512 Bit SRAM Schematic and Layout Simulation

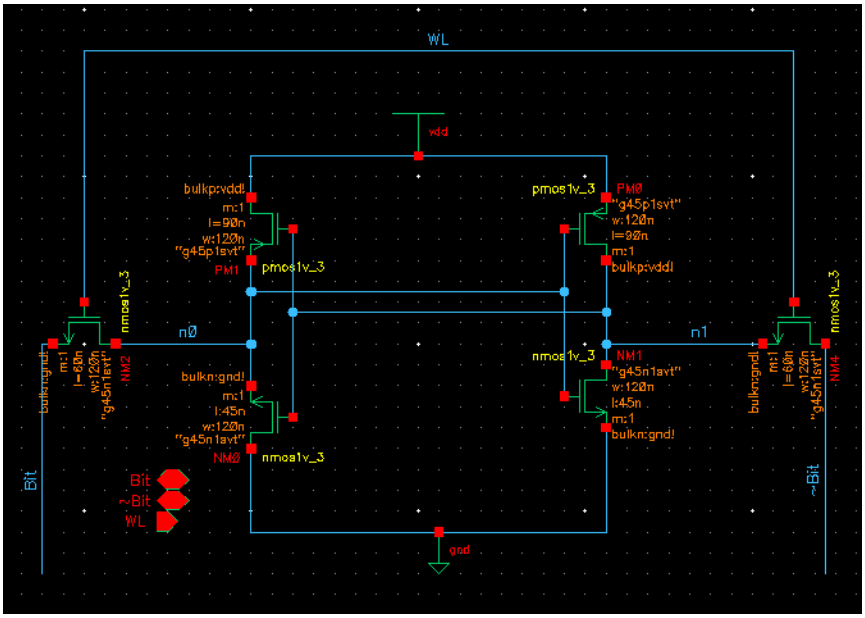
Submitted by –

Rohit Kumar Singh

Content:

SRAM Schematic Design–

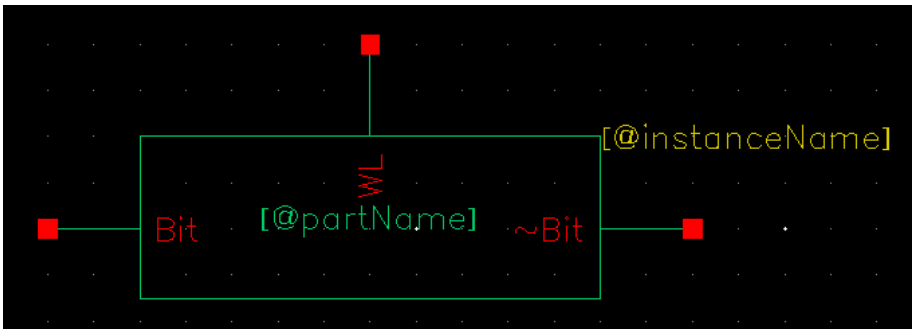
The One-cell SRAM is the fundamental building block of our SRAM Bank. Sizing: PUN: W = 120n, L = 90n, PDN: W = 120n, L = 45n, Access: W = 120n, L = 60n



Fiq 1 : 1 SRAM Cell Schematic

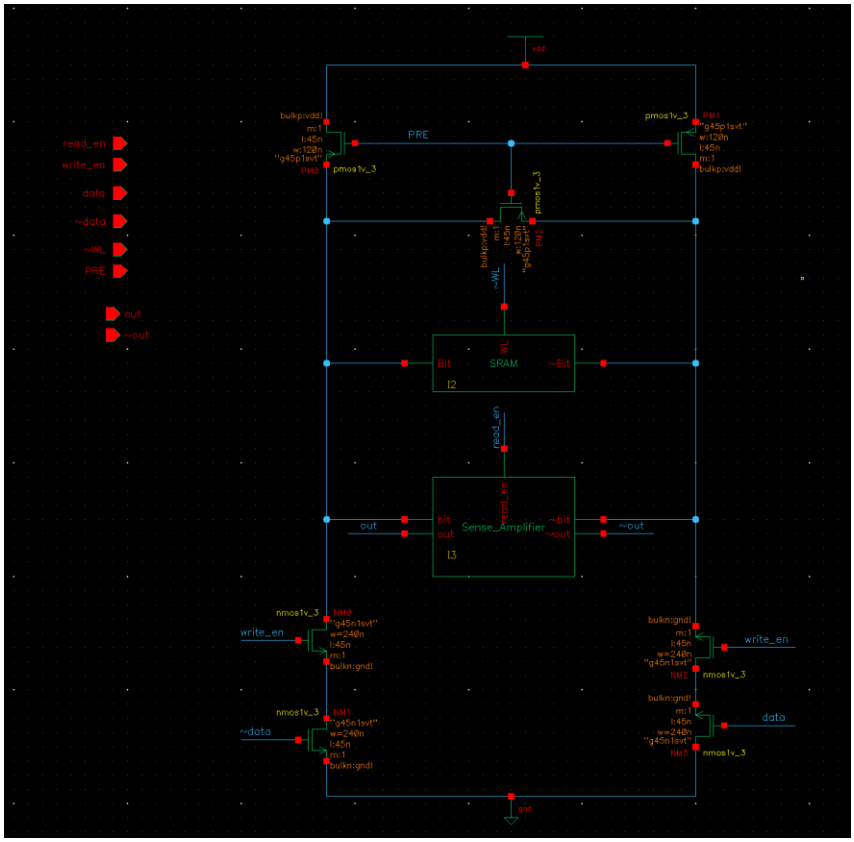
Our 6T SRAM cell contains a pair of cross-coupled CMOS inverters which hold the state of the memory, it also has access transistors which help the read and write states. The pull up and the pu;; down of the CMOS inverters are sized w.r.t the logic that the pull up transistor is required to be the strongest, access should be medium sized and the pull-down transistors will be the weakest. We write into our SRAM Cell driving a 1 or 0 into the desired voltages of the nodes n0 and n1 of the 6T SRAM Cell. The reading is done when the read-enable is turned to be high and the sense amplifier sense a changes in the output of the 512 SRAM Cell.

Below is the symbol of our basic SRAM cell, we will use it to instantiate in our top-level banks design.



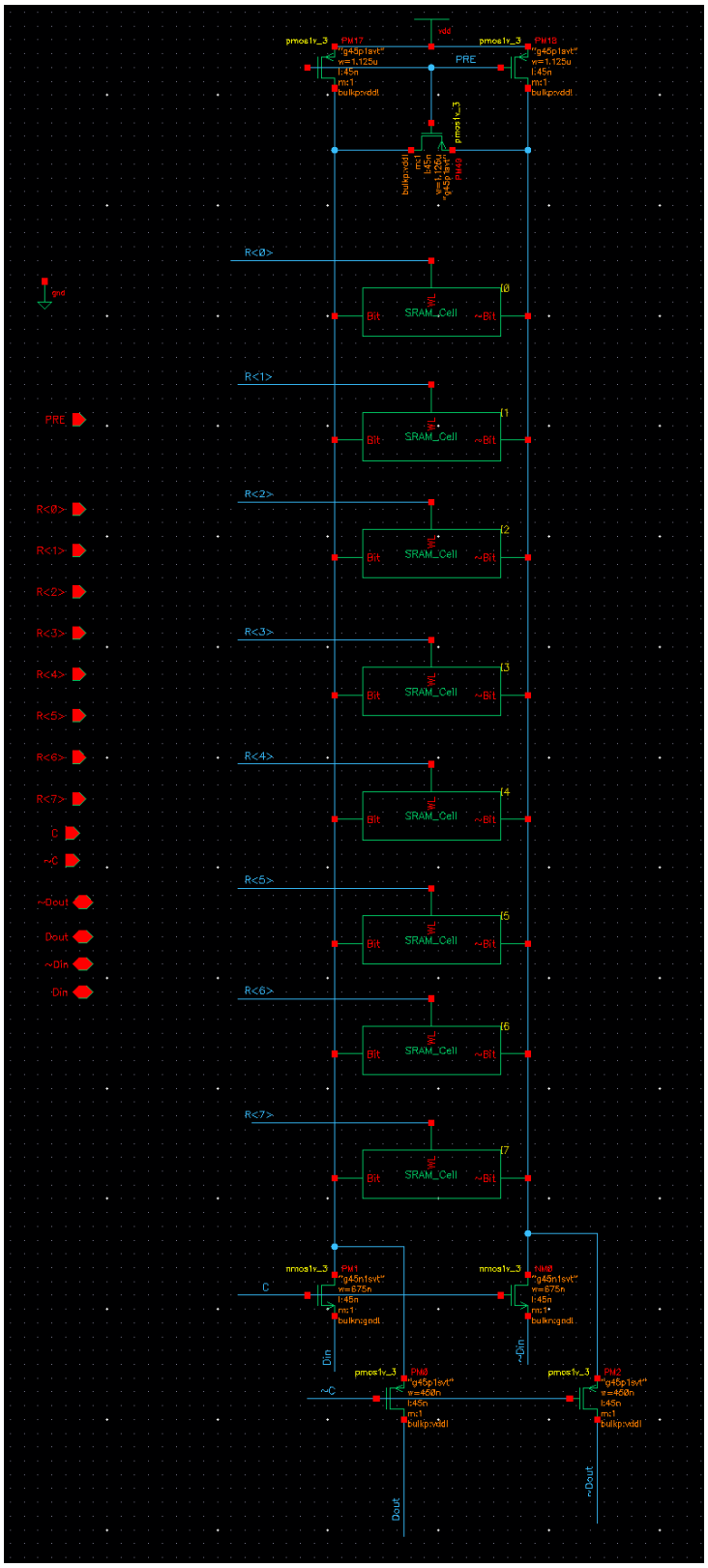
Fiq 2: 1 SRAM Cell Symbol

The 8 individual SRAM cells are placed in parallel in multiple rows to form one column. We will create multiple columns and instantiate those columns 16 times to form a bank. The vector file of this design helps in reading and writing of data, in the vector file, we always set the pre-charging circuit to one when we are writing and the reading. So, you will see that whenever write enable and read enable is on we have the PRE charge pin set to high, whenever it's not set, we have the PRE charge circuit set to low.

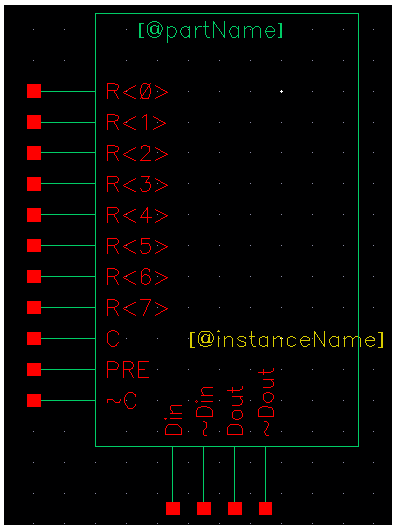


Fiq 3: 1 SRAM Data Path with PRE-Charge

1_SramCell.vec

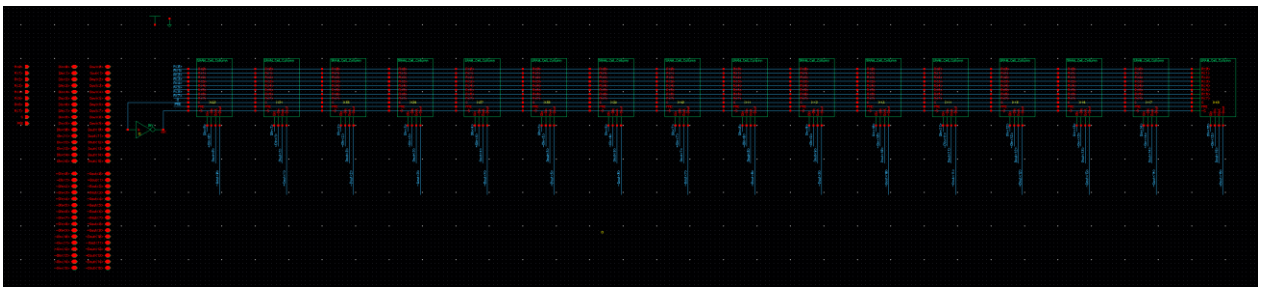


Fiq 3: 1 SRAM Column with Pre-Charge and READ/WRITE Muxes. Pre Charge Sizing: Wpmos= 1.125u, L= 45n. Read Mux Sizing: Wpmos= 450n, L=45n. Write Mux Sizing: Wnmos= 675n, L=45n.



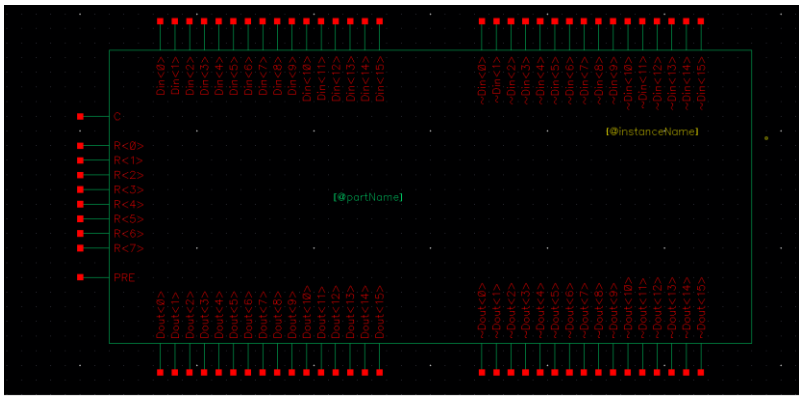
Fiq 4: 1 SRAM Column Symbol

We instantiate the above symbol 16 times to form an SRAM Bank.

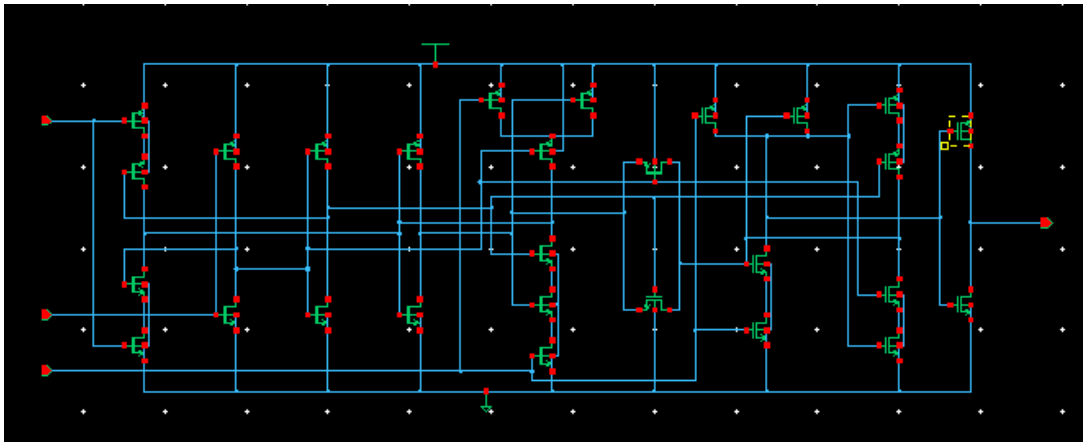


Fiq 5: 1 SRAM BANK (16 SRAM Columns)

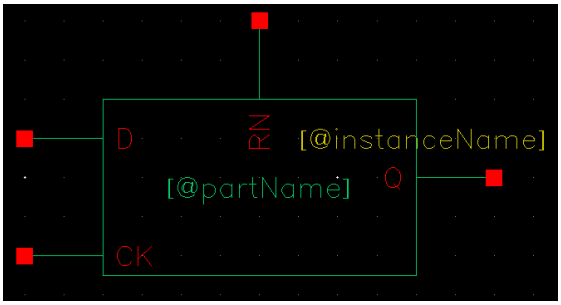
The instantiated symbols are combined and we create the symbol of the 1 BANK.



