Richard Chen
ESE370 Project 3: SRAM
I, Richard Chen, certify that I have complied with the University of Pennsylvania's Code of Academic Integrity in completing this project.

Milestone
Design: 16x4: 8x8, each row has 2 words of 4 bits

5t SRAM cell, pre charge to 0.5 VDD.

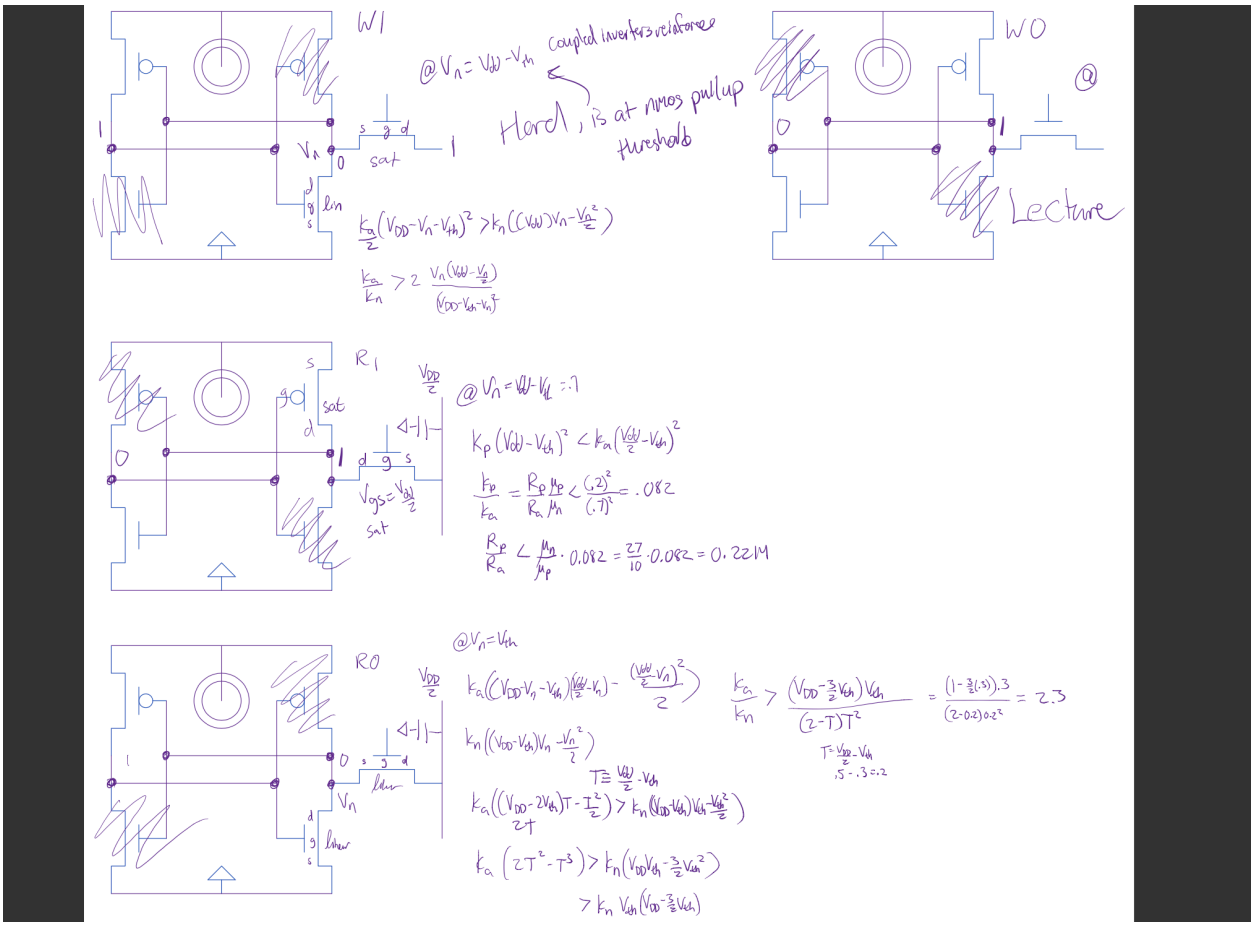
Bitline Capacitance Estimate: 8 rows * W_acc * gamma * C_0

