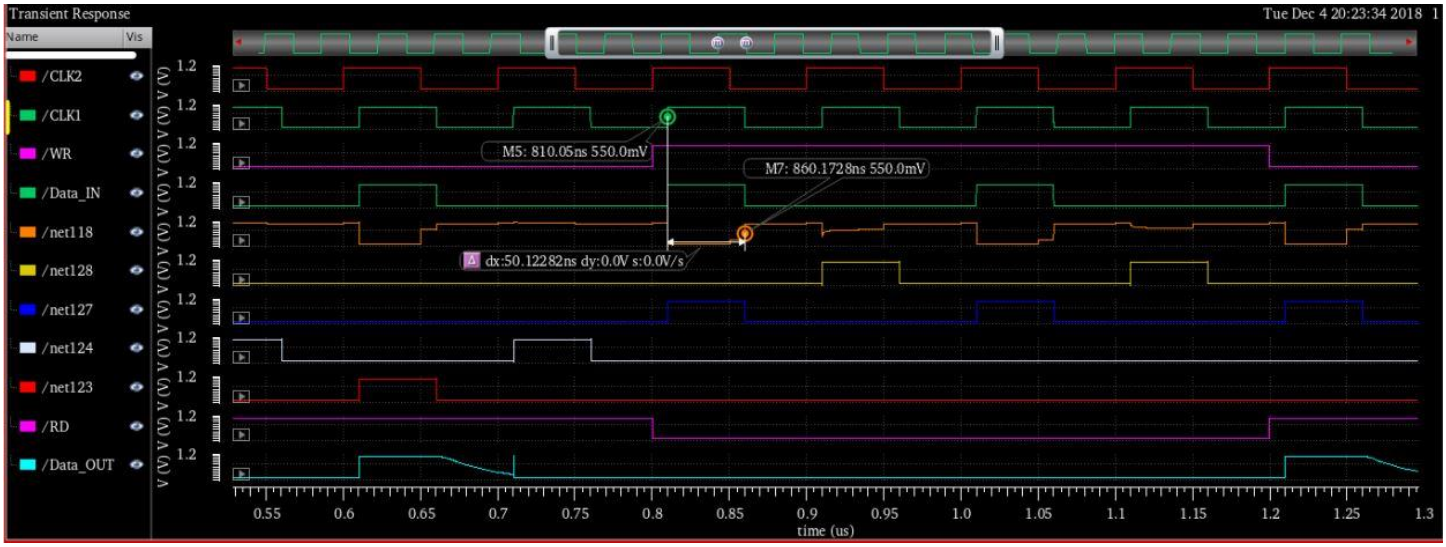


Figure : Waveform for Write 1

Write time for '1' = <u>50.12ns</u>

- Now, let us investigate the write operation into four cells – SRAM[(0,0) , (1,0), (4,0) and (5,0)]
- From the below image we can see that we are writing '0' to SRAM [(0,0) and (4,0)] and '1' to SRAM[(1,0) and (5,0)].
- This happens when we have CLK1, Write and the respective Wordlines HIGH.
- Data_IN is given and when there is a write signal and wordline HIGH we write that respective value to that wordline.
- We aimed at first writing to all the cells and then reading it, thus giving us a clarity in terms of seeing the output.
- To verify if the value we wrote is correct or not we have to investigate the Read Simulation results.

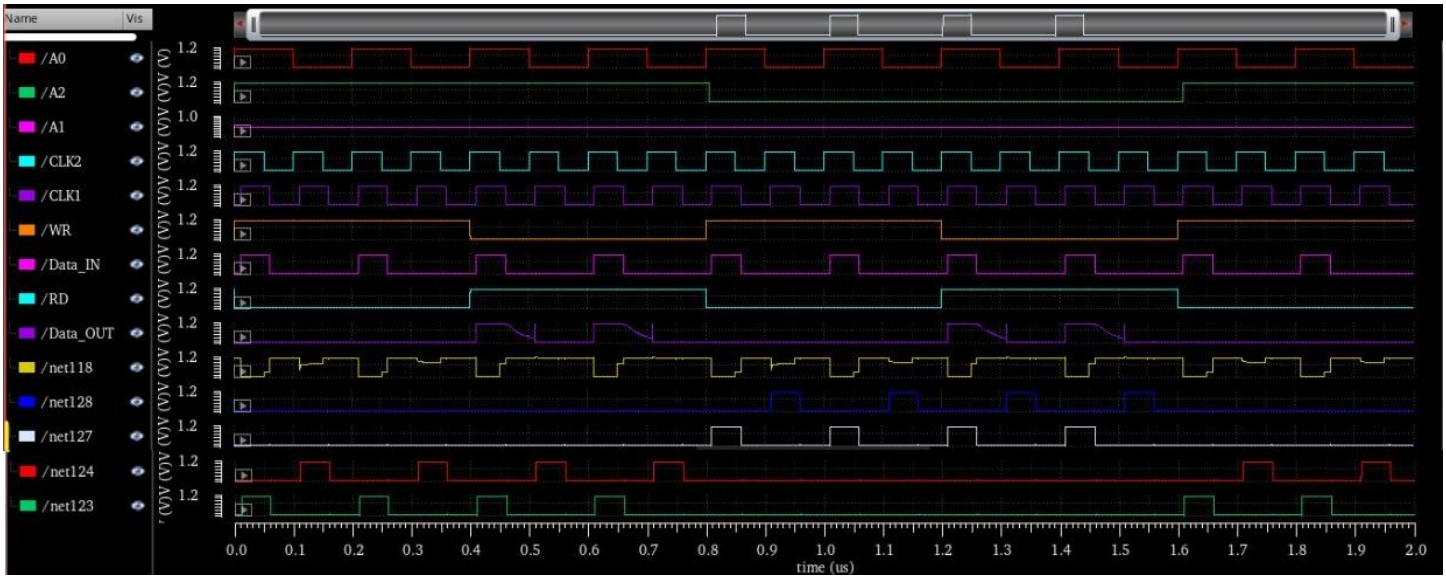


Fig: Simulation Output

3. Read Simulation :

For a read to happen, the Write signal should go LOW and Read signal should go HIGH when CLK1 goes HIGH because, when CLK1 is HIGH buffer is enabled and we don't want to write and read at the same time.
Now let us first look into the Read Times and then comeback to see if what we wrote earlier is being shown at the output of the sense amplifier while reading.

Below is the figure for Read 0 case :

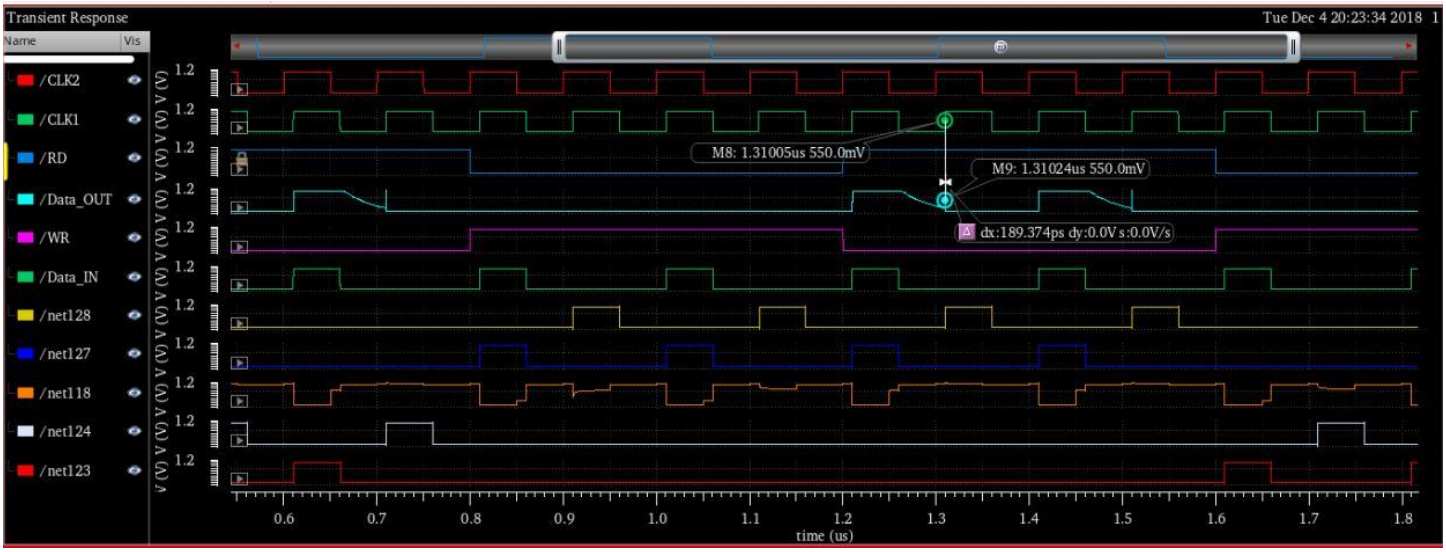


Figure : Waveform for Read 0

As we see from the graph, we are reading '0' from SRAM(0,0) through net128 (WL0). So, as we see that when Read is HIGH and CLK1 is HIGH, Data_IN is '0' is being read from the SRAM(0,0). The data output is showing logic '0' thus making sure that we are reading 0 from net128.
For measuring the read time for storing '0', we have to measure when CLK1 goes to 50% of its final value and output drops to 50% of its final value (in this case the output's final value is 0).

- Although there is a spike which shows that output is changing, we decided to measure from that point because that seemed more accurate because that is the point where we change clock and see that there is a value being read which is '0'.

Read time for 0 = 189.374ps

Now let us look at how we are going to read '1' from the SRAM(1,0). Below graph shows the simulation for reading '1'. The measurement is done when CLK1 and Data_OUT reach 50% of their final value, when Data_IN is HIGH and Read is HIGH. This is now checked for net127 (WL1) being active and the cell which we are reading from is SRAM(1,0).

As we see in the figure the value being read from SRAM while READ = ‘1’ is logic 1 which ensures that our write operation was successful.