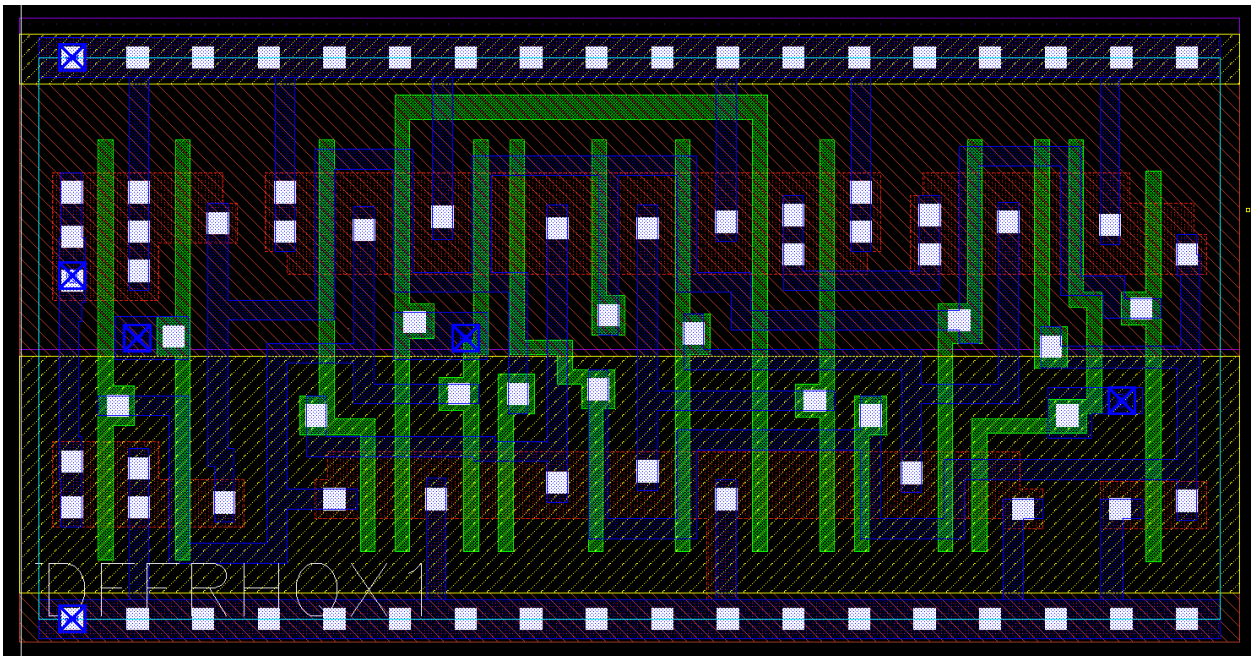
Fiq 6: 1 SRAM BANK



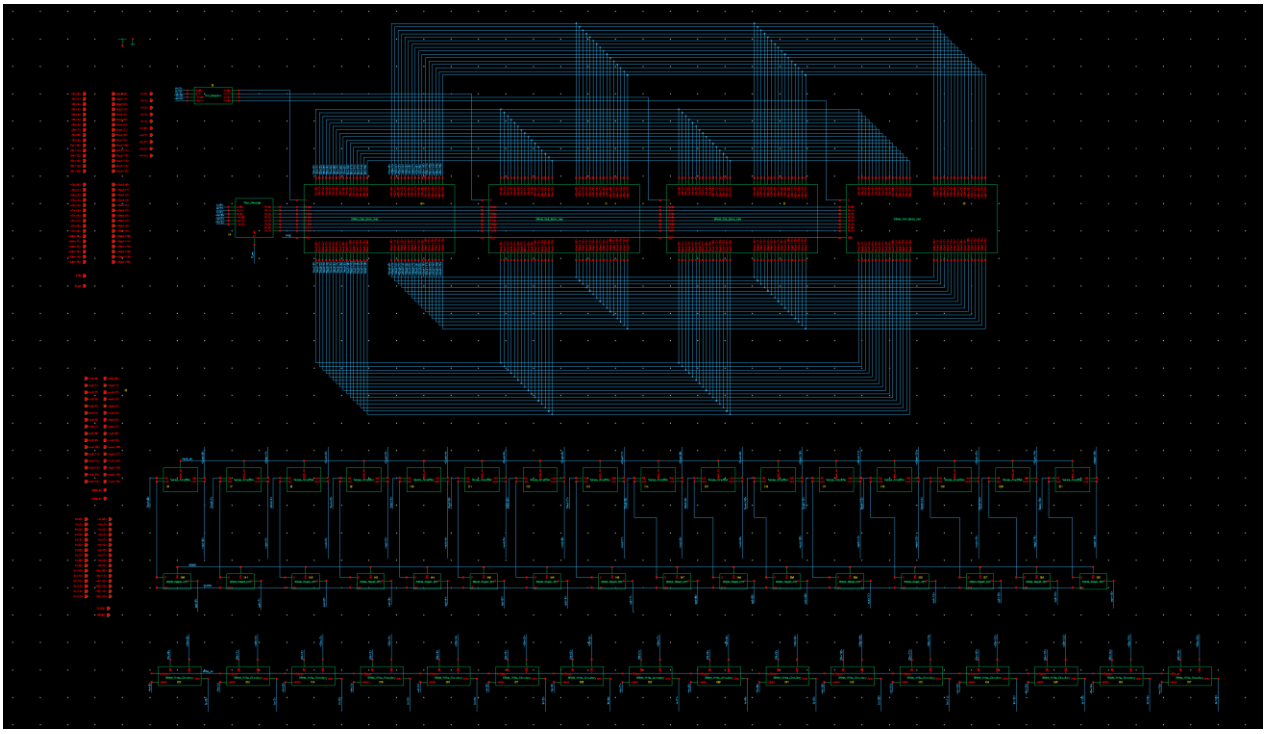
Fiq 7: Library D-Flip Flop: DFFHQRX1



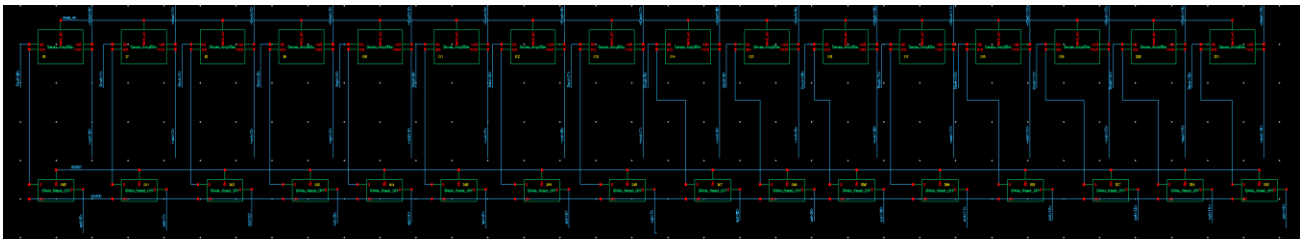
Fiq 8: Symbol D-Flip Flop



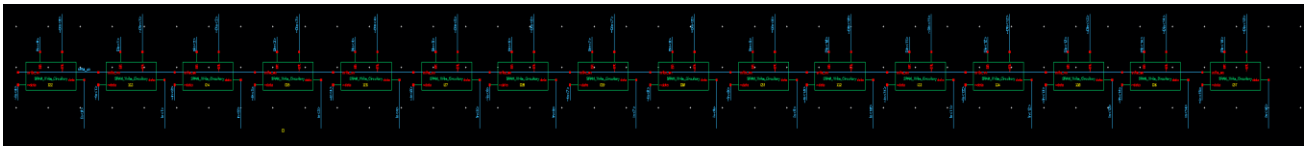
Fiq 9: D-Flip Flop Layout



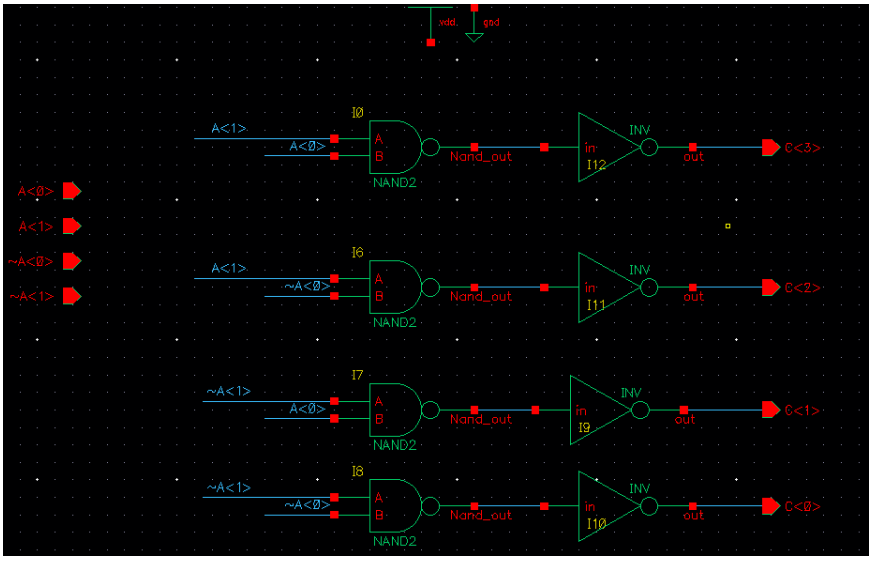
Fiq 9: 512 SRAM consisting of 4 Banks



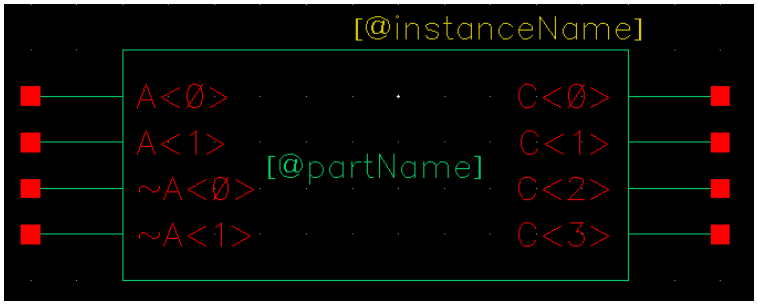
Fiq 10: Magnified Sense Amplifier



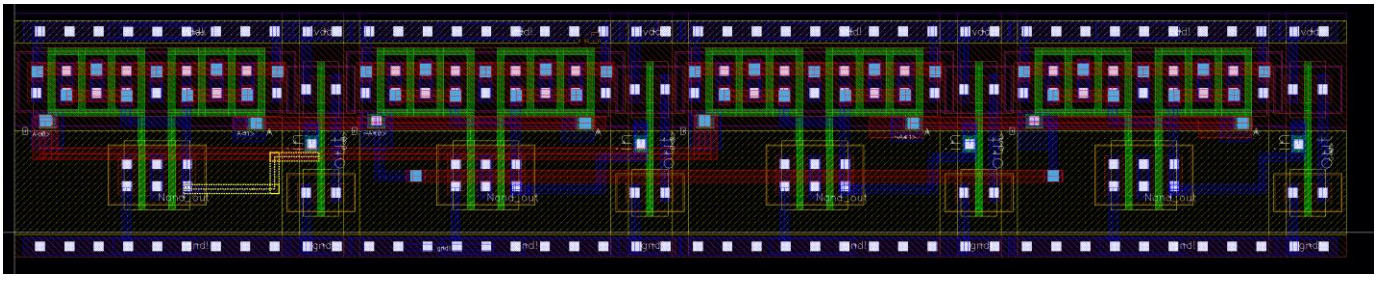
Fiq 11: Magnified Write Circuitry



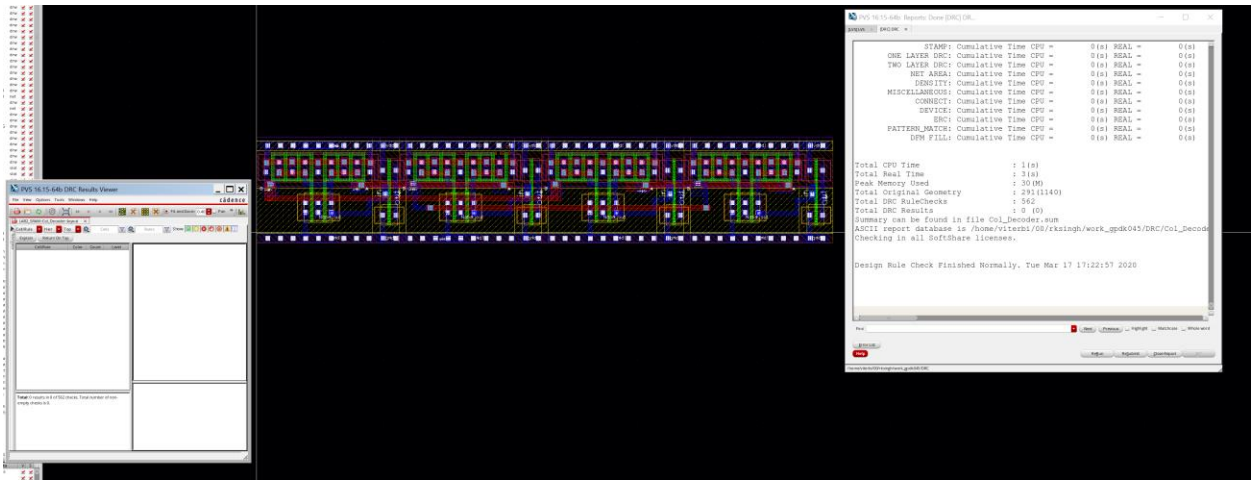
Fiq 12: Column Decoder: 4 AND Gates (INV PMOS W=185n/L=45n NMOS W=120/L=45n) (NAND PMOS W=1.08u/L=45n NMOS W=270n/L=45n)

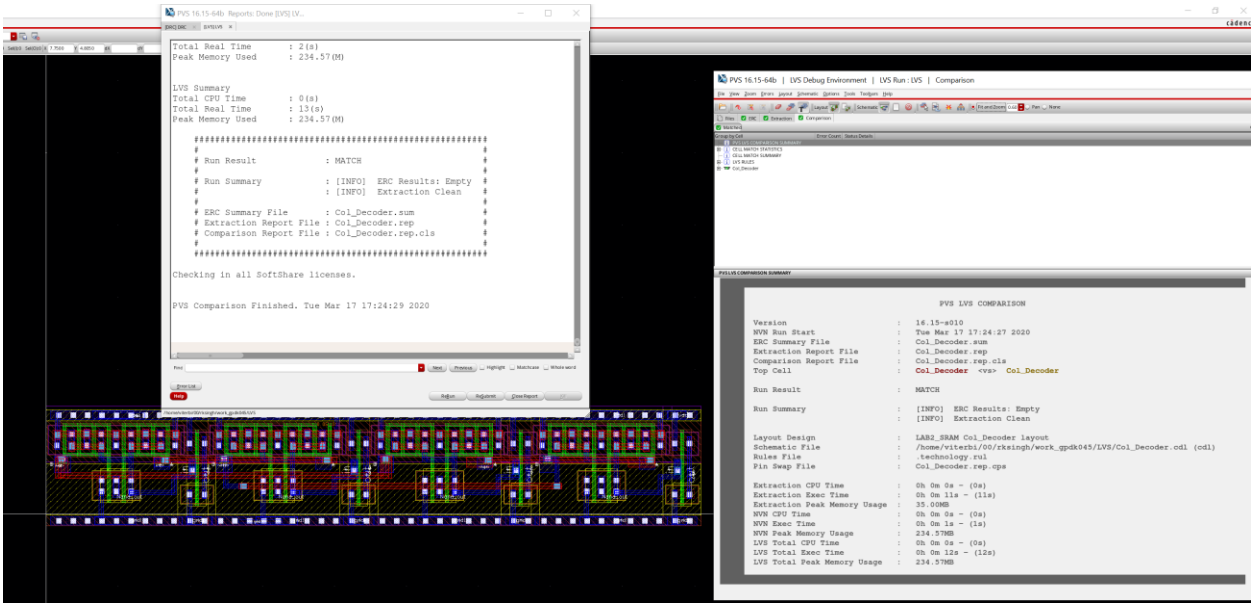


Fiq 13: Symbol Column Decoder



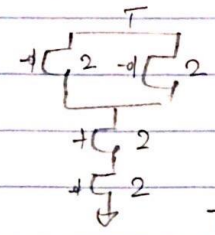
Column Decoder Layout





Logical Effort Calculations:

Sizing w.r.t NAND2. ($M_n/M_p=2$)



$\therefore C_{in} = 4C_g$

$C_p = 6C_d$.

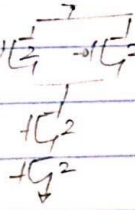
To make our calculations easy Let's make. ($C_d = C_g$)

$\therefore$ Since the decoder uses Predecoding hence (Branching factor = 2) for 1st stage. Also since we have two stages so ($N=2$) NAND2 and NOR3.

Let's find Logical Effort & parasitic Effort of NAND2 and NOR3.

$C_{in} = 4C_g$

$C_p = 6C_d$



(Sized w.r.t $M_n/M_p = 2/1$)

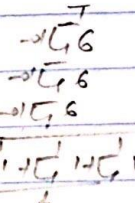
$\therefore$ Logical Effort $= \frac{4C_g}{4C_g} = 1$

parasitic Effort $= \frac{6C_d}{4C_g} = 3/2$.

Branching = 2

$C_{in} = 7C_g$

$C_p = 9C_d$



(Sized w.r.t $M_n/M_p = 2/1$)

$\therefore$ Logical Effort $= \frac{7C_g}{4C_g} = 7/4$

parasitic Effort $= \frac{9C_d}{4C_g} = 9/4$

Branching = 2 As the input is going to feature gain.

Now : $G = Ng = \frac{4}{4} \times \frac{7}{4} = 7/4$

$B = Nb = 2 \times 2 = 4$

$H = 128Cg / 4Cg$

Column Decoder picks one bank at a time. So
1 Bank $\approx 128\,Cg$. (input of NAND2 = $4Cg$)

$$Delay = N(GBH)^{1/N} + P.$$

$$F = GBH = \frac{7}{4} \times 4 \times \frac{128Cg}{4Cg}$$

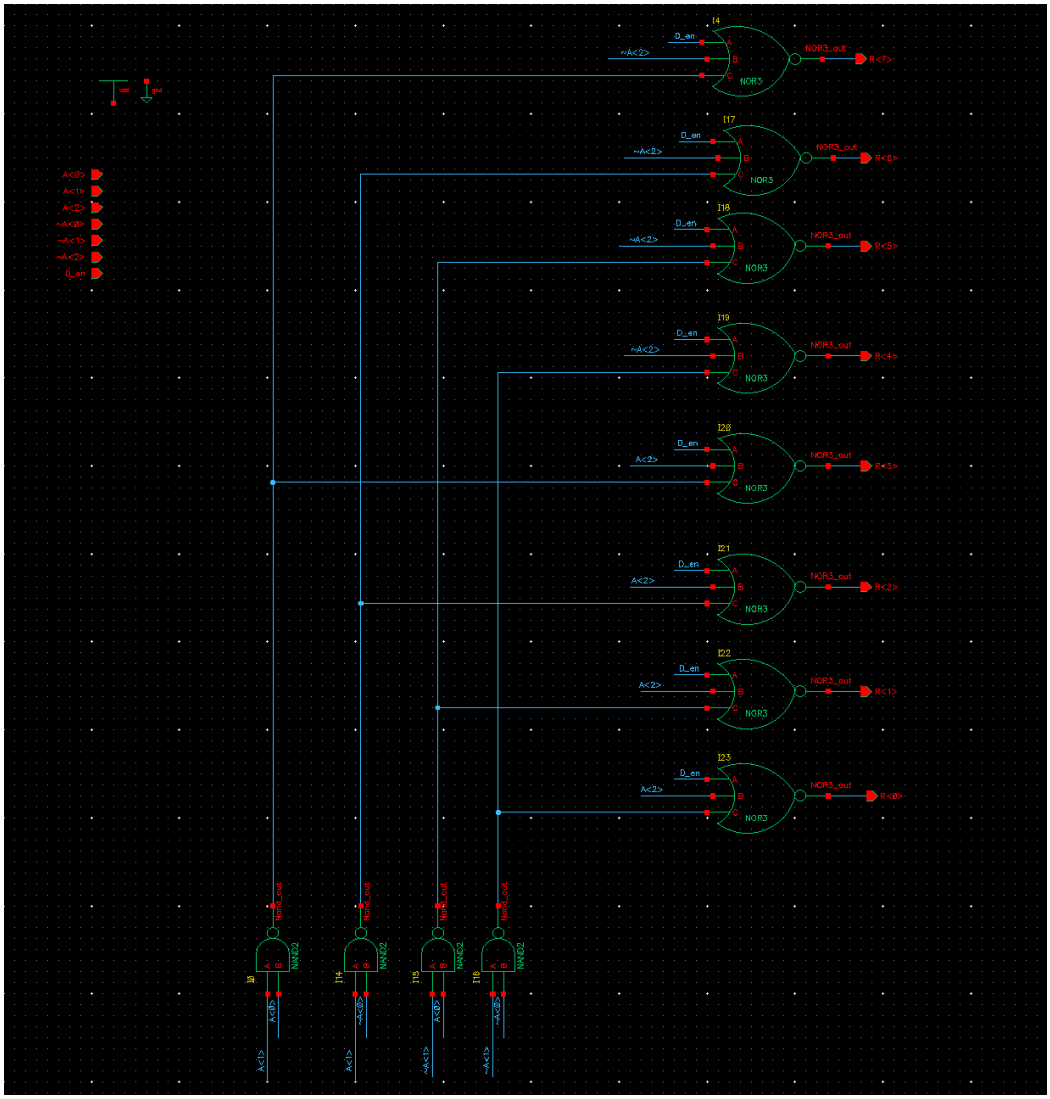
Now $F = 224$

$$f = \sqrt[2]{F}$$
$$= 2\sqrt{224}$$
$$= 14.95\ ns.$$

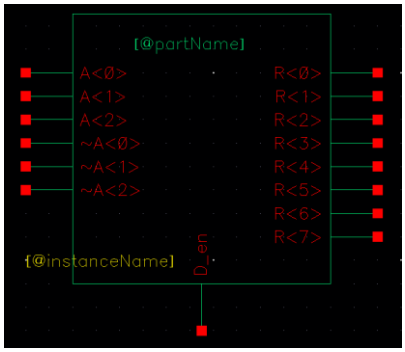
Now path Delay $\Rightarrow$ $P = P_1 + P_2 = \frac{3}{2} + \frac{9}{4} = \frac{15}{4} = 3.75$

$$Delay = 2(N \times f + P)$$
$$= 2(2 \times 14.96 + 3.75)$$
$$= 33.67\ ns.$$

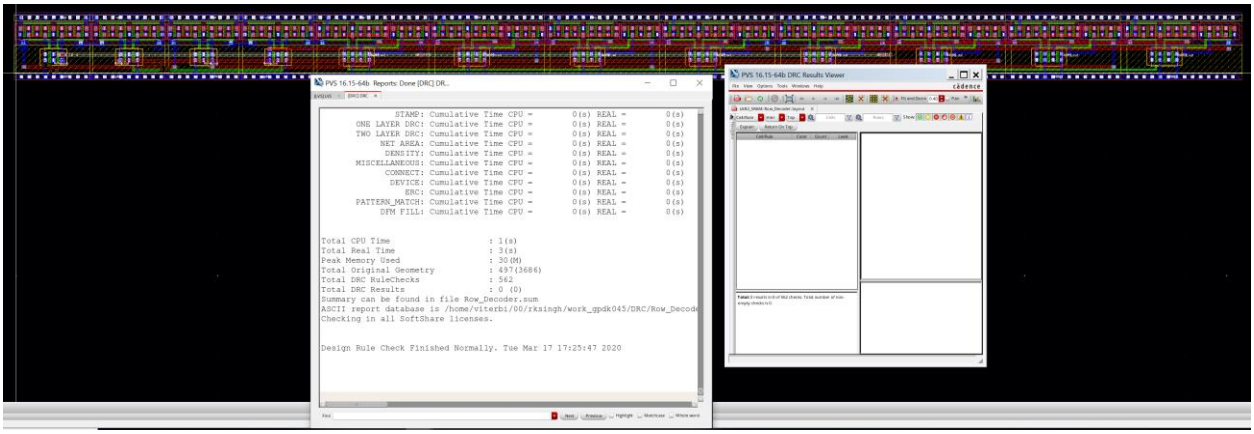
The enable signal is installed in the row decoder.

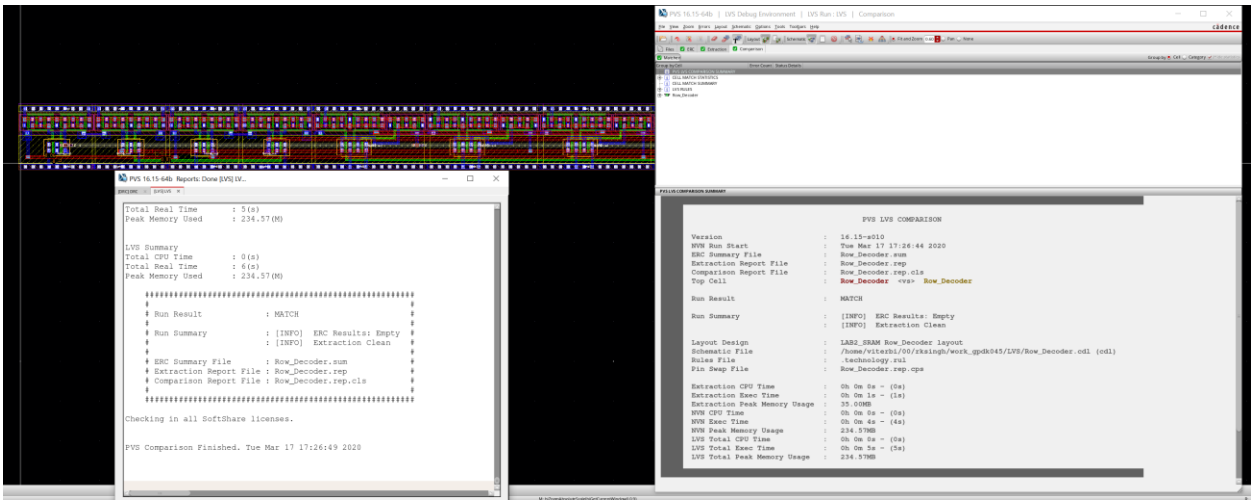


Fiq 14: 3X8 Row Decoder (NOR PMOS W=1.08u/L=45n NMOS W=270/L=45n) (NAND PMOS W=1.08u/L=45n NMOS W=270n/L=45n)

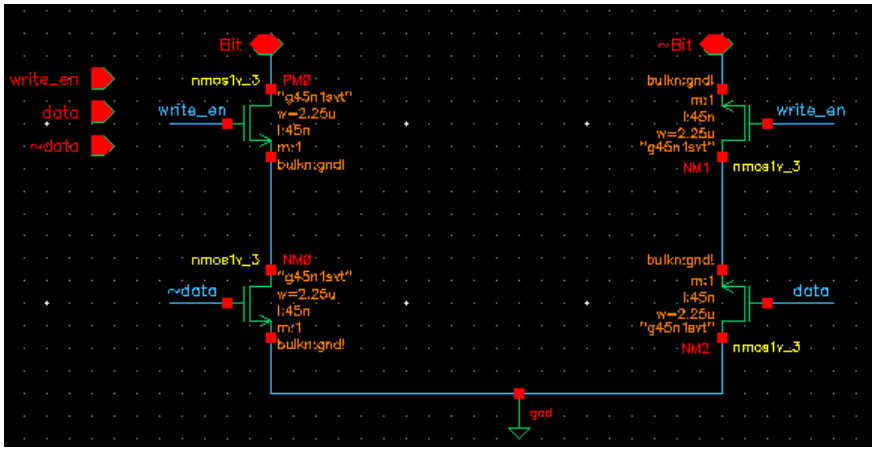


Fiq 15: Symbol of 3X8 Row Decoder

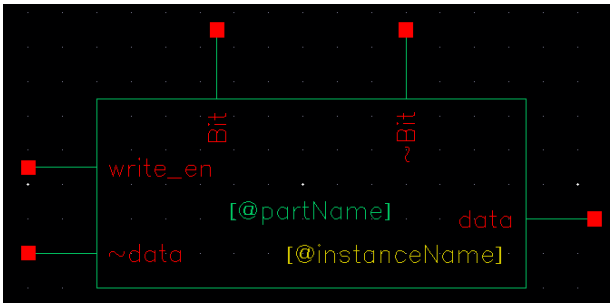

PVS 16.15-64b Reports: Done [DRC] DR...
STAMP: Cumulative Time CPU = 0(s) REAL = 0(s)
ONE LAYER DRC: Cumulative Time CPU = 0(s) REAL = 0(s)
TWO LAYER DRC: Cumulative Time CPU = 0(s) REAL = 0(s)
NET AREA: Cumulative Time CPU = 0(s) REAL = 0(s)
DENSITY: Cumulative Time CPU = 0(s) REAL = 0(s)
MISCELLANEOUS: Cumulative Time CPU = 0(s) REAL = 0(s)
CONNECT: Cumulative Time CPU = 0(s) REAL = 0(s)
DEVICE: Cumulative Time CPU = 0(s) REAL = 0(s)
ERC: Cumulative Time CPU = 0(s) REAL = 0(s)
PATTERN_MATCH: Cumulative Time CPU = 0(s) REAL = 0(s)
DFM FILL: Cumulative Time CPU = 0(s) REAL = 0(s)
Total CPU Time : 1(s)
Total Real Time : 3(s)
Peak Memory Used : 30(M)
Total Original Geometry : 497(3686)
Total DRC RuleChecks : 562
Total DRC Results : 0 (0)
Summary can be found in file Row_Decoder.sum
Checking in all SoftShare licenses.
Design Rule Check Finished Normally. Tue Mar 17 17:25:47 2020
PVS 16.15-64b DRC Results Viewer
cadence


PVS 16.15-64b Reports: Done [LVS] LV...
Total Real Time : 5(s)
Peak Memory Used : 234.57(M)
LVS Summary
Total CPU Time : 0(s)
Total Real Time : 6(s)
Peak Memory Used : 234.57(M)
Run Result : MATCH
Run Summary : [INFO] ERC Results: Empty
: [INFO] Extraction Clean
ERC Summary File : Row_Decoder.sum
Extraction Report File : Row_Decoder.rep
Comparison Report File : Row_Decoder.rep.cls
Checking in all SoftShare licenses.
PVS Comparison Finished. Tue Mar 17 17:26:49 2020
PVS LVS COMPARISON
Version : 16.15-s010
NVN Run Start : Tue Mar 17 17:26:44 2020
ERC Summary File : Row_Decoder.sum
Extraction Report File : Row_Decoder.rep
Comparison Report File : Row_Decoder.rep.cls
Top Cell : Row_Decoder <vs> Row_Decoder
Run Result : MATCH
Run Summary : [INFO] ERC Results: Empty
[INFO] Extraction Clean
Layout Design : LAB2_SRAM Row_Decoder layout
Schematic File : /home/viterbi/00/rksingh/work_gpdk045/LVS/Row_Decoder.cdl (cdl)
Rules File : .technology.rul
Pin Swap File : Row_Decoder.rep.cps
Extraction CPU Time : 0h 0m 0s - (0s)
Extraction Exec Time : 0h 0m 1s - (1s)
Extraction Peak Memory Usage : 35.00MB
NVN CPU Time : 0h 0m 0s - (0s)
NVN Exec Time : 0h 0m 4s - (4s)
NVN Peak Memory Usage : 234.57MB
LVS Total CPU Time : 0h 0m 0s - (0s)
LVS Total Exec Time : 0h 0m 5s - (5s)
LVS Total Peak Memory Usage : 234.57MB

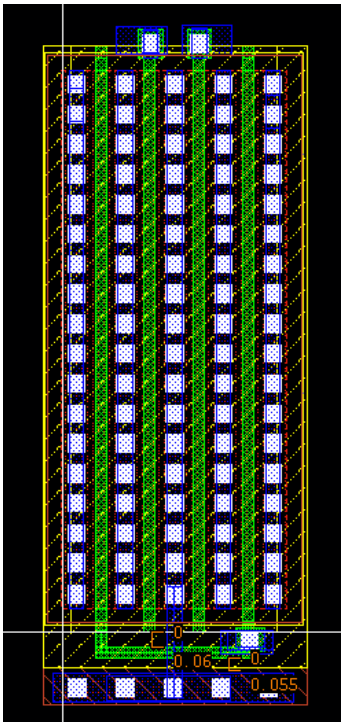
Write Circuitry performs the writing of the data in the SRAM, during the writing process one of the access transistors are on and we transfer the values from the bit and bit bar lines to the voltage nodes of our 6T SRAM cell. In our vector file it's clear that we have provided PRE charge signal when the write enable signal is set. We have taken the bit and Bit bar lines as input and output as those act both as input and output, regarding the action they are executing. For example, for reading purpose the Bit and Bit bar pins act as output but for the writing purpose these bit and bit_bar lines act as inputs – driving values to the SRAM.