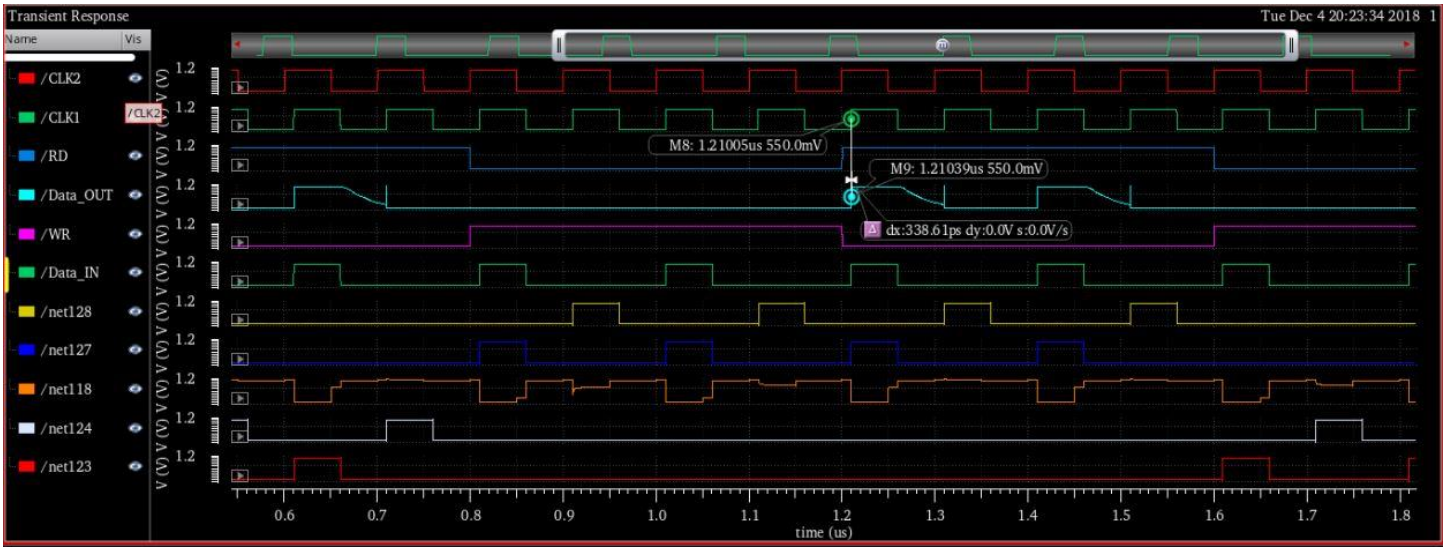


Figure : Waveform for Read 1

Read time for ‘1’ = <u>338.61ps</u>

Now let us look at the whole picture.

- Let us look at all the four cells we are writing and reading from - SRAM[(0,0) , (1,0), (4,0) and (5,0)]
- These correspond to net128, net127, net124 and net123 Wordlines.
- Initially data ‘1’ was written to SRAM[(1,0) and (5,0)]. We did calculate the Write times for that as well.
  In the first half as we see we are writing and reading from SRAM[(5,0)]. We see that Data_OUT is going HIGH as long as we have our WL5 HIGH. This gives us definite proof that we were successful in writing to ‘1’ to that corresponding SRAM cell. The same follows for SRAM[(1,0)]. We get to see that ‘1’ is being read.
- Let us now see what is happening when we write and read ‘0’. We are writing ‘0’to SRAM[(0,0) and (4,0)].
  Same goes here as well. When the respective Wordline is HIGH we get to see that for Read = HIGH logic and CLK1 enabled, we see a output LOW. The spike there is showing that while switching the clock we are reading logic ‘0’ from the corresponding cell.

  Thus, we have sufficient data and results to conclude that we were successful in creating a SRAM array that can store data according to our simulation.

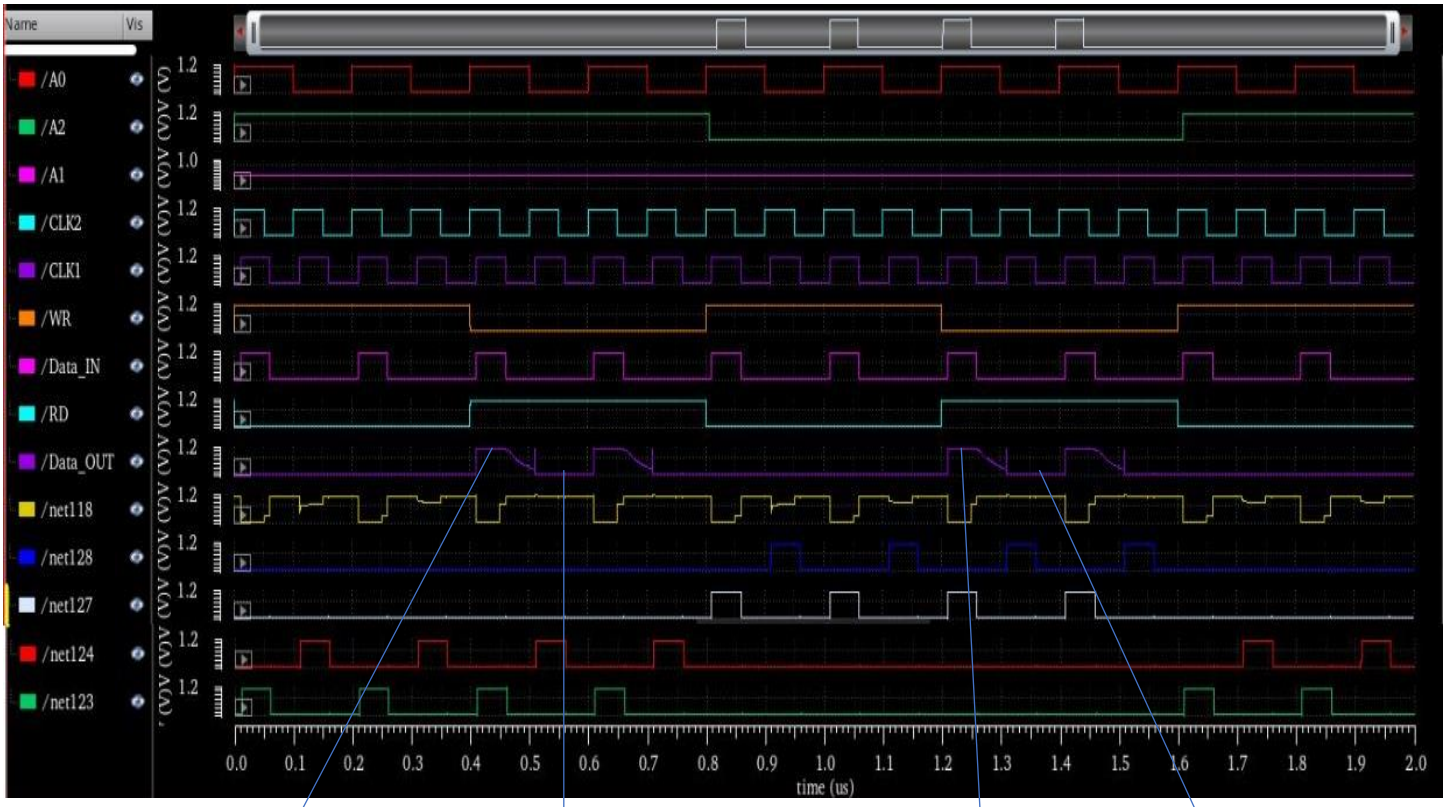


Fig: Output Graph

Reading 1 from SRAM(5,0)

Reading 0 from SRAM(4,0)

Reading 1 from SRAM(1,0)

Reading 0 from SRAM(0,0)

LAYOUT VERIFICATION :

Layout Considerations :

These are the basic rules for 45nm process taken into consideration for designing layouts.

Poly Rules:

1. Gate width = 0.045 μm
2. Poly extension from the active = 0.12 μm
3.

Active Width:

1. Active width = 0.12 μm
2. Min. Active Area to Implant enclosure = 0.07 μm
3. Min. Active Area to N-Well = 0.16 μm

Metal 1 Rules:

1. Minimum width =0.01 μm
2. Min. Metal 1 to Mental 1 = 0.06 μm
3. Min. Metal 1 to Metal 1 spacing if width of Metal 1 > 0.1 = 0.1 μm
4. Min. Metal 1 to Contact enclosure on two opposite sides of the contact =0.03 μm

Metal 2 Rules:

1. Minimum width =0.08 μm
2. Min. Metal 2 to Mental 2 = 0.07 μm
3. Min. Metal 2 to Metal 2 spacing if width of Metal 2 > 0.1 = 0.15 μm

Metal 3 Rules:

1. Minimum width =0.08 μm
2. Min. Metal 3 to Mental 3 = 0.07 μm
3. Min. Metal 3 to Metal 3 spacing if width of Metal 3 > 0.1 = 0.15 μm

Metal 4 Rules:

1. Minimum width =0.08 μm
2. Min. Metal 4 to Mental 4 = 0.07 μm
3. Min. Metal 4 to Metal 4 spacing if width of Metal 4 > 0.1 = 0.15 μm

Contact Rules:

1. Contact Width/Length = 0.06 μm
2. Minimum Poly to Contact enclosure = 0.03 μm

Metal 1 to Poly Via Rules:

1. Metal 1 to Poly Via Width = 0.07
2. Min. Metal 1 to Via 1 enclosure on both sides = 0.03 μm

Via 1 Rules:

1. Via 1 Width = 0.07
2. Min. Metal 1 to Via 1 enclosure on both sides = 0.03 μm

Via 2 Rules:

1. Via 2 Width = 0.07
2. Min. Metal 2 to Via 2 enclosure on both sides = 0.03 μm

Via 3 Rules:

1. Via 3 Width = 0.07
2. Min. Metal 3 to Via 3 enclosure on both sides = 0.03 μm

Design Practices :

- We made sure that all the **poly, metal 3 and metal 4** are in a single direction.
- We used a **wide cell SRAM**.
- All the blocks have been designed to the **minimum layout requirements** individually.
- While connecting the all the layouts, we connected a block to the array block, checked for the DRC and LVS with the modified schematic and when that worked, we moved on adding other blocks till completion. This helped in debugging.

Issues :

- We faced issues while designing for the minimum requirements. But that later got sorted when we looked up the DRC file provided.
- While designing the SRAM array, we faced difficulty in adjusting the NMOS and PMOS for the 8 X 2 array. We had a problem with the LVS and for rectifying it we used the 'error display' option. This made it a bit easy.
- Checking if all the bodies were connected became necessary for every connection we made.
- Connecting the decoder to the buffers and from there to the wordlines had tested both patience and focus. We had to take the consideration of area as well.
- Metal spacings for different metal widths were different and so while running DRC we got this error.
- Different metals have different metal widths and we had to consider them as well.
- When one via is placed on another we had to make sure a minimum metal area had to be laid over the via overlap.
- The contact has to have a metal over it and it should have either an active or gate under it.

The rest were all minor issues regarding few missing connections, minimum spacing etc.