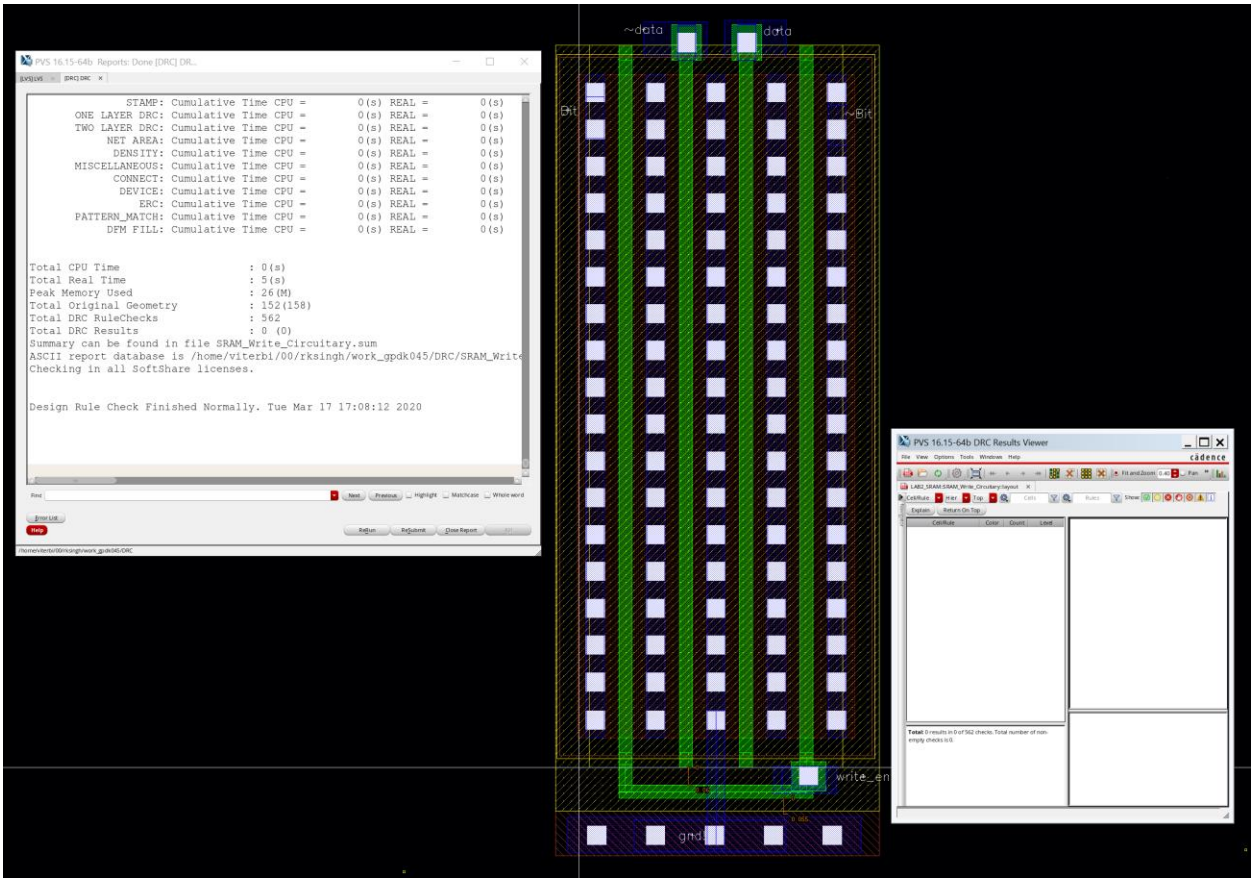
Fiq 16: Write Circuitry Schematic Sizing: NMOs, W=2.25u, L=45n



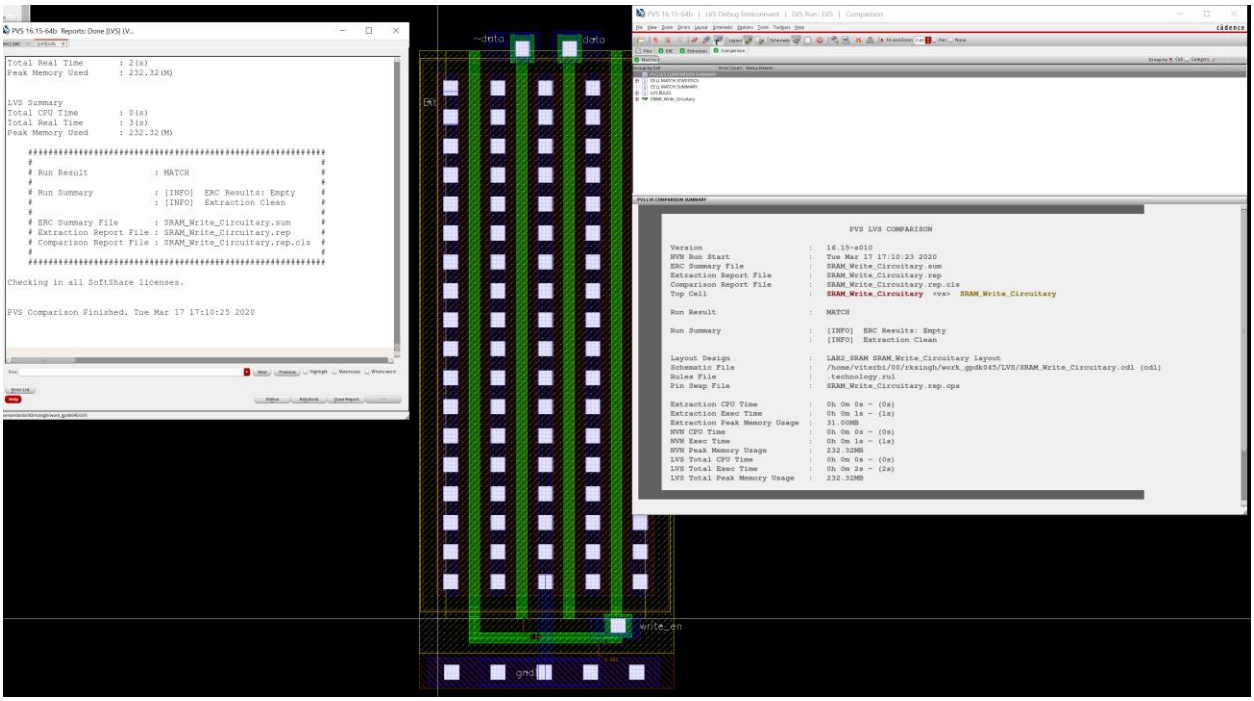
Fiq 17: Symbol Write Circuitary



Fiq 18: Write Circuitry Layout

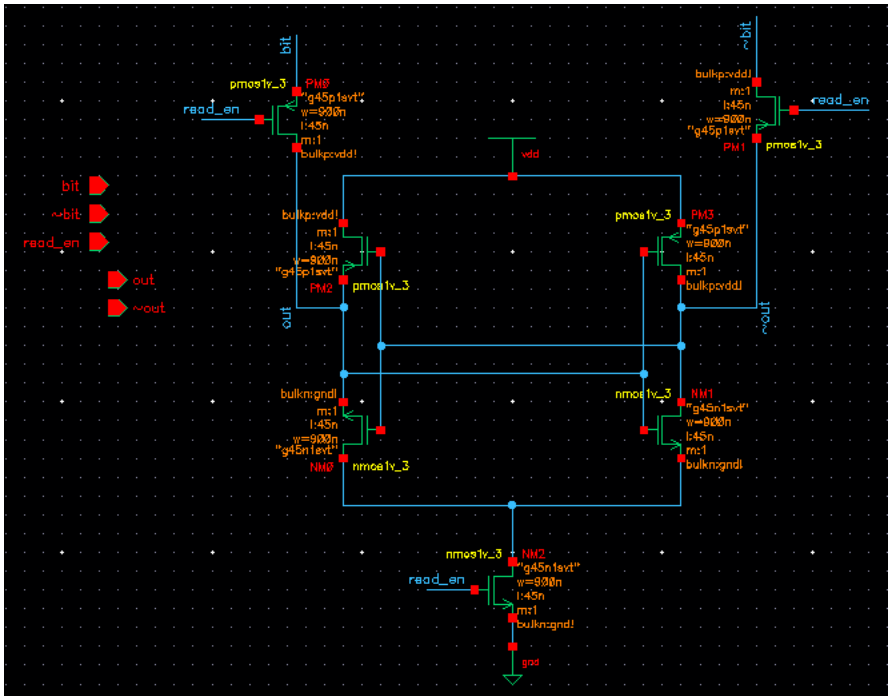


DRC Check – Write Circuitry

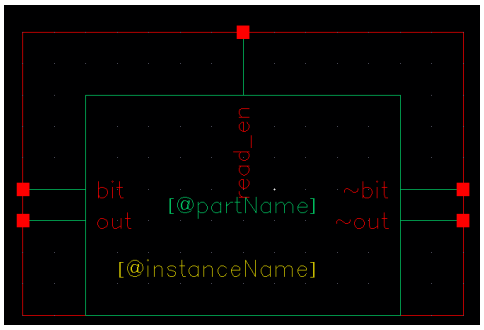


LVS Match – Write Circuitry

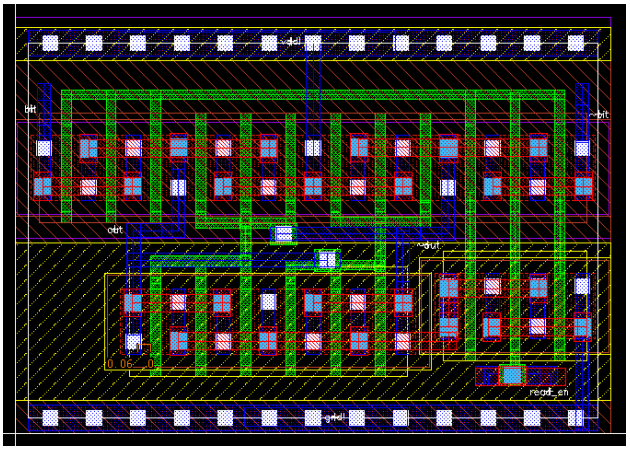
We have constructed an un-clocked Sense Amplifier, it get's activated even when there's light delta V change in the output of the 512 SRAM Cell. The Sense Amplifier helps reduce the power consumption of our SRAM, as we save immense amount of power by generating a voltage output by sensing a slight change in the input voltage. As we are aware that for reading any voltages from the 6T SRAM cell, we need to have written a value in the nodes of the 6T SRAM Cell, therefore we write a value through the write circuitry before reading the output from the Sense Amplifier. In our vector file we have selected the PRE-charging circuit to be on in order to read the value.



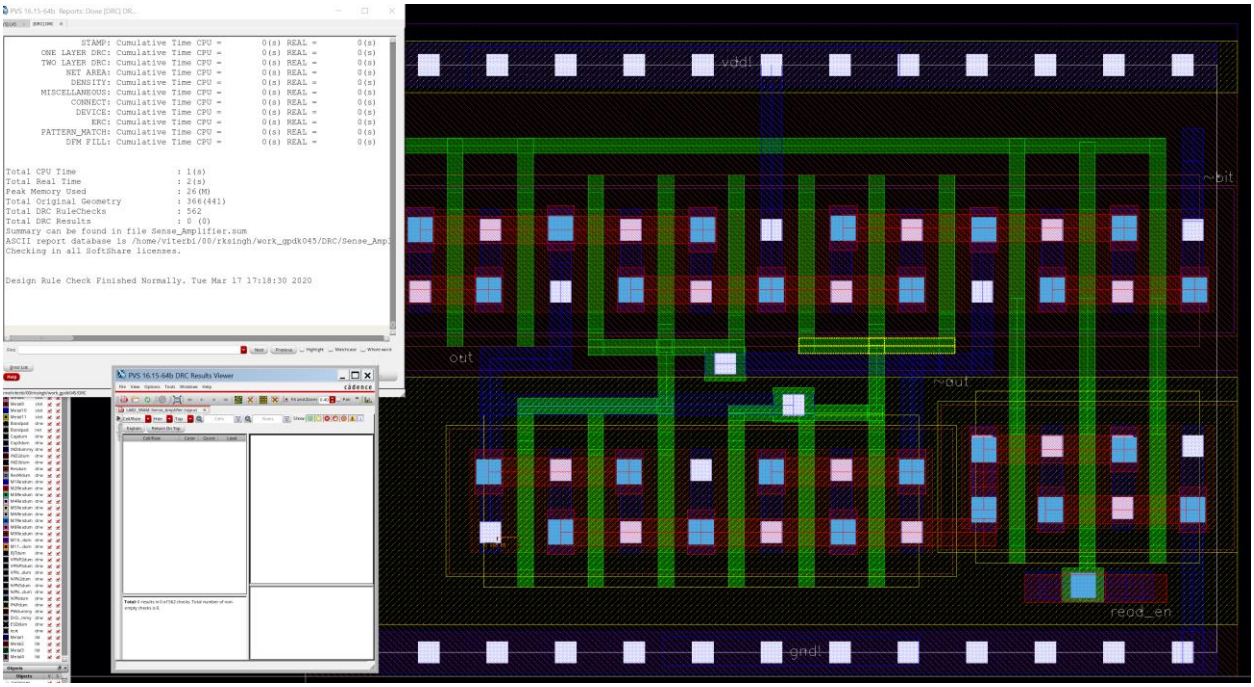
Fiq 19: Sense Amplifier Schematic Sizing: PMOS W = 900n, L=45n. NMOS W = 900n, L=45n

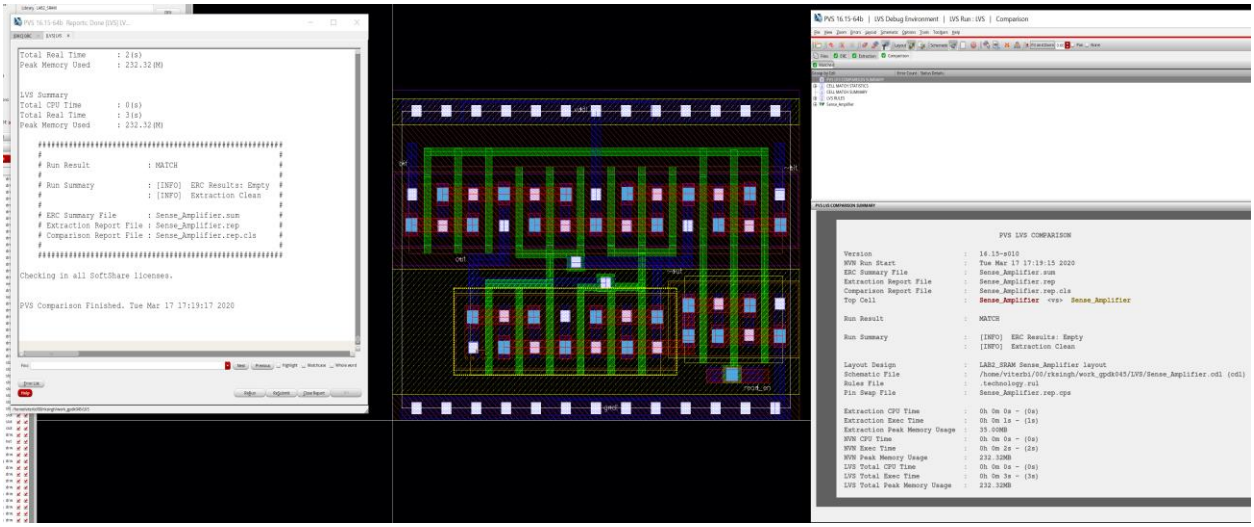


Fiq 19: Symbol Sense Amplifier

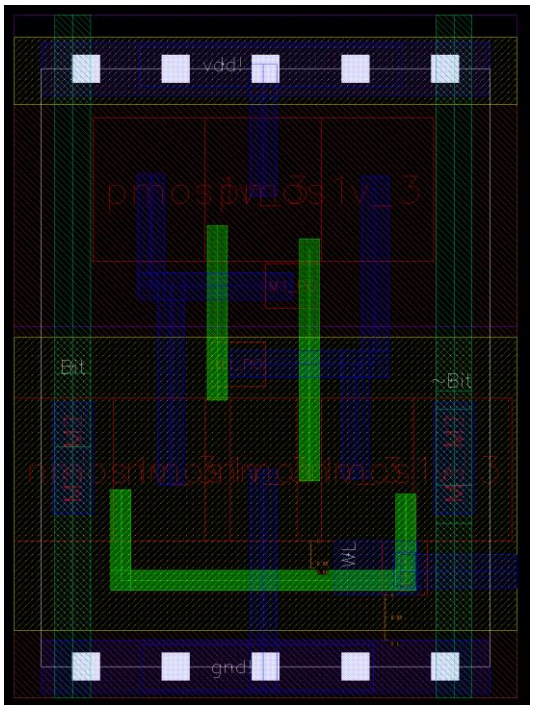


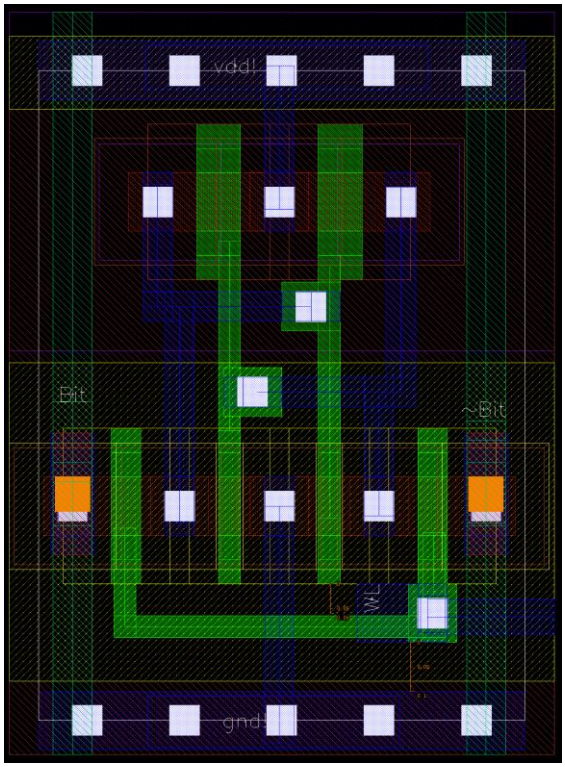
Fiq 20: Sense Amplifier Layout


STAMP: Cumulative Time CPU = 0(s) REAL = 0(s)
ONE LAYER DRC: Cumulative Time CPU = 0(s) REAL = 0(s)
TWO LAYER DRC: Cumulative Time CPU = 0(s) REAL = 0(s)
NET AREA: Cumulative Time CPU = 0(s) REAL = 0(s)
DENSITY: Cumulative Time CPU = 0(s) REAL = 0(s)
MISCELLANEOUS: Cumulative Time CPU = 0(s) REAL = 0(s)
CONNECT: Cumulative Time CPU = 0(s) REAL = 0(s)
DEVICE: Cumulative Time CPU = 0(s) REAL = 0(s)
ERC: Cumulative Time CPU = 0(s) REAL = 0(s)
PATTERN_MATCH: Cumulative Time CPU = 0(s) REAL = 0(s)
DFM FILL: Cumulative Time CPU = 0(s) REAL = 0(s)
Total CPU Time : 1(s)
Total Real Time : 2(s)
Peak Memory Used : 26(M)
Total Original Geometry : 366(441)
Total DRC RuleChecks : 562
Total DRC Results : 5 (0)
Summary can be found in file Sense_Amplifier.sum
Checking in all SoftShare licenses.
Design Rule Check Finished Normally. Tue Mar 17 17:18:30 2020
PVS 16.15-64b DRC Results Viewer
vdd!
out
~out
read_en
gnd!