1. SRAM Cell :

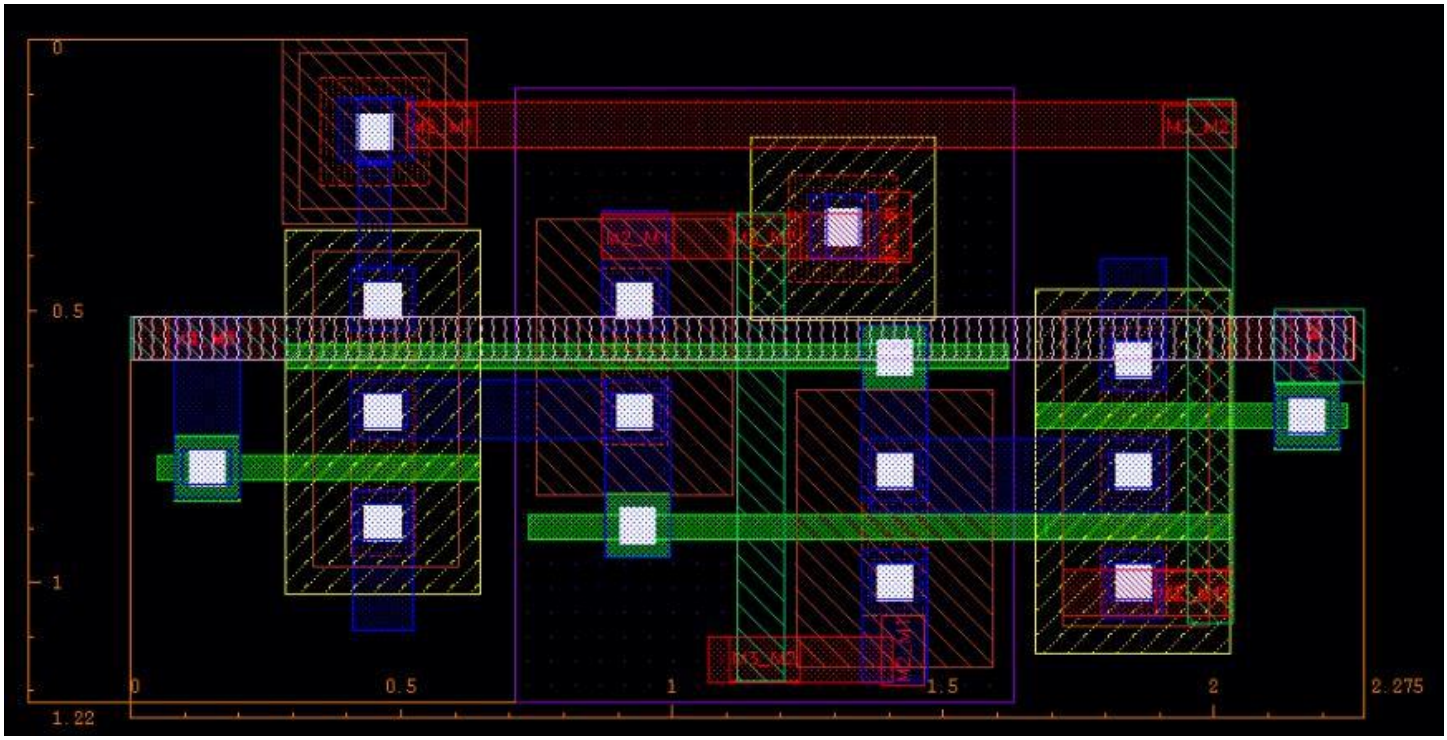


Fig : SRAM Cell Layout

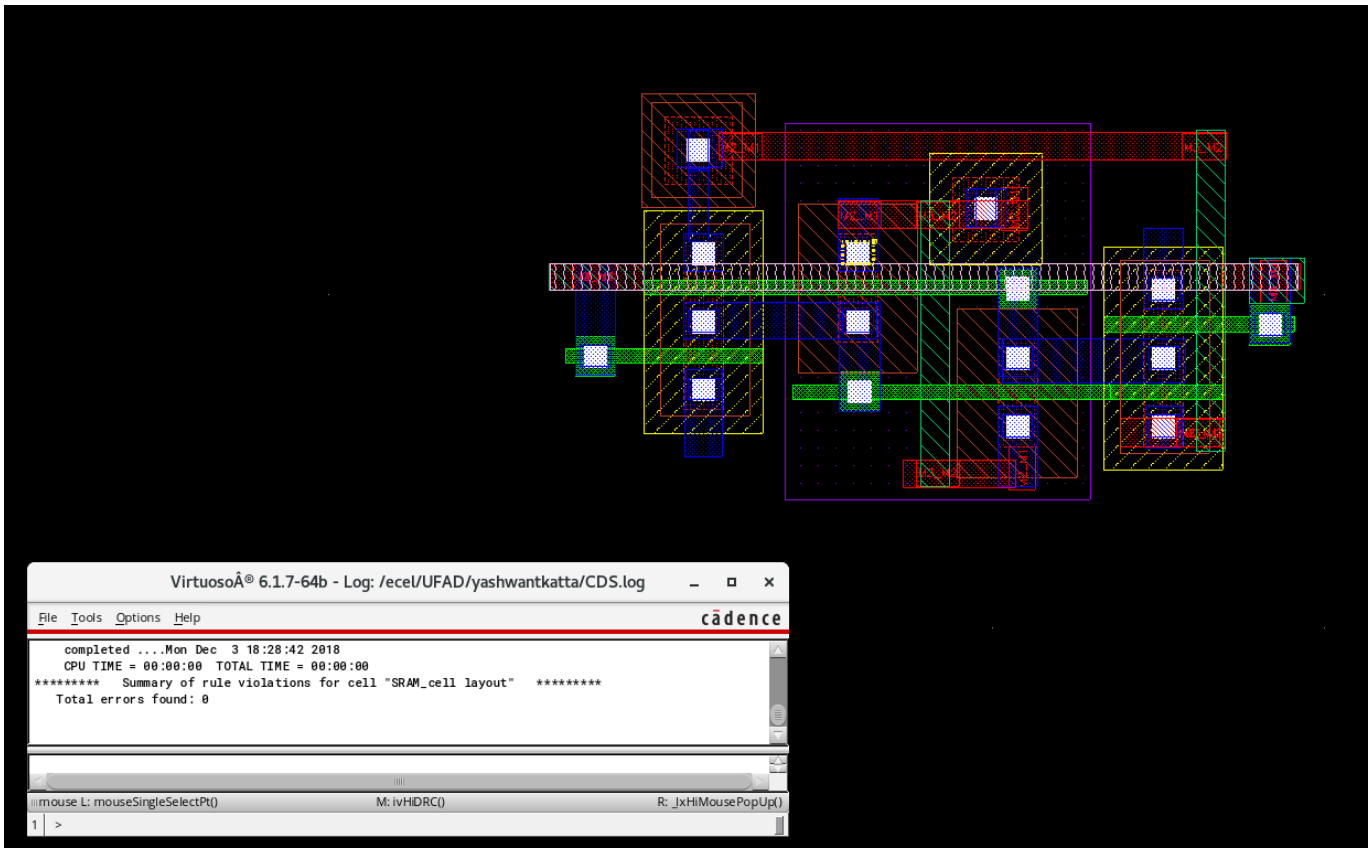


Fig : SRAM Cell DRC

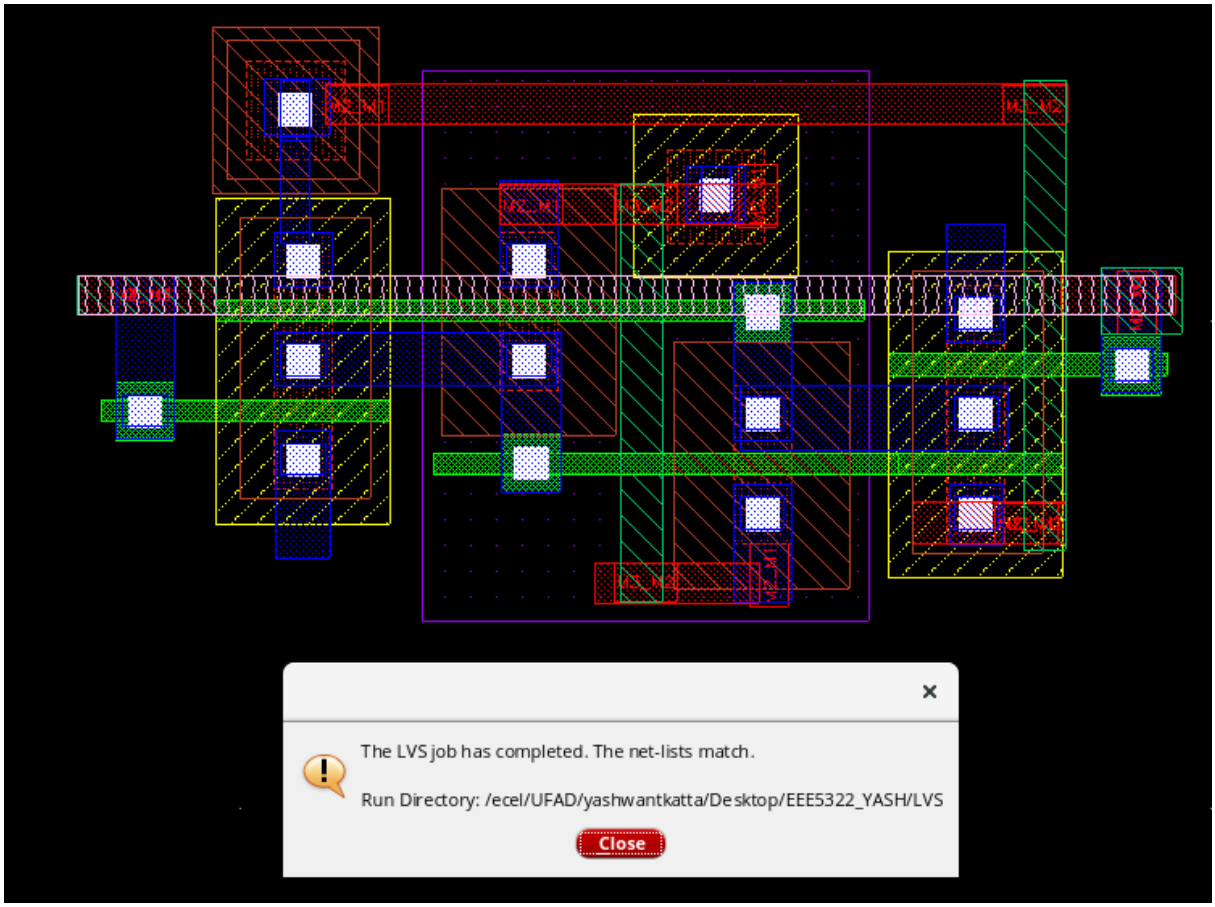


Fig : SRAM Cell LVS

- **Area of the single SRAM Cell = 2.275 × 1.22 $(um)^2$**
  = <u>2.7755 $(um)^2$</u>

2. Column Decoder :

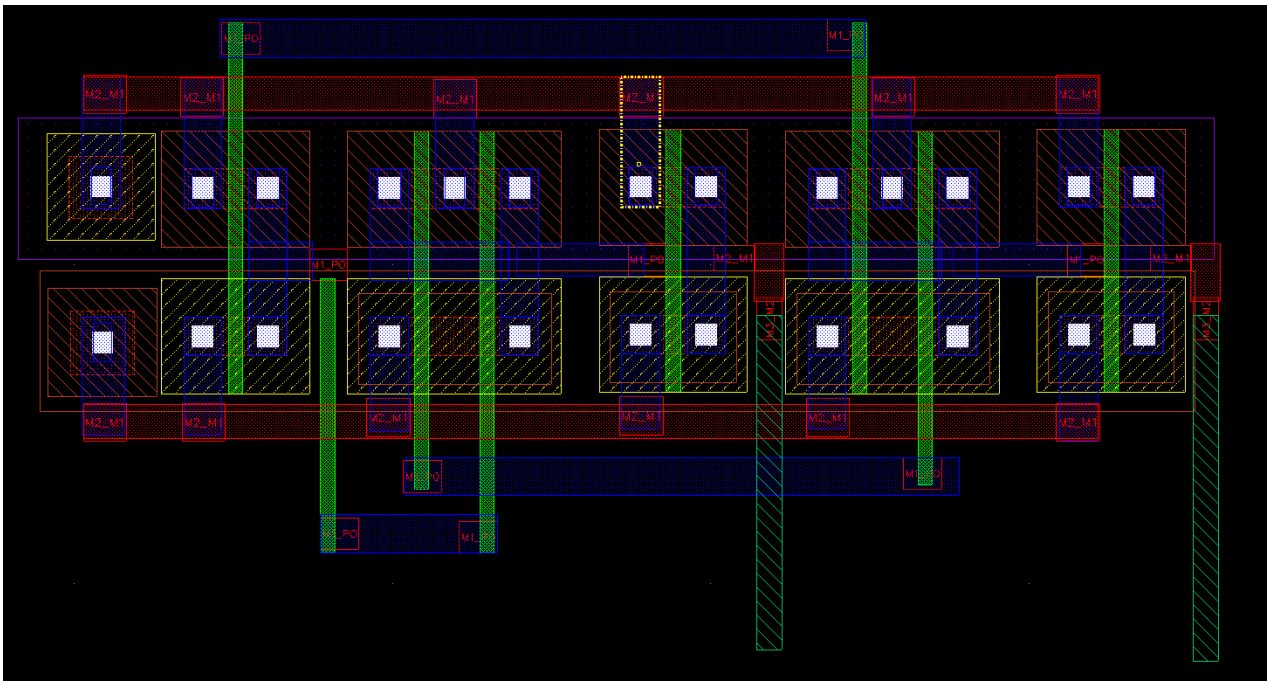


Fig : Column Decoder Layout

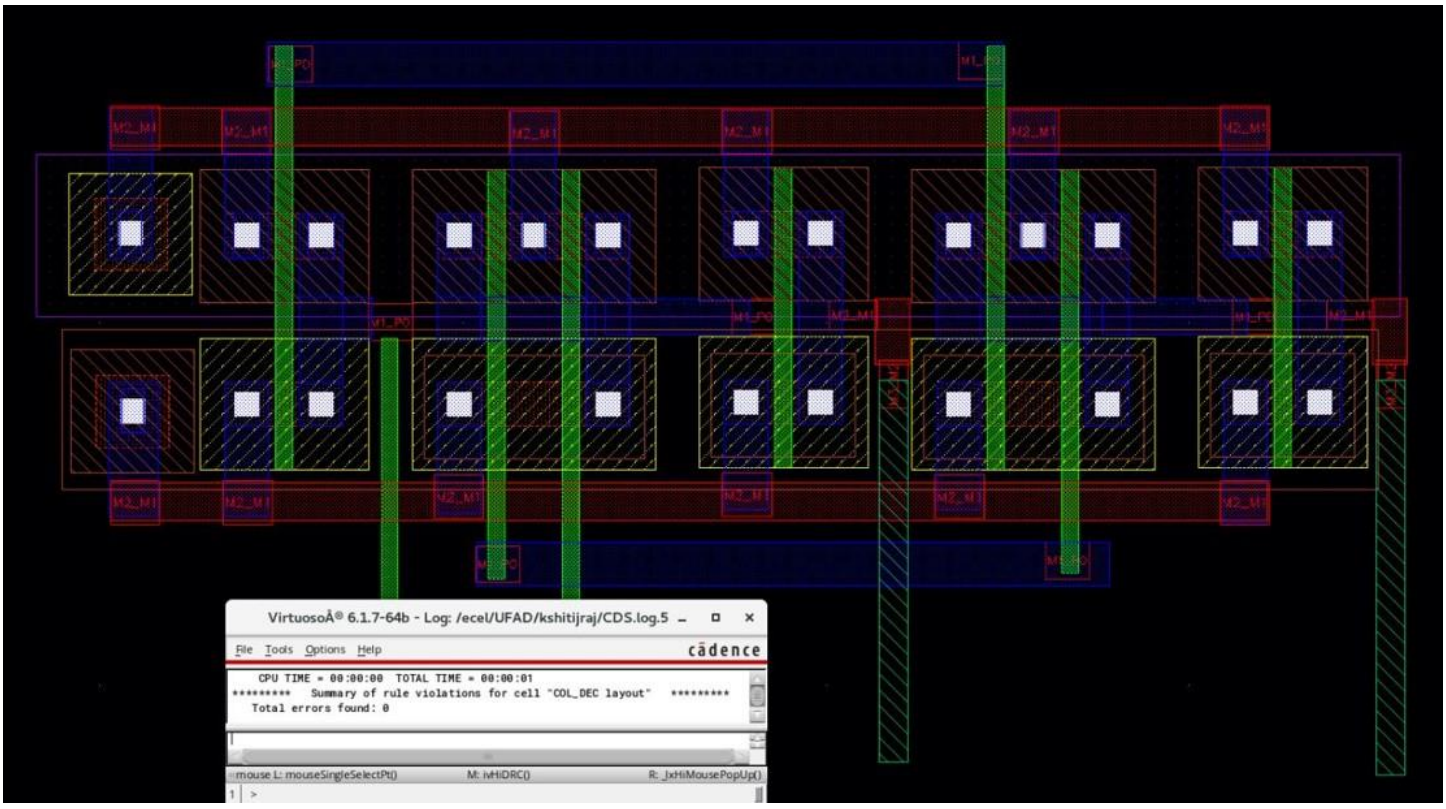


Fig : Column Decoder - DRC

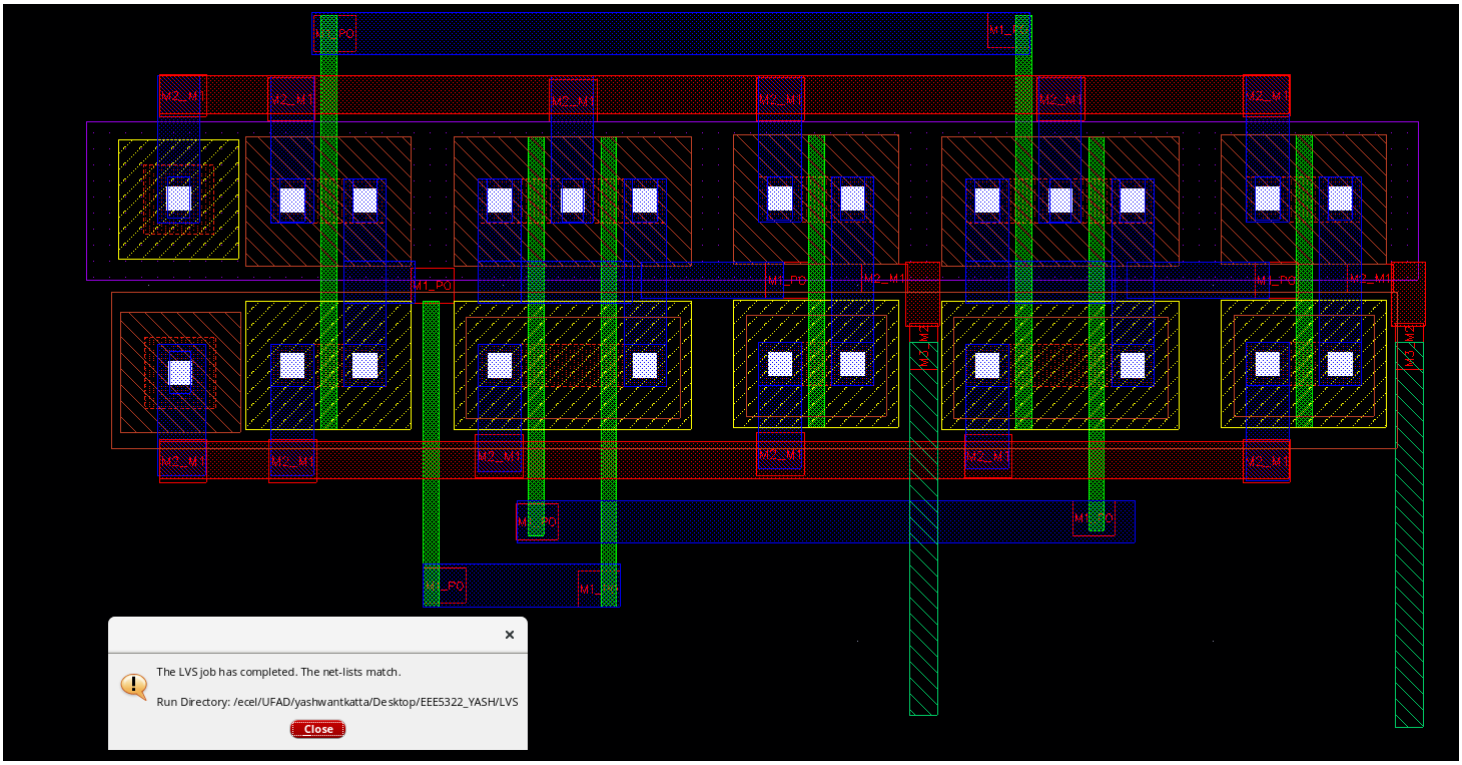


Fig : Column Decoder - LVS

3. Row Decoder :