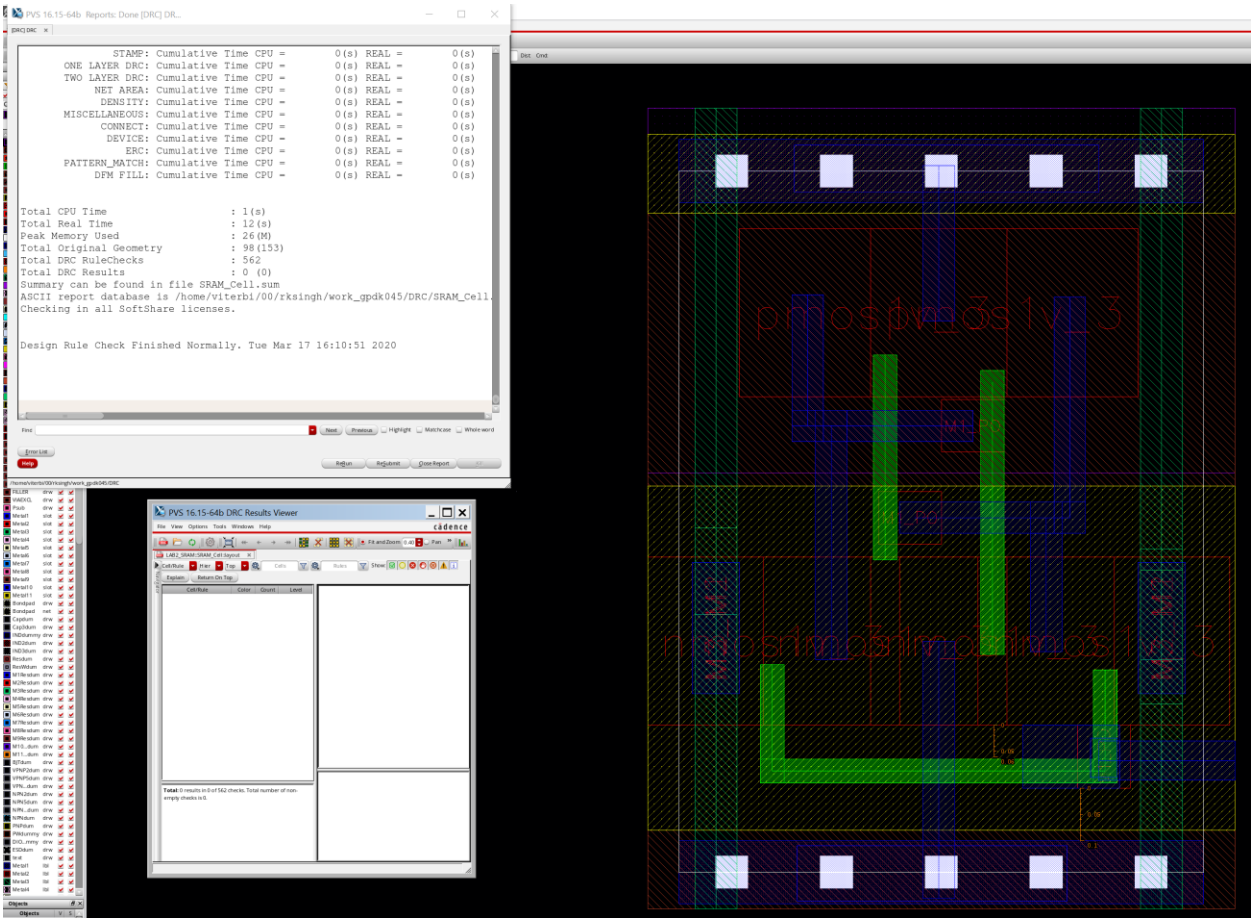

Total Real Time : 2(s)
Peak Memory Used : 232.32(M)
LVS Summary
Total CPU Time : 0(s)
Total Real Time : 3(s)
Peak Memory Used : 232.32(M)
Run Result : MATCH
Run Summary : [INFO] ERC Results: Empty
: [INFO] Extraction Clean
ERC Summary File : Sense_Amplifier.sum
Extraction Report File : Sense_Amplifier.rep
Comparison Report File : Sense_Amplifier.rep.cls
Checking in all SoftShare licenses.
PVS Comparison Finished. Tue Mar 17 17:19:17 2020
PVS LVS COMPARISON
Version : 16.15-s010
NVN Run Start : Tue Mar 17 17:19:15 2020
ERC Summary File : Sense_Amplifier.sum
Extraction Report File : Sense_Amplifier.rep
Comparison Report File : Sense_Amplifier.rep.cls
Top Cell : Sense_Amplifier <vs> Sense_Amplifier
Run Result : MATCH
Run Summary : [INFO] ERC Results: Empty
: [INFO] Extraction Clean
Layout Design : LAB2_SRAM Sense_Amplifier layout
Schematic File : /home/viterbi/00/rksingh/work_gpdk045/LVS/Sense_Amplifier.cdl (cdl)
Rules File : .technology.rul
Pin Swap File : Sense_Amplifier.rep.cps
Extraction CPU Time : 0h 0m 0s - (0s)
Extraction Exec Time : 0h 0m 1s - (1s)
Extraction Peak Memory Usage : 35.00MB
NVN CPU Time : 0h 0m 0s - (0s)
NVN Exec Time : 0h 0m 2s - (2s)
NVN Peak Memory Usage : 232.32MB
LVS Total CPU Time : 0h 0m 0s - (0s)
LVS Total Exec Time : 0h 0m 3s - (3s)
LVS Total Peak Memory Usage : 232.32MB