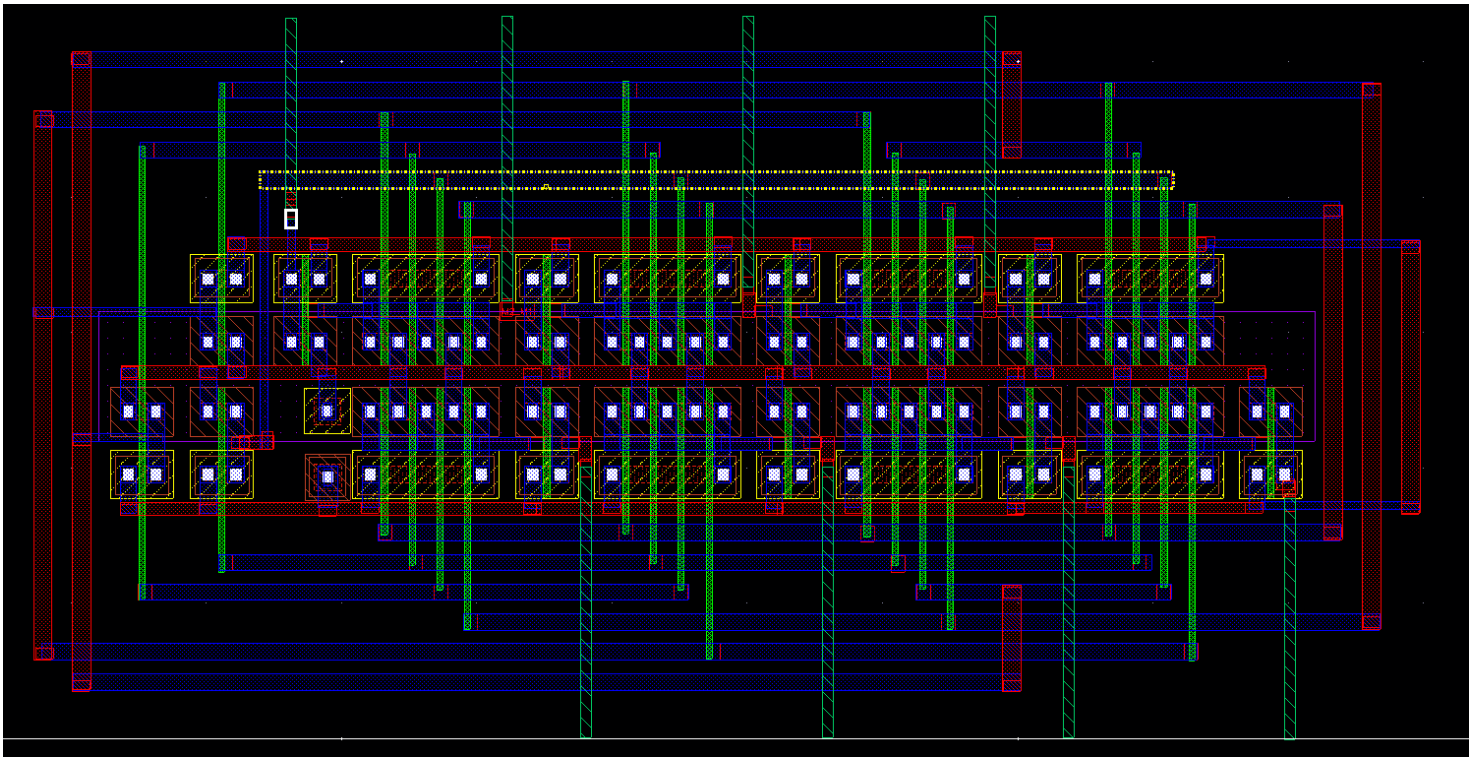


Fig : Row Decoder Layout

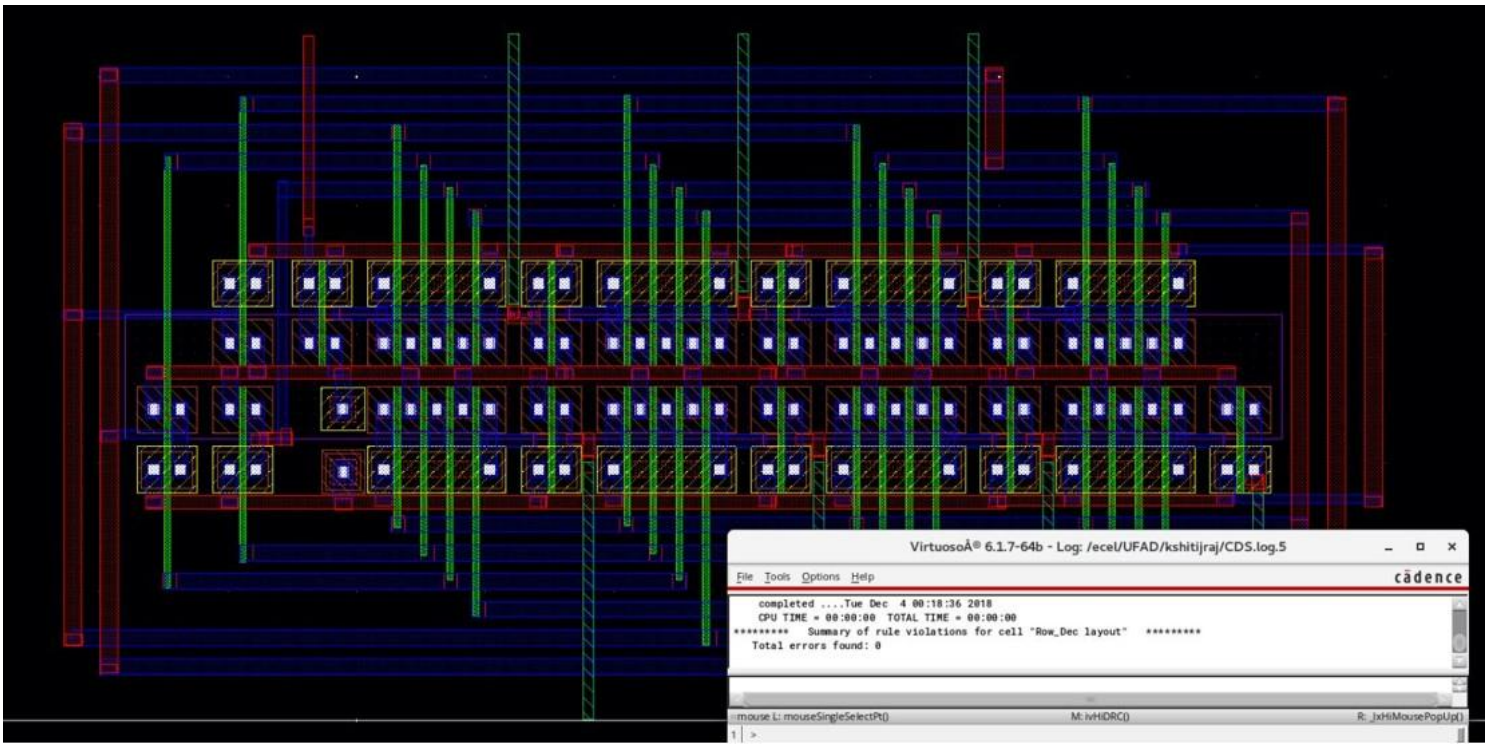


Fig : Row Decoder - DRC

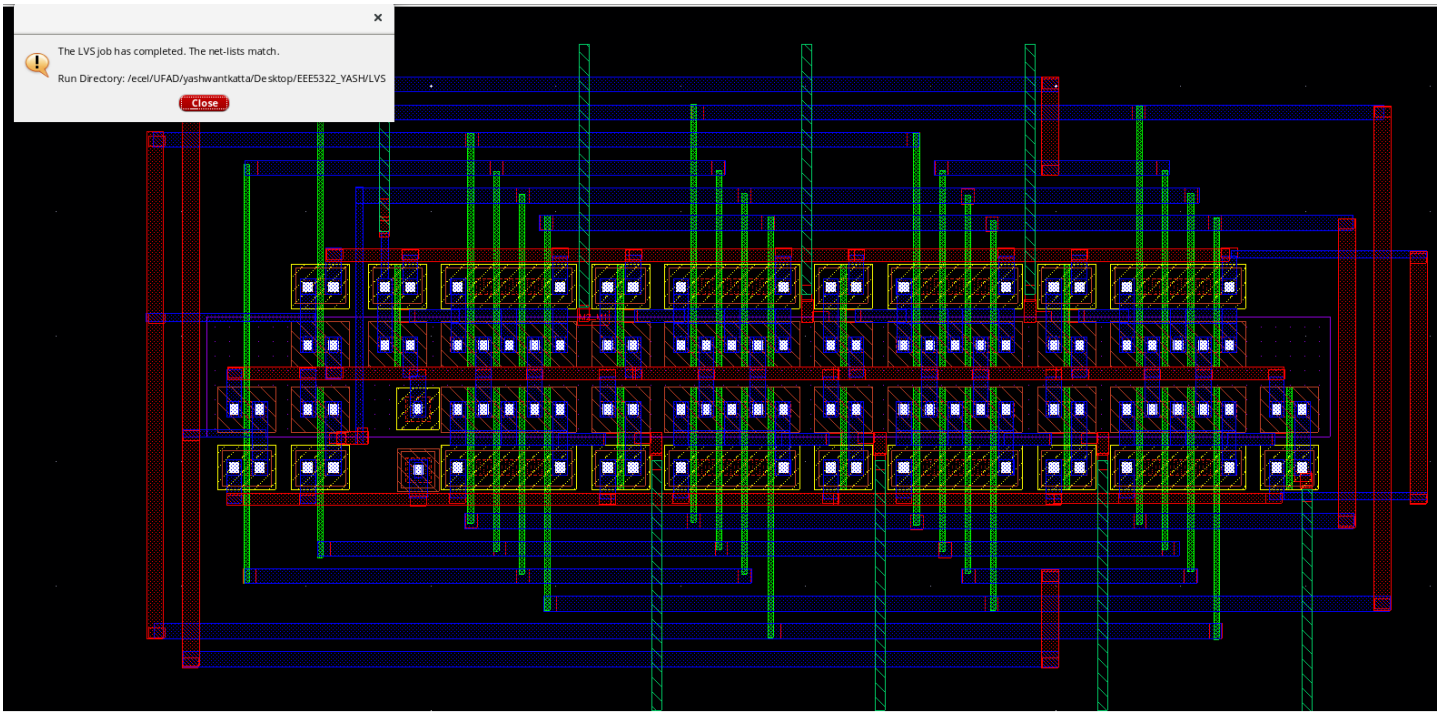


Fig : Row Decoder - LVS

4. Read and Write Circuit :

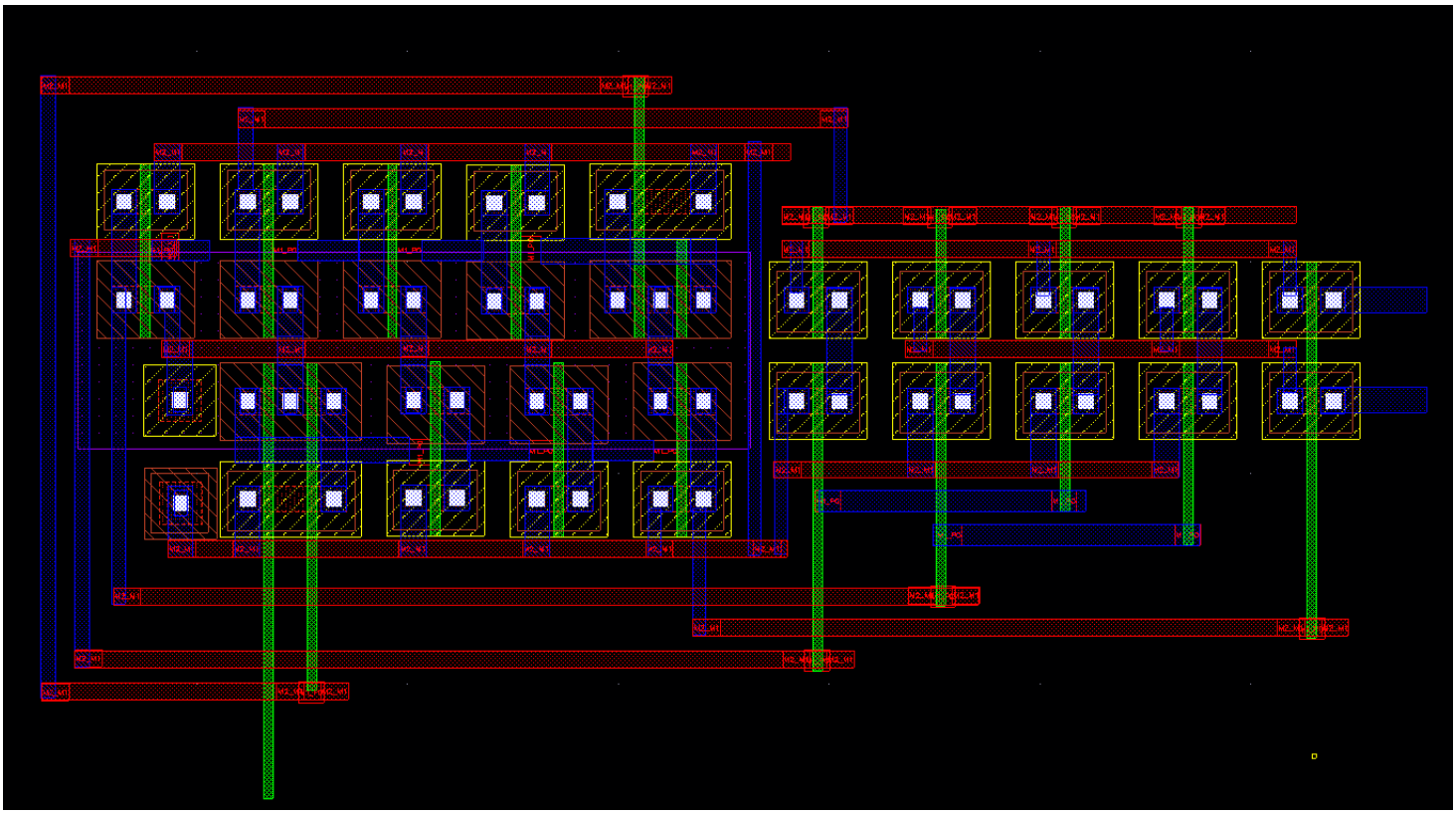


Fig : Read and Write Circuit Layout

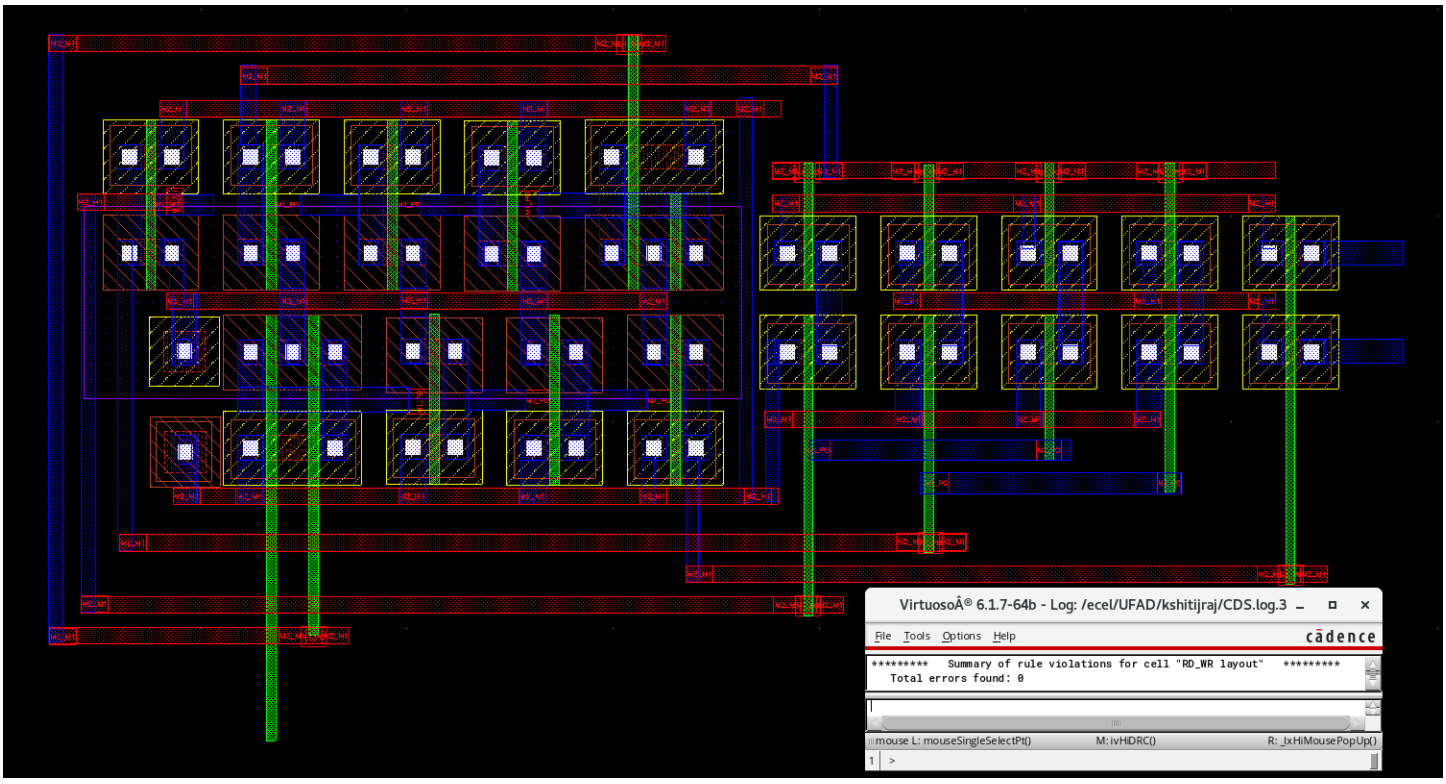


Fig : Read and Write Circuit - DRC

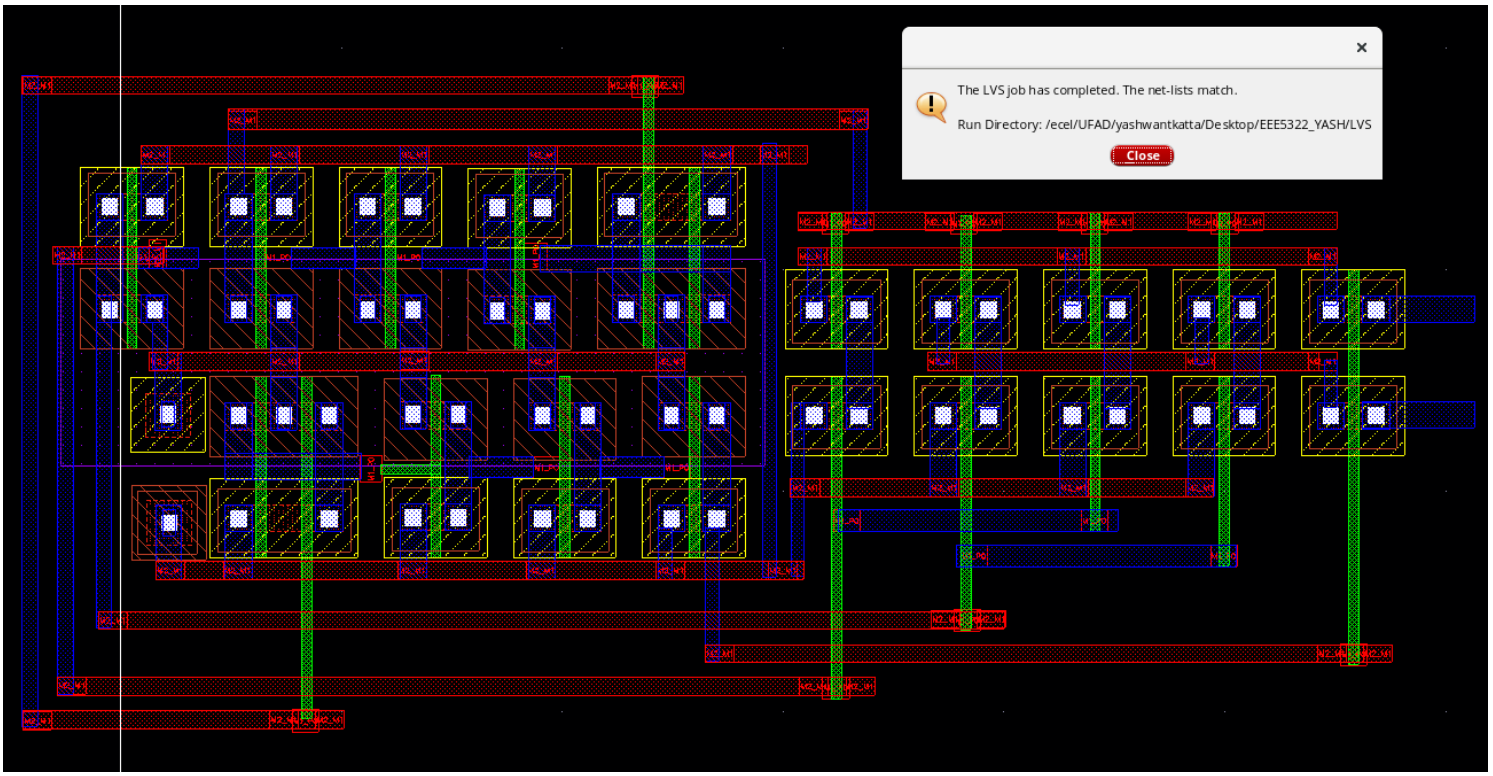


Fig : Read and Write Circuit - LVS

5. Sense Amplifier :

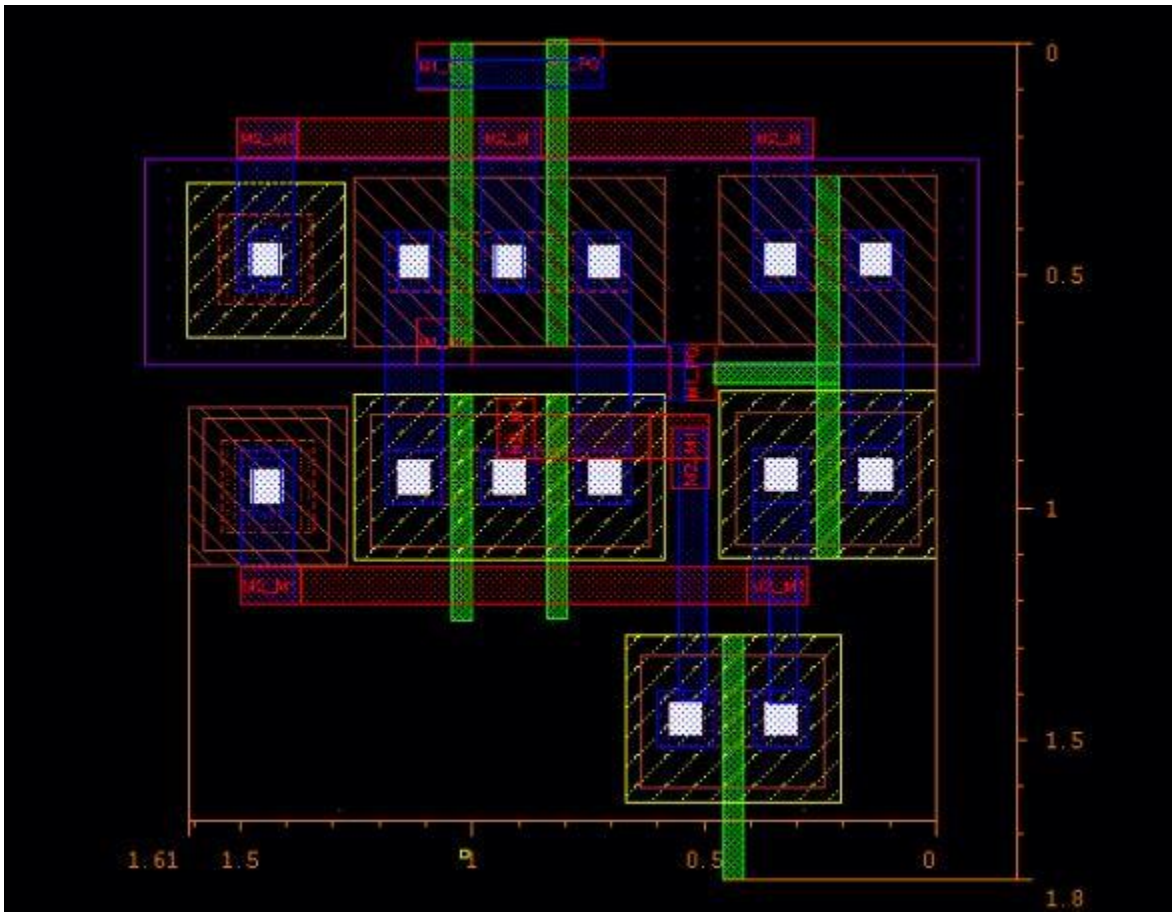