SRAM Layout Design–

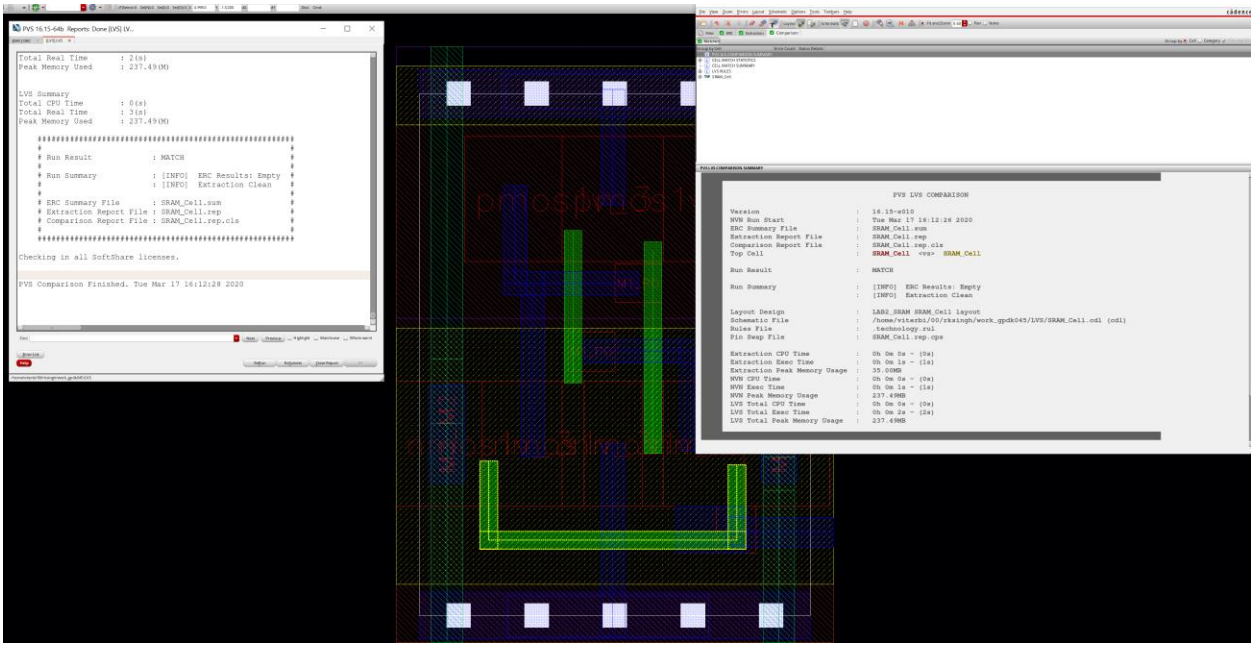




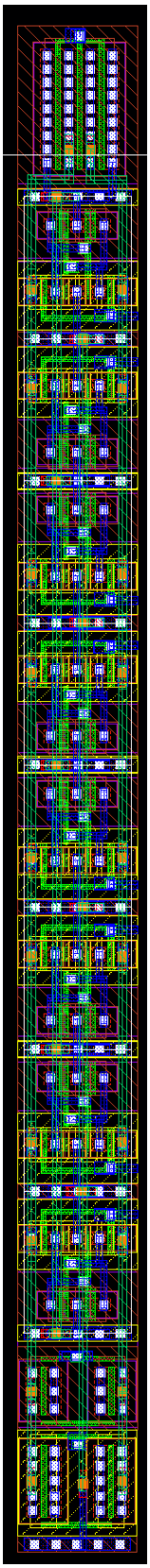
Fiq 1 : 1- Bit Full Adder Layout



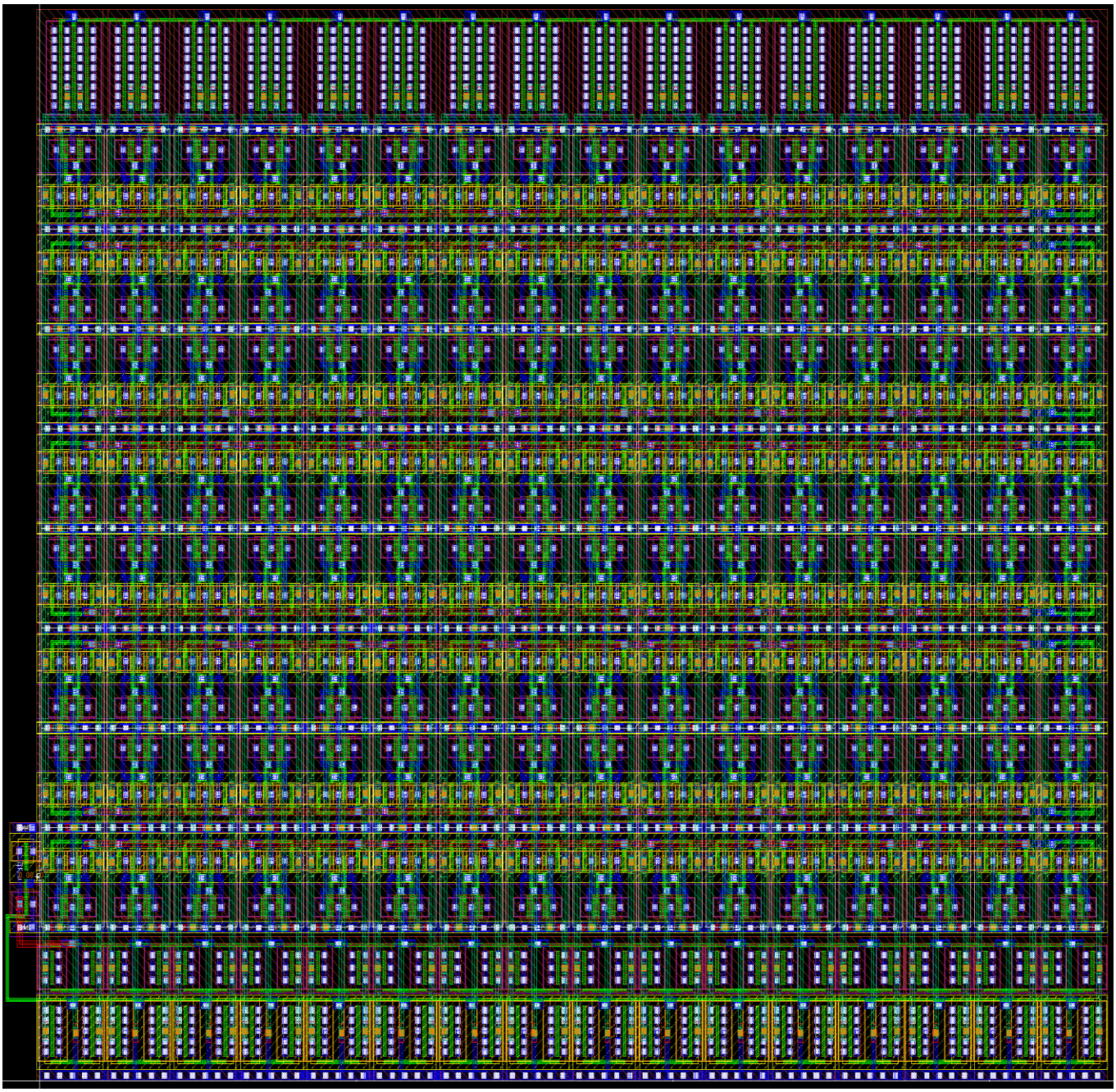
SRAM Cell DRC Check



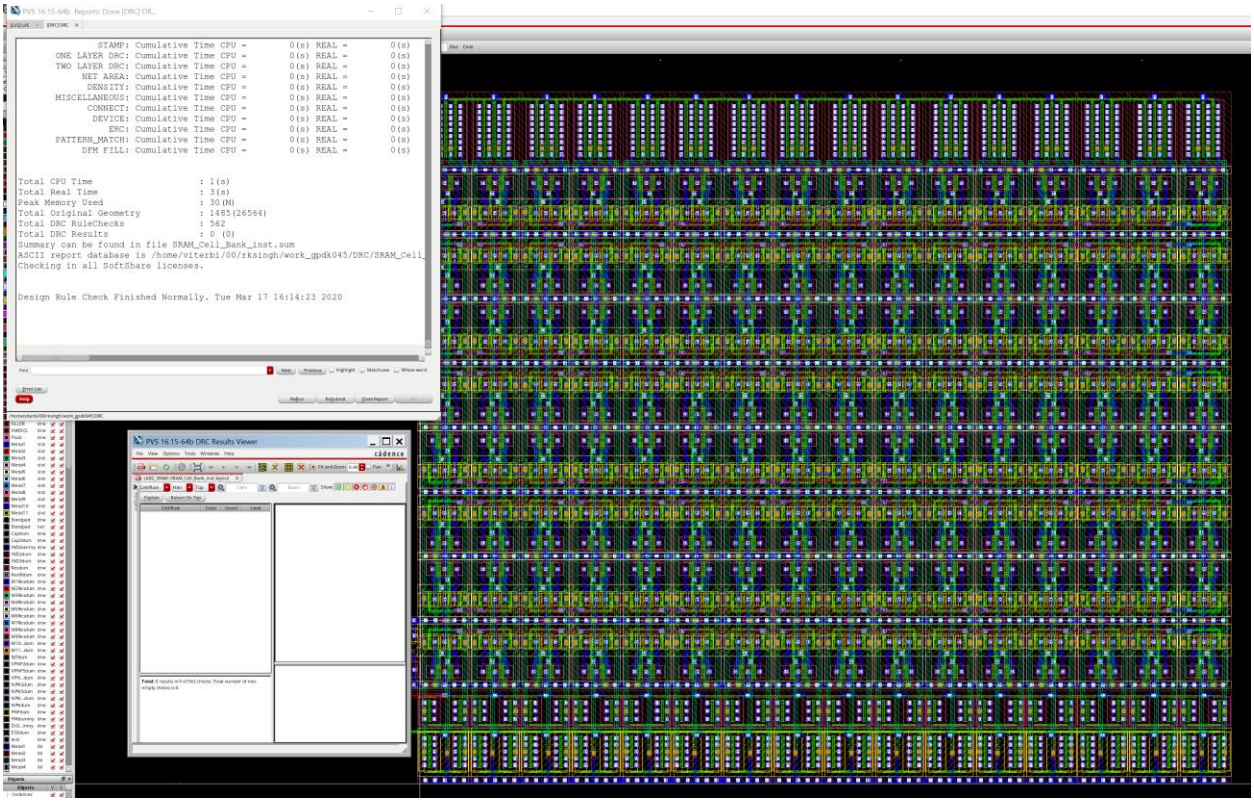
SRAM LVS Check



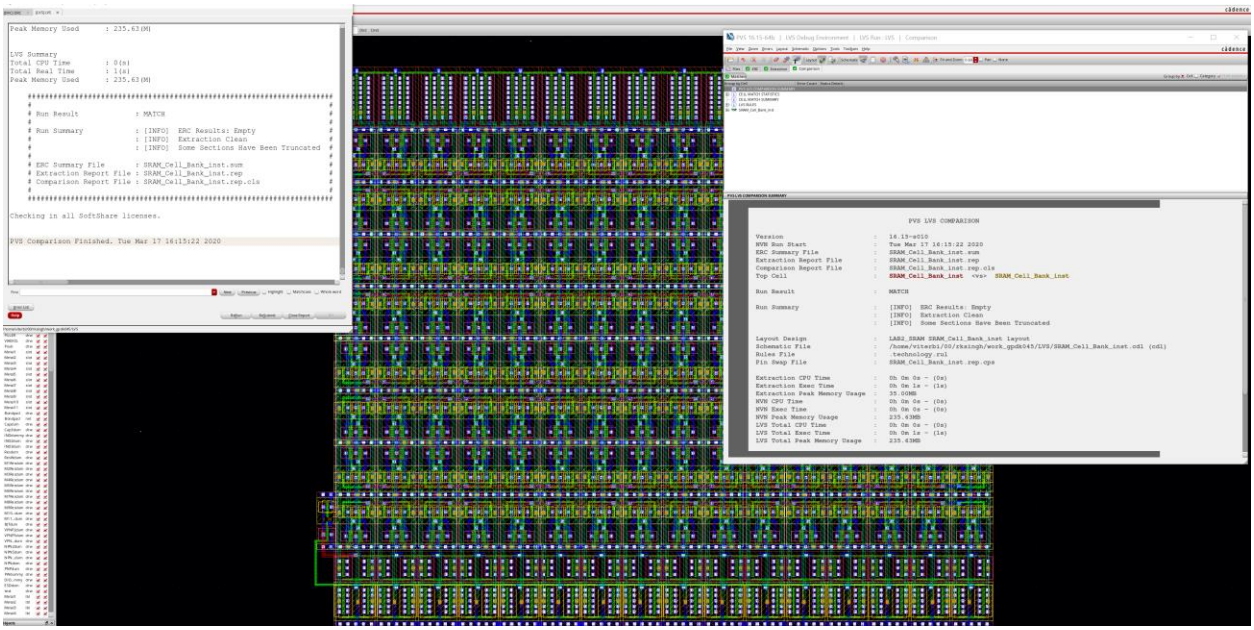
Fiq 2: 1 SRAM Column with Pre-Charge and READ/WRITE Muxes



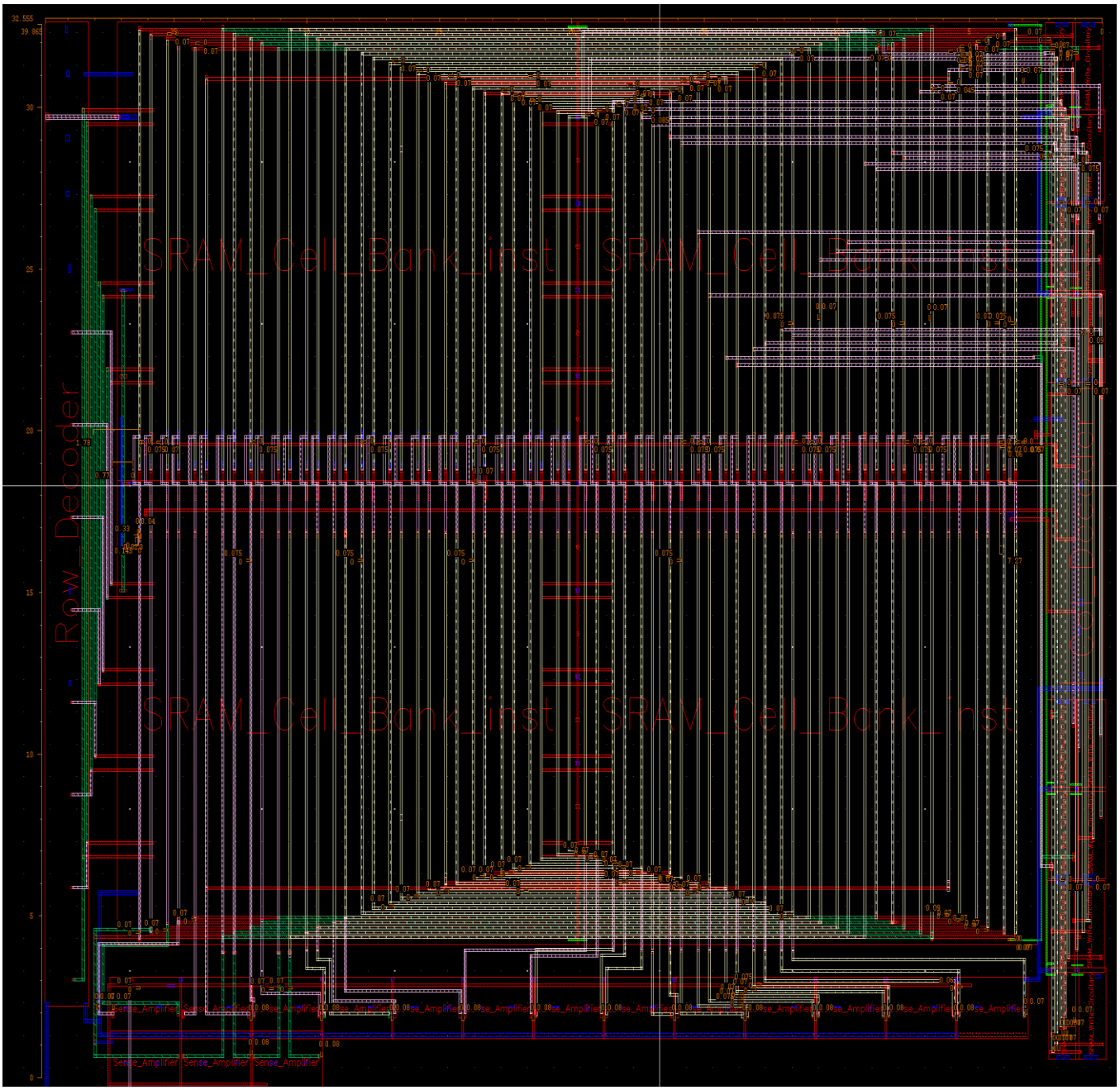
Fiq 2: 1 SRAM Bank Layout



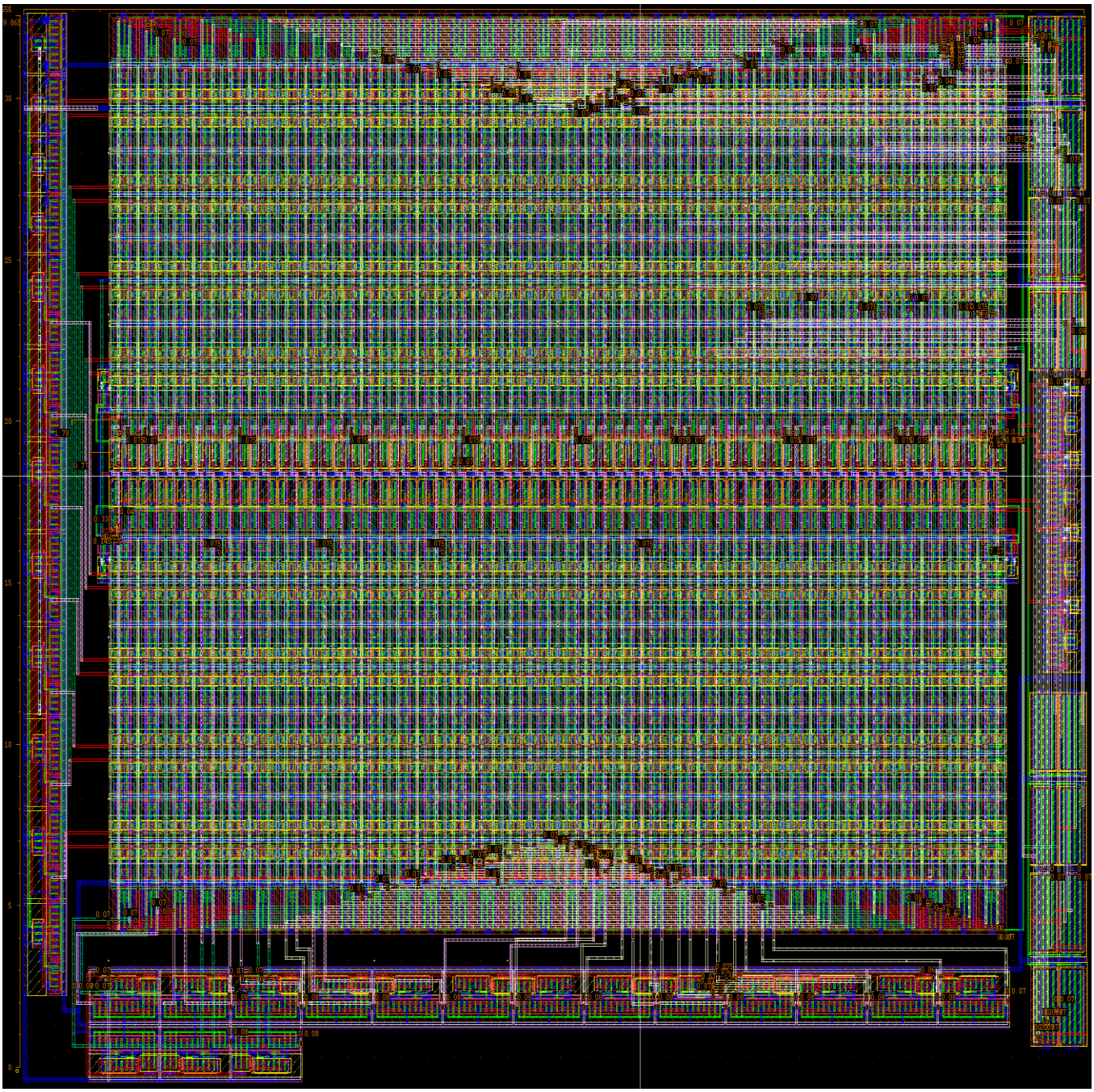
SRAM 1 Bank DRC Check



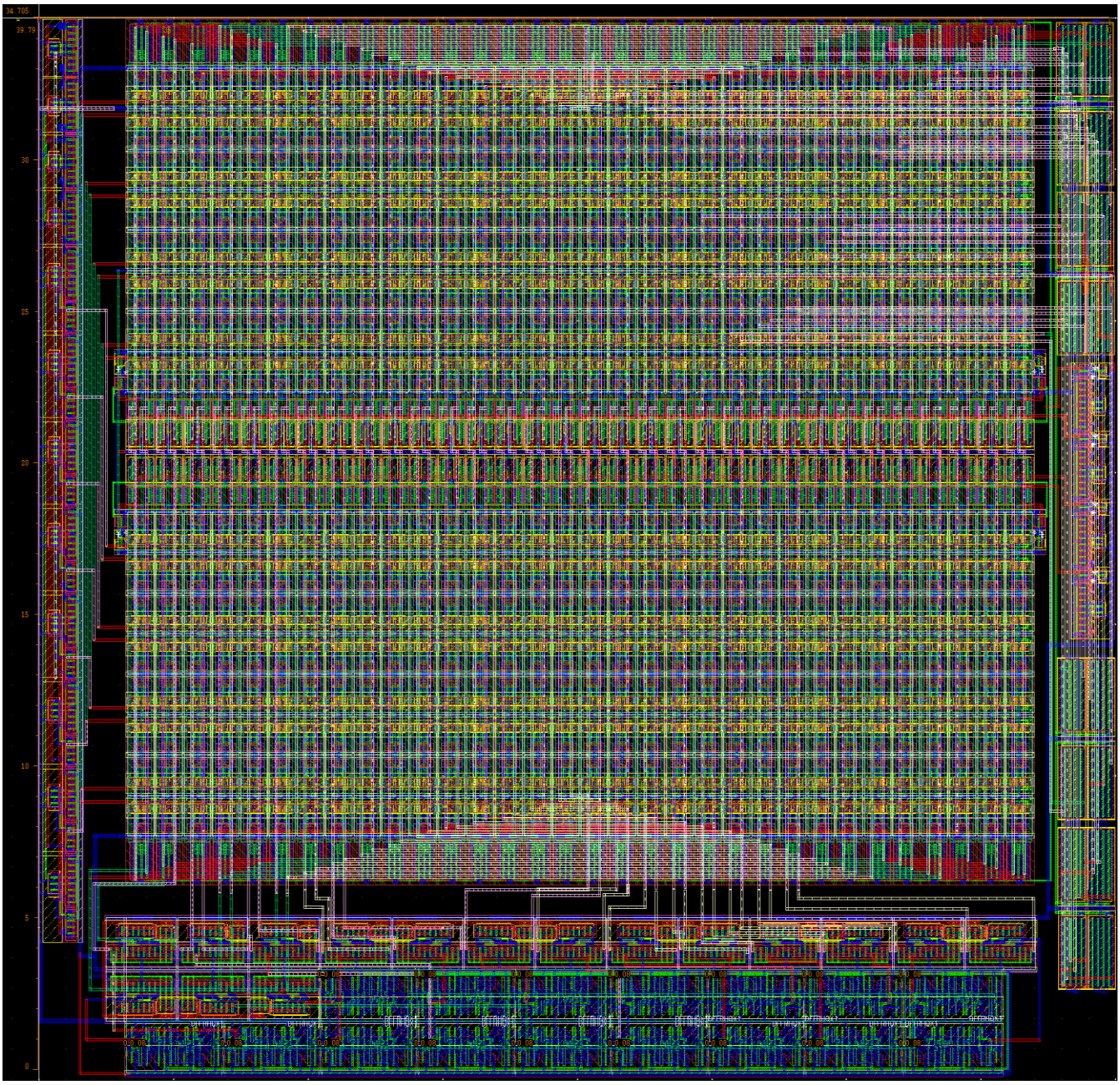
SRAM 1 Bank LVS Check



Fiq 3: 4 Bank 512 SRAM Bank Layout without DFF



Fiq 4: 4 Bank 512 SRAM Bank Layout without DFF



Fiq 5: 4 Bank 512 SRAM Bank Layout with DFF

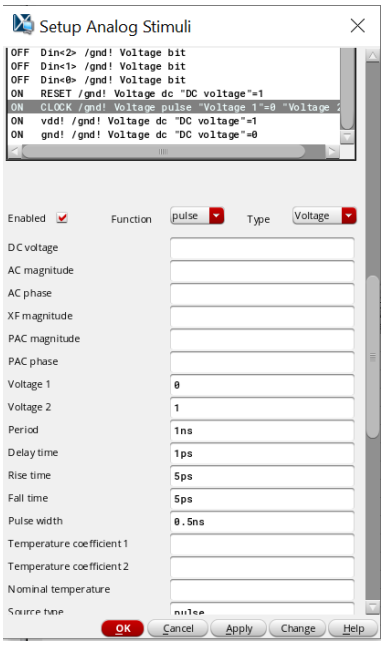
Functional Verification:

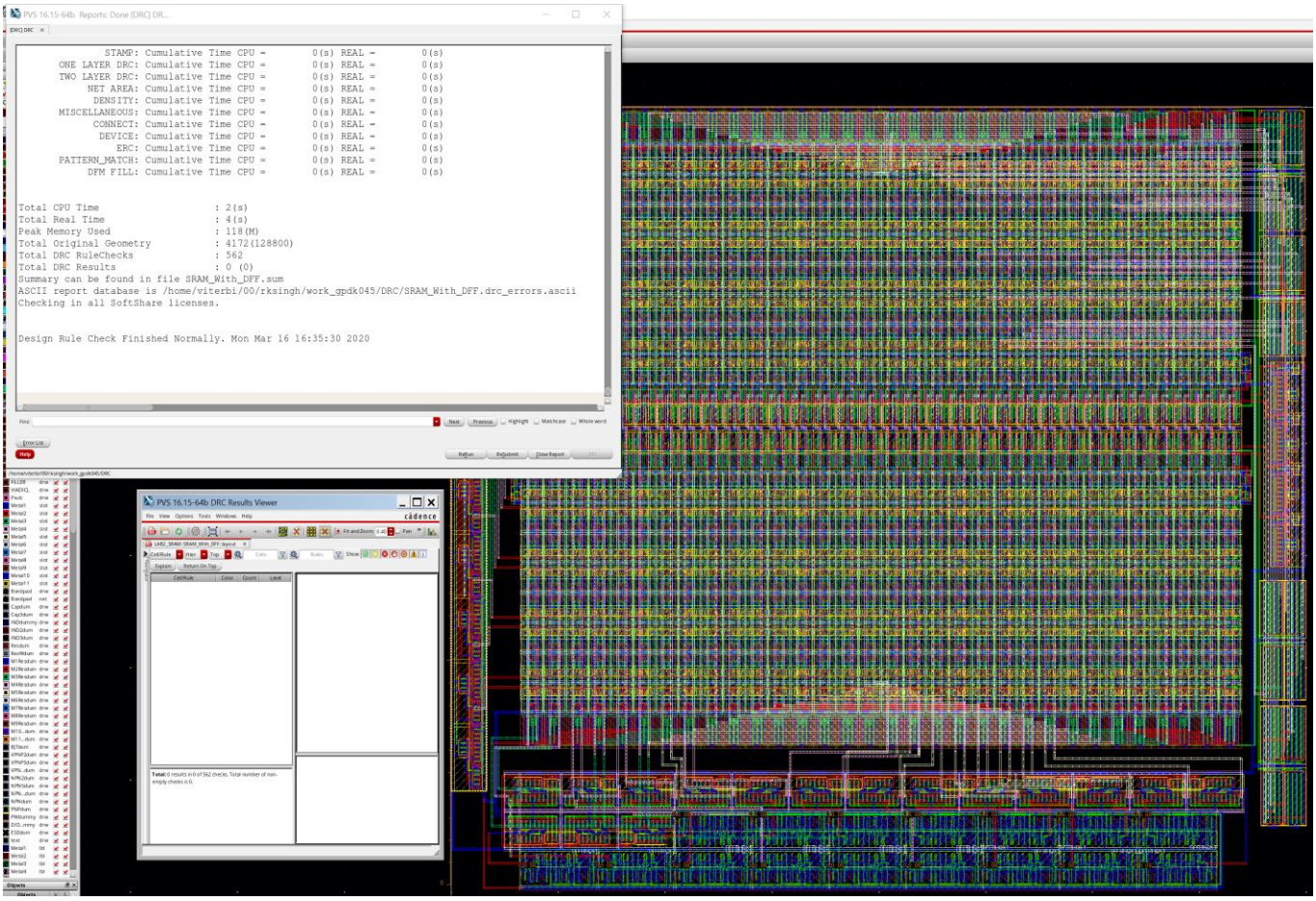
```
radix 1 1 1 1 4 4 4 4 4 4 4 4 4 3 2 3 2
io i i i i i i i i i i i i i i i i
vname write_en D_en read_en PRE in<[15:12]> in<[11:8]> in<[7:4]> in<[3:0]> ~in<[15:12]> ~in<[11:8]> ~in<[7:4]> ~in<[3:0]> A<[4:2]> A<[1:0]> ~A<[4:2]> ~A<[1:0]>
tunit ns
slope 0.005
vih 1.0
vil 0.0
trise 0.005
tfall 0.005

0    0 0 1 0 0 0 0 0 F F F F 0 0 7 3        ;precharge. Active low.
4    1 0 0 1 9 8 4 C 6 7 B 3 2 2 5 1        ;write for 1.
8    0 0 1 0 0 0 0 0 F F F F 0 0 7 3
12   1 0 0 1 2 1 1 B D E E 4 5 2 2 1        ;write 2
16   0 0 1 0 0 0 0 0 F F F F 0 0 7 3
20   1 0 0 1 4 C 3 6 B 3 C 9 2 2 5 1        ;write 3
24   0 0 1 0 0 0 0 0 F F F F 0 0 7 3
28   1 0 0 1 E 1 5 C 1 E A 3 4 3 3 0        ;write 4
32   0 0 1 0 0 0 0 0 F F F F 0 0 7 3
36   1 0 0 1 5 F 2 B A 0 D 4 7 3 0 0        ;write 5
40   0 0 1 0 0 0 0 0 F F F F 0 0 7 3
44   1 0 0 1 9 0 5 4 6 F A B 5 0 2 3        ;write 6
48   0 0 1 0 0 0 0 0 F F F F 0 0 7 3
52   1 0 0 1 F F 8 3 0 0 7 C 1 3 6 0        ;write 7
56   0 0 1 0 0 0 0 0 F F F F 0 0 7 3
60   1 0 0 1 1 9 D 7 E 6 2 8 4 1 3 2        ;write 8
64   0 0 1 0 0 0 0 0 F F F F 0 0 7 3
68   1 0 0 1 1 A 2 B E 5 D 4 6 3 1 0        ;write 9
72   0 0 1 0 0 0 0 0 F F F F 0 0 7 3
76   1 0 0 1 1 4 3 B E B C 4 2 3 5 0        ;write 0
80   0 1 0 0 0 0 0 0 F F F F 5 0 2 3        ;precharge
84   0 0 0 1 0 0 0 0 F F F F 5 0 2 3        ;precharge
88   0 0 1 0 0 0 0 0 F F F F 5 0 2 3        ;read 6
92   0 1 0 0 0 0 0 0 F F F F 5 0 2 3        ;precharge
96   0 0 0 1 0 0 0 0 F F F F 5 0 2 3        ;precharge
100  0 0 1 0 0 0 0 0 F F F F 5 0 2 3        ;read 6
104  0 1 0 0 0 0 0 0 F F F F 4 1 3 2        ;precharge
108  0 0 0 1 0 0 0 0 F F F F 4 1 3 2        ;precharge
112  0 0 1 0 0 0 0 0 F F F F 4 1 3 2        ;read 8
116  0 1 0 0 0 0 0 0 F F F F 6 3 1 0        ;precharge
120  0 0 0 1 0 0 0 0 F F F F 6 3 1 0        ;precharge
124  0 0 1 0 0 0 0 0 F F F F 6 3 1 0        ;read 9
128  0 1 0 0 0 0 0 0 F F F F 2 2 5 1        ;precharge
132  0 0 0 1 0 0 0 0 F F F F 2 2 5 1        ;precharge
136  0 0 1 0 0 0 0 0 F F F F 2 2 5 1        ;read 3
140  0 1 0 0 0 0 0 0 F F F F 4 1 3 2        ;precharge
144  0 0 0 1 0 0 0 0 F F F F 4 1 3 2        ;precharge
148  0 0 1 0 0 0 0 0 F F F F 4 1 3 2        ;read 8
152  0 1 0 0 0 0 0 0 F F F F 1 3 6 0        ;precharge
156  0 0 0 1 0 0 0 0 F F F F 1 3 6 0        ;precharge
160  0 0 1 0 0 0 0 0 F F F F 1 3 6 0        ;read 7
164  0 1 0 0 0 0 0 0 F F F F 4 1 3 2        ;precharge
168  0 0 0 1 0 0 0 0 F F F F 4 1 3 2        ;precharge
172  0 0 1 0 0 0 0 0 F F F F 4 1 3 2        ;read 8
176  0 1 0 0 0 0 0 0 F F F F 4 3 3 0        ;precharge
180  0 0 0 1 0 0 0 0 F F F F 4 3 3 0        ;precharge
184  0 0 1 0 0 0 0 0 F F F F 4 3 3 0        ;read 4
188  0 1 0 0 0 0 0 0 F F F F 2 2 5 1        ;precharge
192  0 0 0 1 0 0 0 0 F F F F 2 2 5 1        ;precharge
196  0 0 1 0 0 0 0 0 F F F F 2 2 5 1        ;read 1
```

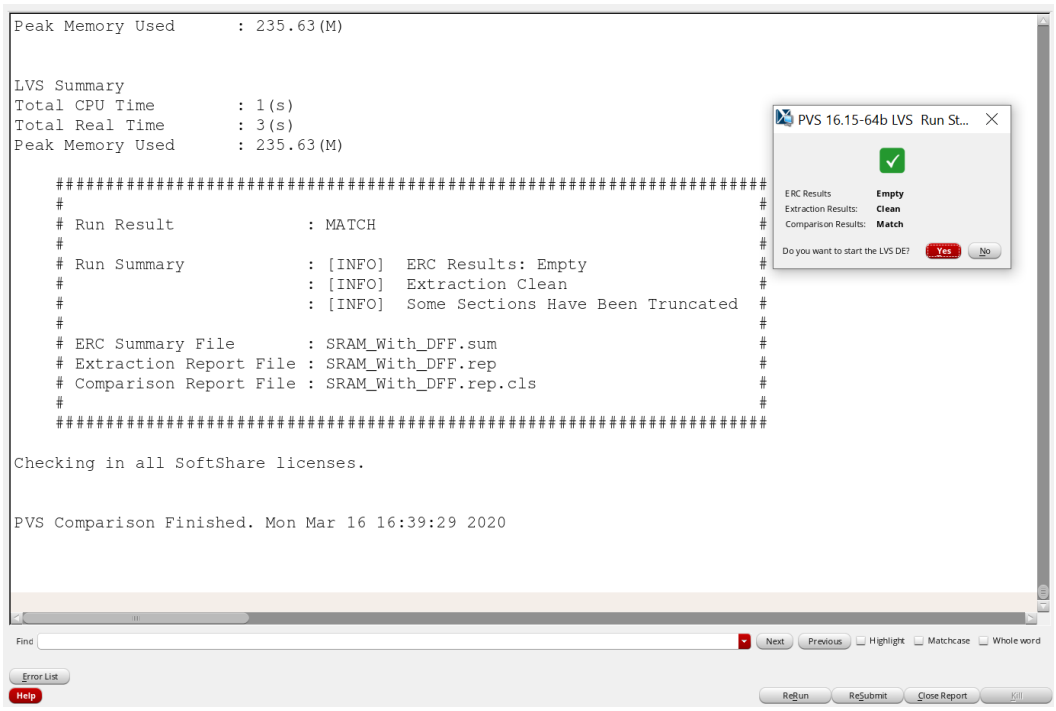
4_Bank_Final_USCID.vec

According to the data provided to us in the table of our assignment, we have deciphered that for writing a 1 we need to writ data 984c in address 2251(when represented in HEX) so whenever we need to read 1 we need to call or read the data written in the address 2251. Similarly as per instruction, we need to write our last digit of our USC ID first and then in alphabetically increasing order write the consecutive digits, likewise when performing reading we read the digits or values as per the same order as our USC ID, so we create a repository or a table of data being stored in particular addresses and whenever we need to read or call the data we call the particular address, and read the data stored in that address.

So upon verifying it, I concluded that my SRAM was executing successful reading and writing and to calculate the delay in reading and writing, I used markers to note the points where the write enable was going high and marked the point where the output was changing, i.e being high to low or vice-versa.


Setup Analog Stimuli
OFF Din<2> /gnd! Voltage bit
OFF Din<1> /gnd! Voltage bit
OFF Din<0> /gnd! Voltage bit
ON RESET /gnd! Voltage dc "DC voltage"=1
ON CLOCK /gnd! Voltage pulse "Voltage 1"=0 "Voltage 2
ON vdd! /gnd! Voltage dc "DC voltage"=1
ON gnd! /gnd! Voltage dc "DC voltage"=0
Enabled
Function
pulse
Type
Voltage
DC voltage
AC magnitude
AC phase
XF magnitude
PAC magnitude
PAC phase
Voltage 1
0
Voltage 2
1
Period
1ns
Delay time
1ps
Rise time
5ps
Fall time
5ps
Pulse width
0.5ns
Temperature coefficient 1
Temperature coefficient 2
Nominal temperature
Source type
pulse
OK
Cancel
Apply
Change
Help


STAMP: Cumulative Time CPU = 0(s) REAL = 0(s)
ONE LAYER DRC: Cumulative Time CPU = 0(s) REAL = 0(s)
TWO LAYER DRC: Cumulative Time CPU = 0(s) REAL = 0(s)
NET AREA: Cumulative Time CPU = 0(s) REAL = 0(s)
DENSITY: Cumulative Time CPU = 0(s) REAL = 0(s)
MISCELLANEOUS: Cumulative Time CPU = 0(s) REAL = 0(s)
CONNECT: Cumulative Time CPU = 0(s) REAL = 0(s)
DEVICE: Cumulative Time CPU = 0(s) REAL = 0(s)
ERC: Cumulative Time CPU = 0(s) REAL = 0(s)
PATTERN_MATCH: Cumulative Time CPU = 0(s) REAL = 0(s)
DFM FILL: Cumulative Time CPU = 0(s) REAL = 0(s)
Total CPU Time : 2(s)
Total Real Time : 4(s)
Peak Memory Used : 118(M)
Total Original Geometry : 4172(128800)
Total DRC RuleChecks : 562
Total DRC Results : 0 (0)
Summary can be found in file SRAM_With_DFF.sum
ASCII report database is /home/viterbi/00/rksingh/work_gpdk045/DRC/SRAM_With_DFF.drc_errors.ascii
Checking in all SoftShare licenses.
Design Rule Check Finished Normally. Mon Mar 16 16:35:30 2020
PVS 16.15-64b DRC Results Viewer