Fig: Sense Amplifier Layout

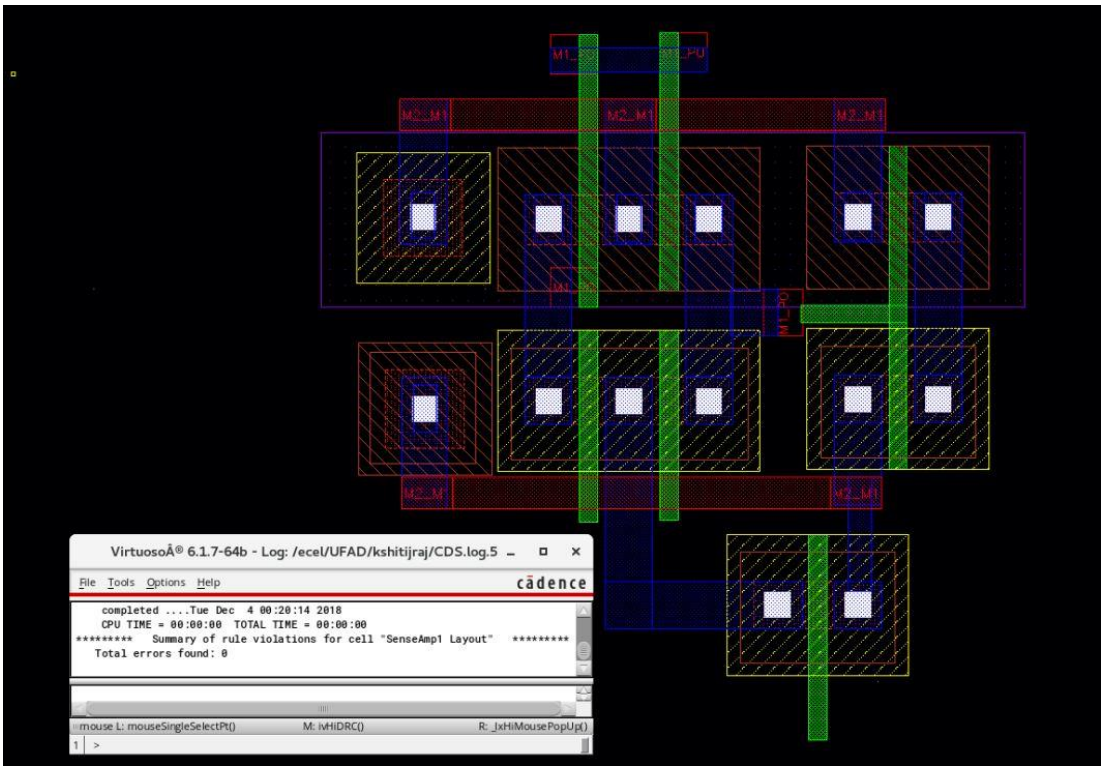


Fig: Sense Amplifier DRC

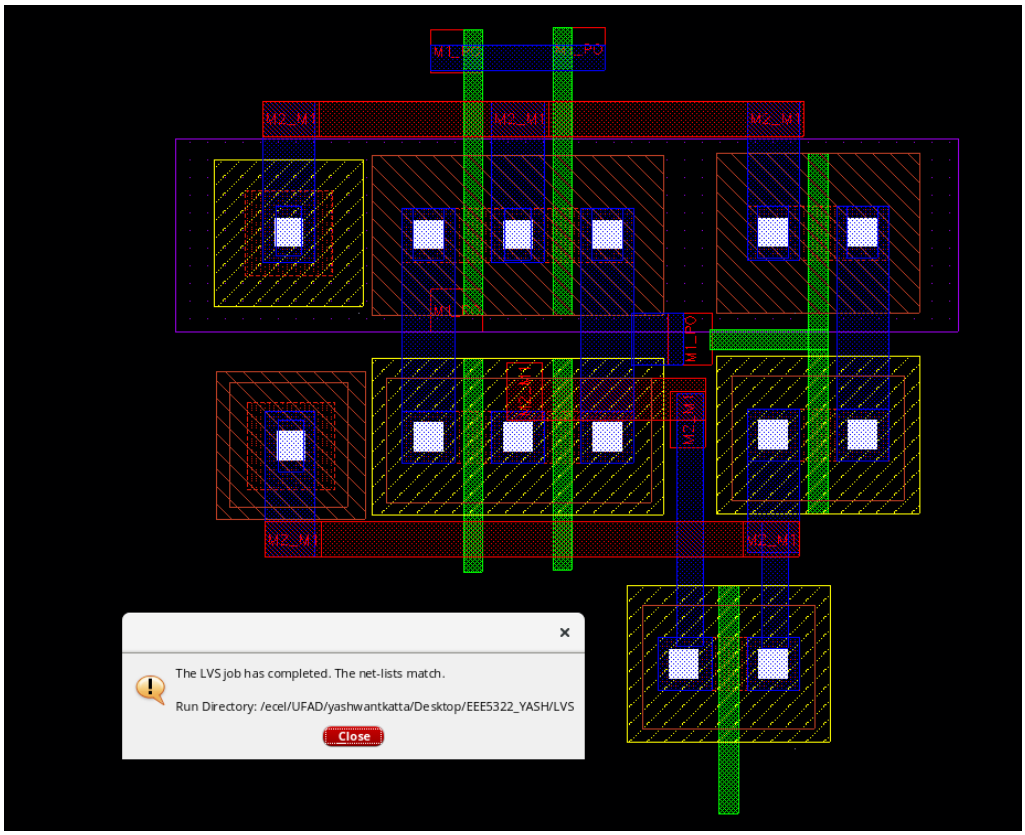


Fig: Sense Amplifier LVS

- **Area of the single SRAM Cell = 1.61 × 1.8 $(um)^2$**
  = <u>2.898 $(um)^2$</u>

6. Full SRAM Array :

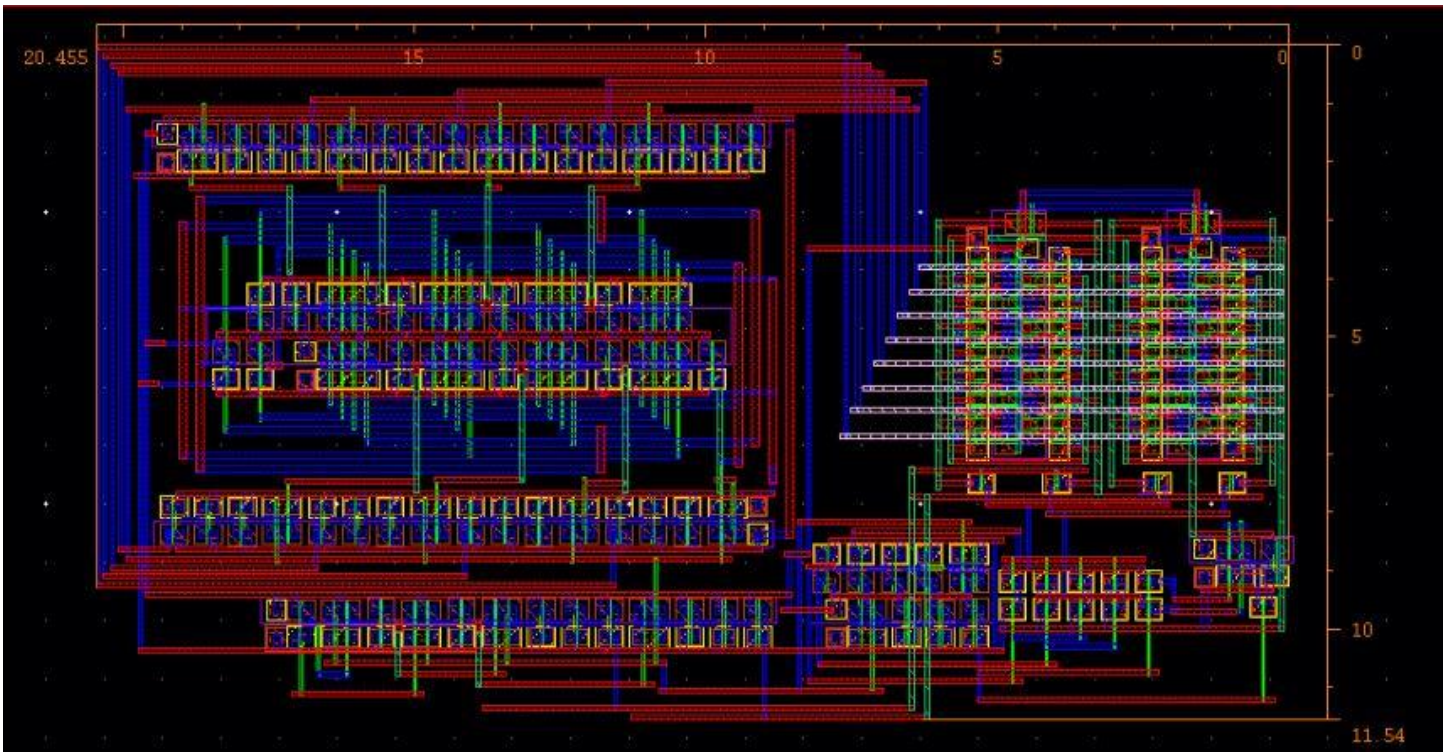


Fig: SRAM Array Layout

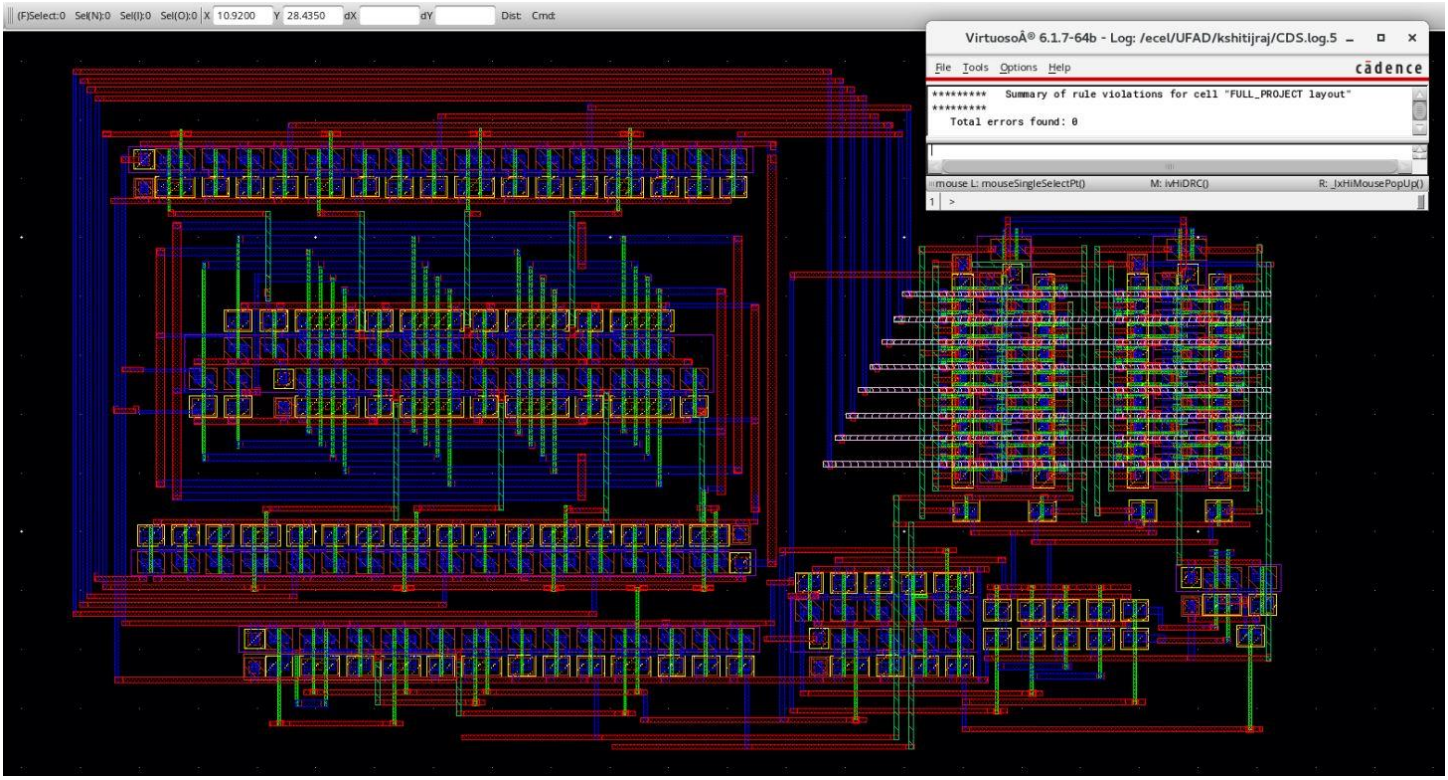


Fig: SRAM Array DRC

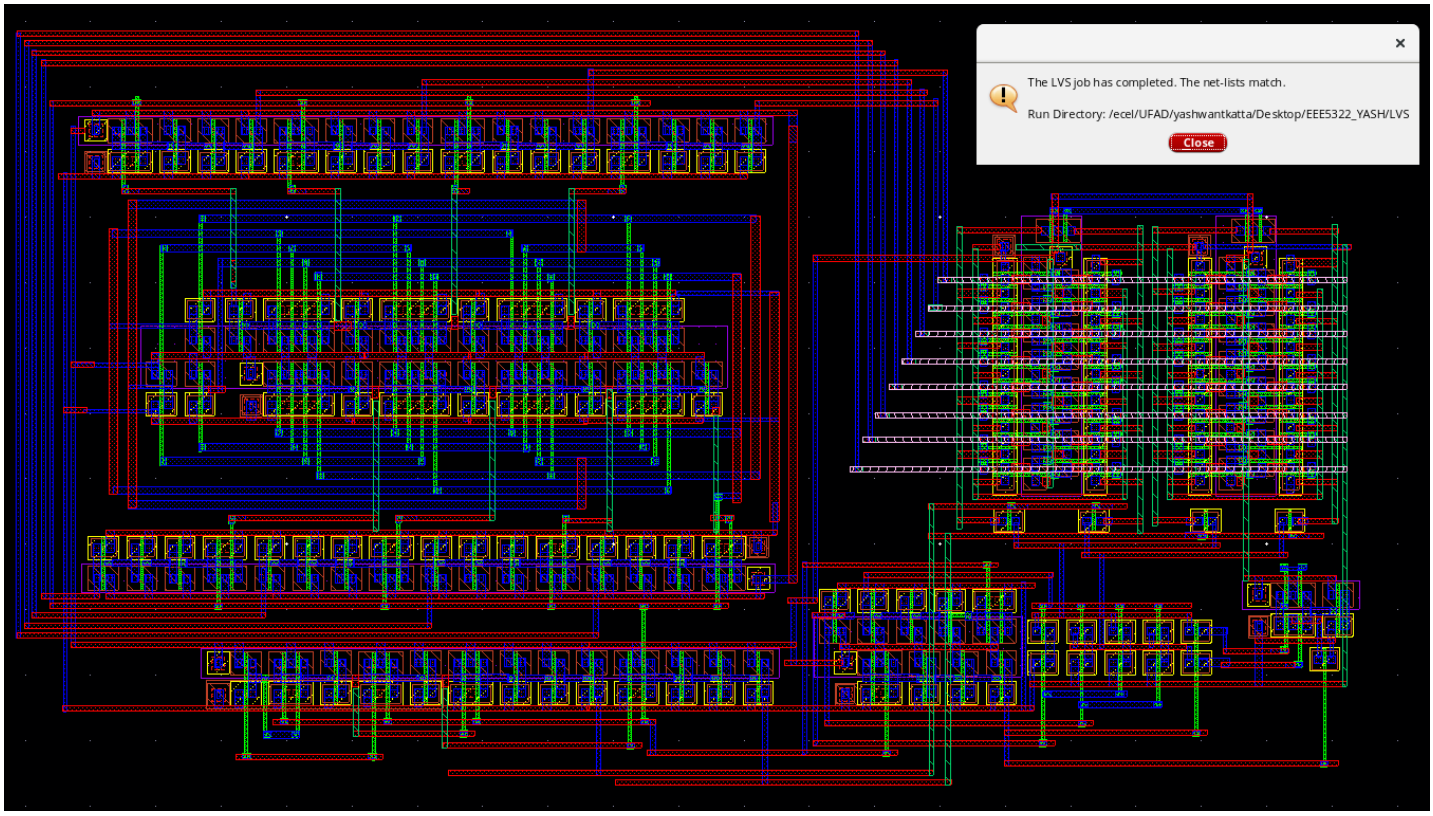


Fig: SRAM Array LVS

- **Area of the single SRAM Cell = 20.455 × 11.54 $(um)^2$**
  = <u>236.0507 $(um)^2$</u>

❖ Si.out file:

```
@(#)$CDS: LVS version 6.1.7-64b 11/21/2017 20:15 (sjfhw306) $
Command line: /ecel/apps/cds/ic617/tools.lnx86/dfII/bin/64bit/LVS -dir
/ecel/UFAD/yashwantkatta/Desktop/EEE5322_YASH/LVS -l -s -
x/ecel/UFAD/yashwantkatta/Desktop/EEE5322_YASH/LVS/xref.out -f -t
/ecel/UFAD/yashwantkatta/Desktop/EEE5322_YASH/LVS/layout
/ecel/UFAD/yashwantkatta/Desktop/EEE5322_YASH/LVS/schematic
Like matching is enabled.
Net swapping is enabled.
Creating cross reference file
/ecel/UFAD/yashwantkatta/Desktop/EEE5322_YASH/LVS/xref.out.
Fixed device checking is enabled.
Using terminal names as correspondence points.

 GENERIC PDK LVS Rules

    Net-list summary for
/ecel/UFAD/yashwantkatta/Desktop/EEE5322_YASH/LVS/layout/netlist
       count
        170             nets
        0               terminals
        150             g45p1svt
        193             g45n1svt

    Net-list summary for
/ecel/UFAD/yashwantkatta/Desktop/EEE5322_YASH/LVS/schematic/netlist
       count
        170             nets
        12              terminals
        150             g45p1svt
        193             g45n1svt

Devices in the netlist but not in the rules:
        g45p1svt g45n1svt
Devices in the rules but not in the netlist:
        g45pd1lvt g45nd1hvt g45nd1lvt g45nd1nvt g45nd2svt g45nd2nvt g45pd1hvt
        g45pd2svt g45rm1 g45rm2 g45rm3 g45rm4 g45rm5 g45rm6 g45rm7 g45rm8
        g45rm9 g45rm10 g45rm11 g45rnsnd g45rnsnp g45rnspd g45rnspp g45rnwoxide
        g45rnwsti g45rsnd g45rsnp g45rspd g45rspp g45ncap1 g45ncap2 g45pcap1
        g45pcap2 pnp npn g45vnpn2 g45vnpn5 g45vnpn10 g45vpnp2 g45vpnp5
        g45vpnp10 g45nd1svt g45pd1svt g45cmim g45inda g45inds

4 net-list ambiguities were resolved by random selection.

The net-lists match.

                            layout  schematic
                               instances
        un-matched              0       0
        rewired                 0       0
        size errors             0       0
        pruned                  0       0
        active                  343     343
        total                   343     343

                                 nets
        un-matched              0       0
        merged                  0       0
        pruned                  0       0
        active                  170     170
        total                   170     170
```

	terminals	
un-matched	0	0
matched but different type	0	0
total	0	12

Probe files from /ecel/UFAD/yashwantkatta/Desktop/EEE5322_YASH/LVS/schematic

devbad.out:

netbad.out:

mergenet.out:

termbad.out:

prunenet.out:

prunedev.out:

audit.out:

Probe files from /ecel/UFAD/yashwantkatta/Desktop/EEE5322_YASH/LVS/layout

devbad.out:

netbad.out:

mergenet.out:

termbad.out:

prunenet.out:

prunedev.out:

audit.out:

CONCLUSION

The 8X2 SRAM Array schematic has been created and verified. All the individual blocks have been designed separately and tested. Their layouts have been created and tested for the corresponding schematic. The DRC and LVS were passed successfully for each individual block. Once the individual parts have been tested, the entire circuit was put together and the layouts have been connected. The final layout passed it's DRC and LVS flawlessly. This ensured that the circuit was complete. Now, we checked for the timing analysis. All the inputs have been carefully selected in order to get a valid output. Eventually the output came as expected. Four SRAM cells were selected which were written to and read from. The output waveform reads a '1' when that corresponding SRAM has a '1' stored and same is the case for '0'. Following this we measured the precharge time, write and read times for '1' and '0' respectively.

Thus, the project ended in success. The timing analysis matched our expectations and the final layout passed all the tests successfully. This project is a great opportunity to clearly understand the working of the SRAM Array and apply the knowledge practically and we would like to thank the instructor for it. Doing this project did give us an advantage in expanding our vision in designing and testing a circuit which would prove to be quite useful in the future.

The results thus prove the functionality of our SRAM array. It establishes the fact that the storing of data is being done efficiently and the layout is error free.