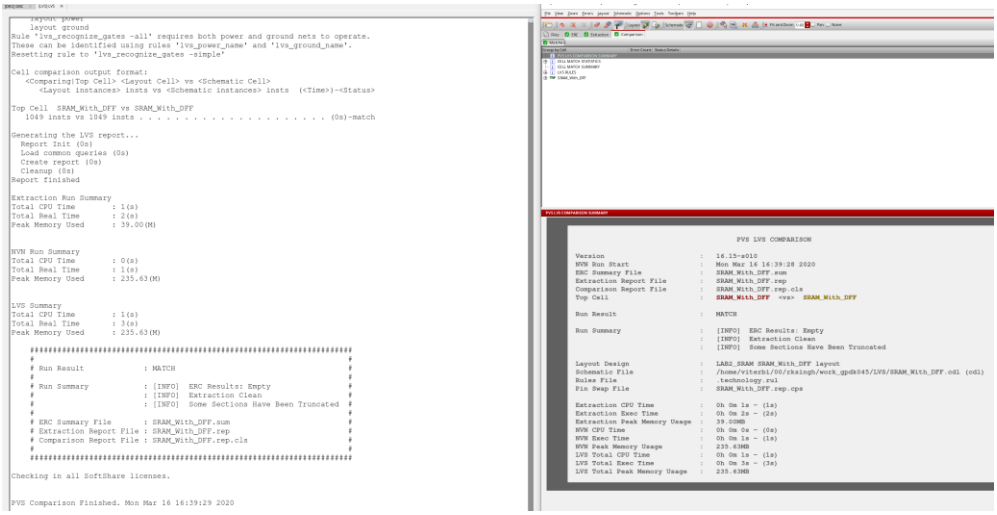
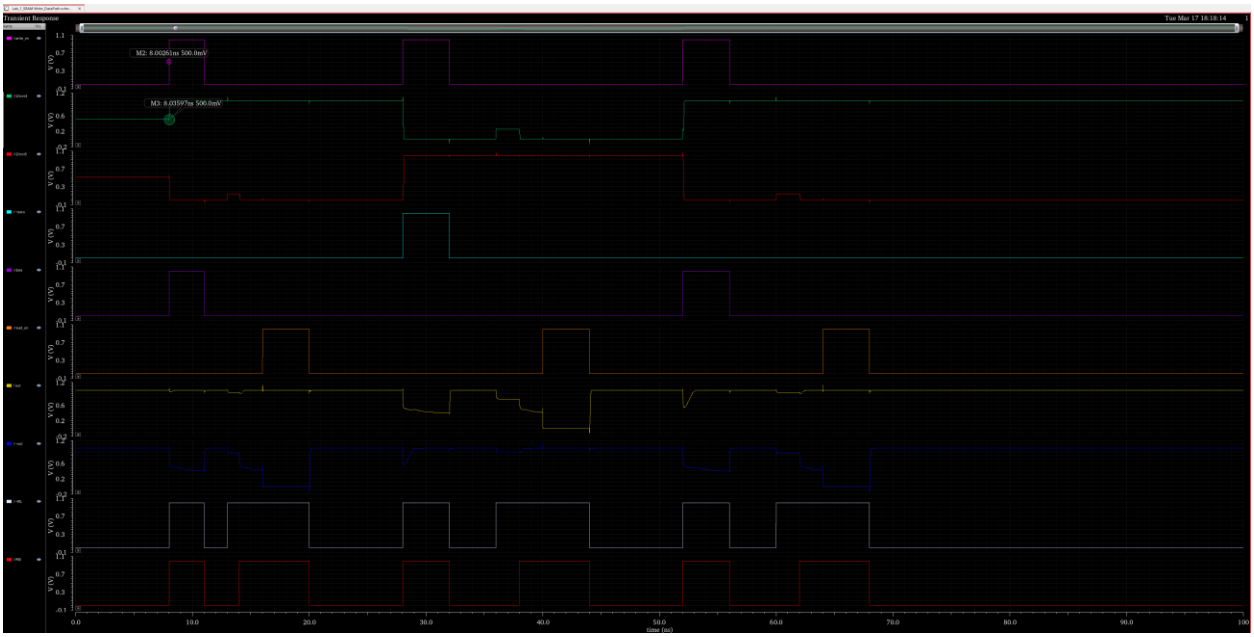


Table:

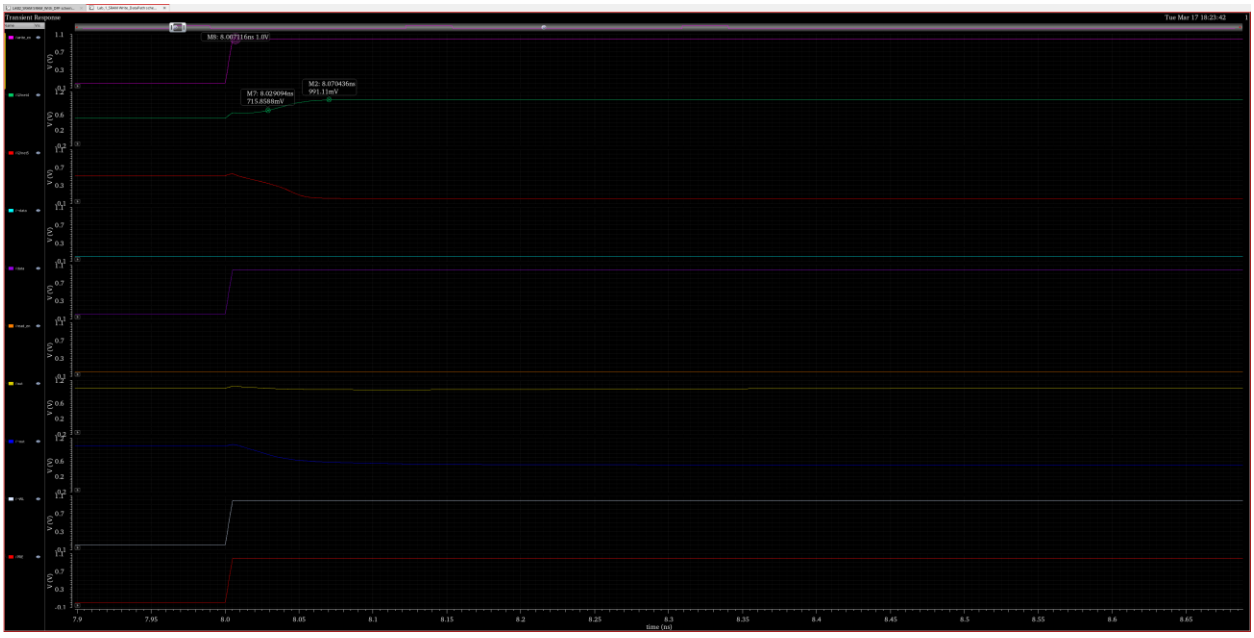
The table below highlights the SRAM parameters of the SRAM with the DFF.

Length	Width	Area
34.705um	39.79um	1381um2
Read Timing 1 SRAM Cell	0.064ns	
Read Timing 512 SRAM	1.08ns	
Write Timing 1 SRAM Cell	0.062ns	
Write Timing 512 SRAM	Approx 0.08ns	
Area X Delay	1.08ns * 1381 um2 = 1491.48 um2*ns	

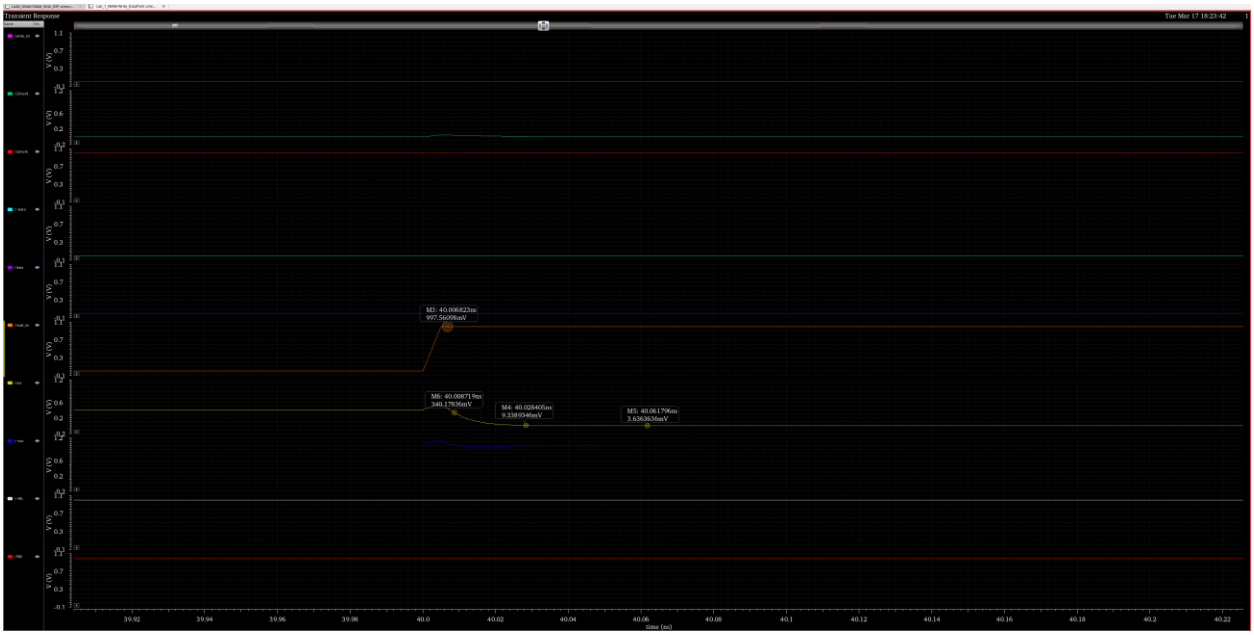
Below is the waveform of 6T SRAM Cell, in the graphs it's evident that we are writing 1, writing 0 and then writing 1 as well as reading after every consecutive writing. So, whenever there's any writing the PRE charge is set to 11 and the write-Enable is set to high, the write delay is calculated by finding the difference in the timing of the circuit when the write enable is high and when the data is written successfully in the voltage nodes. Similarly, for reading we calculate the read delay, by calculating the point where the read enable is high and the data is successfully read in the bit and bit bar.



Write and Read Timings

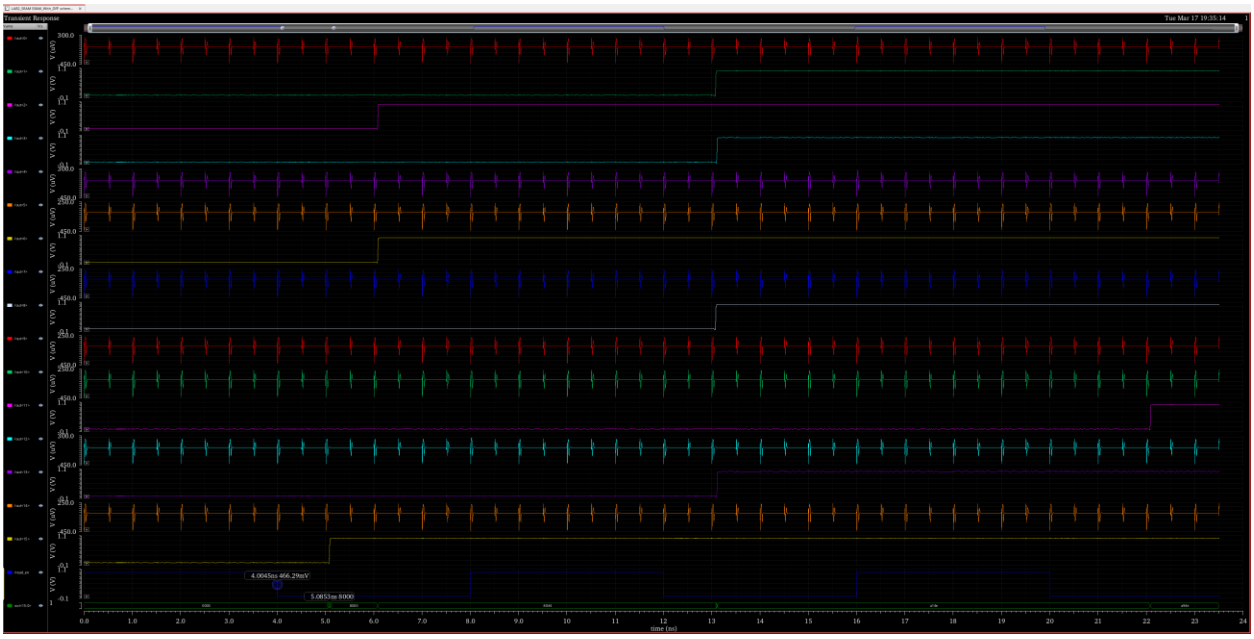


Write Delay = 8.0704ns-8.008ns =0.062ns



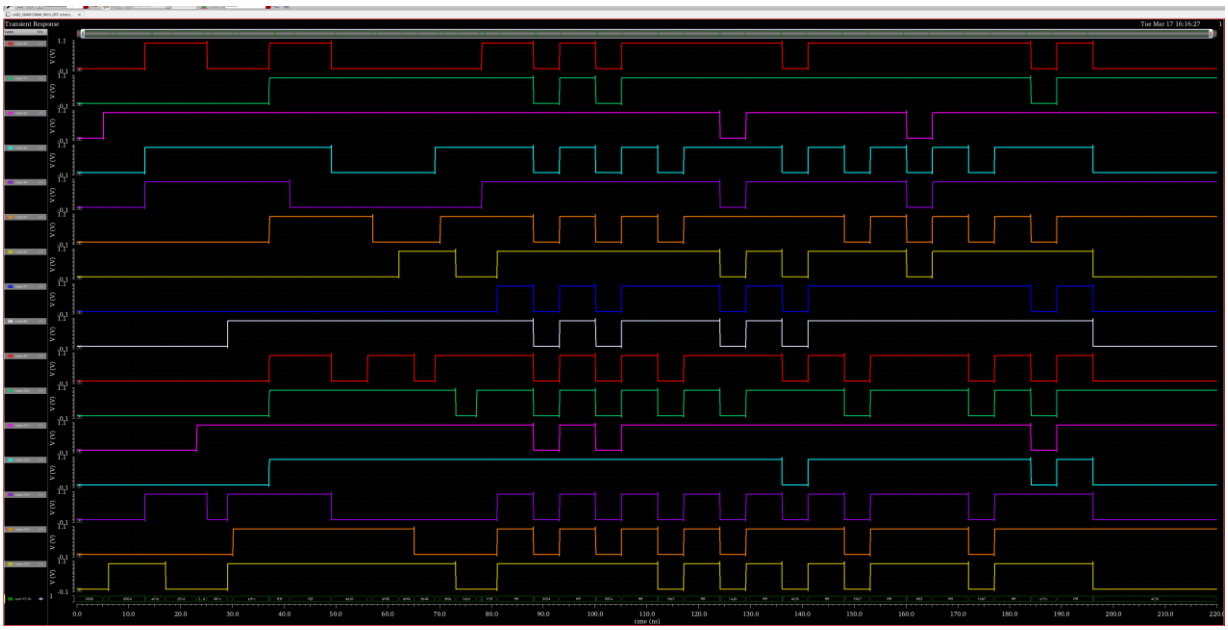
Read Delay = 40.062ns – 40.007ns=0.064ns

Below is the read delay of the 512 SRAM, we calculate the read delay by subtracting the time when the read enable is set to high and the time when the first Hex data is read successfully.

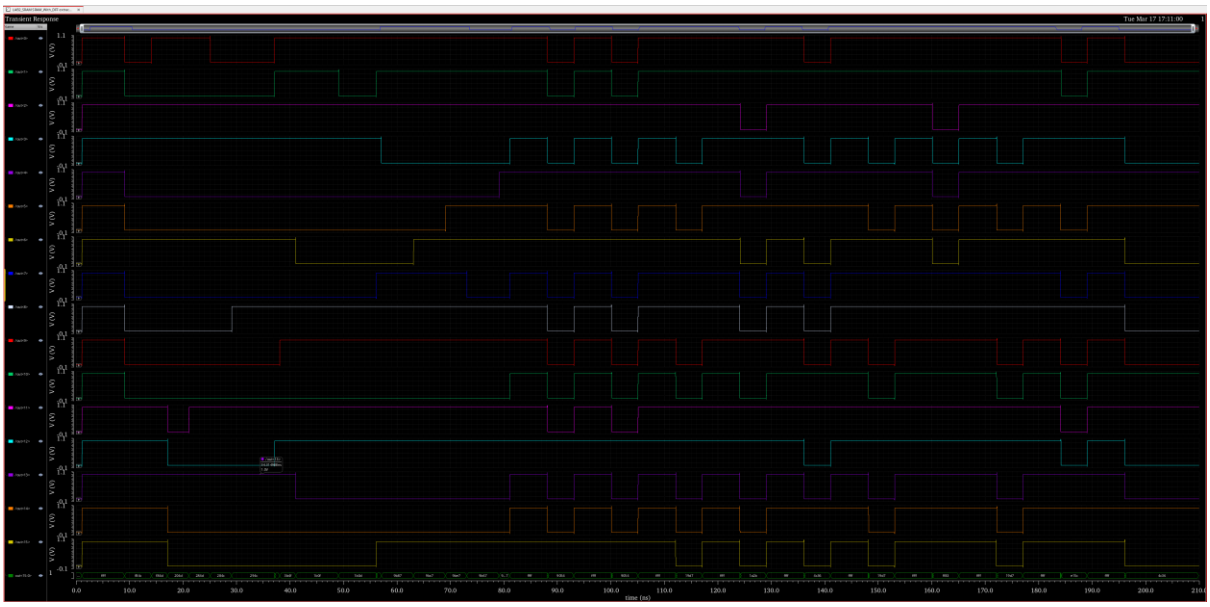


Read Delay of 512 SRAM: 5.0853ns-4.0045ns=1.08ns

Below is the waveform of the 512 SRAM Schematic and Layout. The Hexadecimal values mentioned in the below half of the waveform denote the data required to be verified for the waveform to work properly, both the schematic and the layout are providing correct values and hence the waveforms are verified to provide correct results.



Simulation of the 4 Bank with DFF – Schematic



Simulation of 4 Bank with DFF – Layout/Extracted

Result and Summary –

Hence the 512 SRAM having 4 banks are reading and writing successfully, having writing delay as 0.062ns for 1 SRAM cell and reading delay and writing delay as approximately 1ns. All the sizing parameters of the individual cells and blocks are determined as per the instructions provided in the pdf of this assignment.