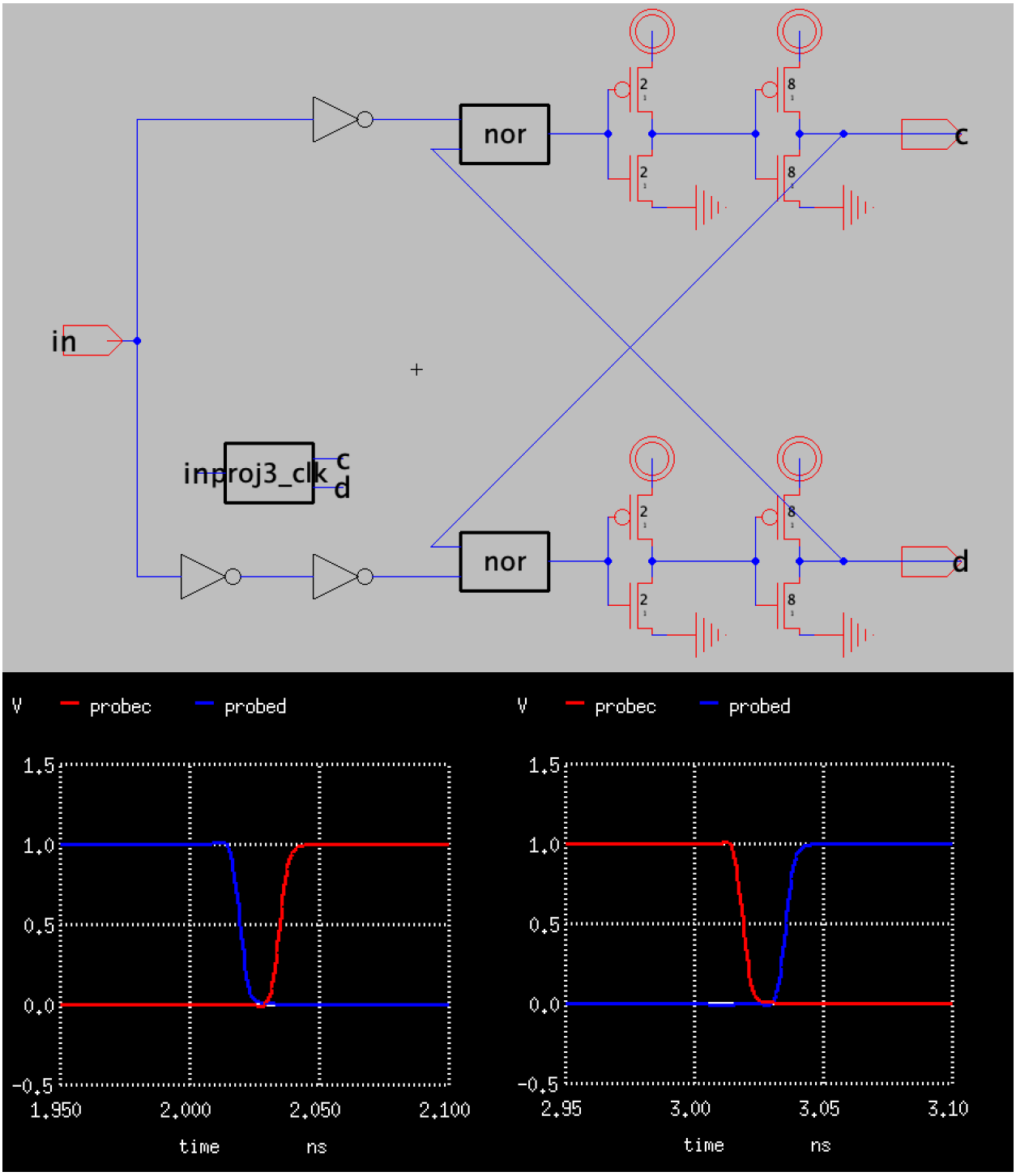
READ:
Precharge has to happen before a read
vdd/2 has to disconnect
read happens on other clock
clk1 AND write -> write
clk1 AND !write -> precharge
clk2 AND !write -> read
clk2 = word line; needs time to set BL/precharge beforehand but read as soon as word line set.

Cell Sizes: The access transistor was width 8 because width 4 was too small.

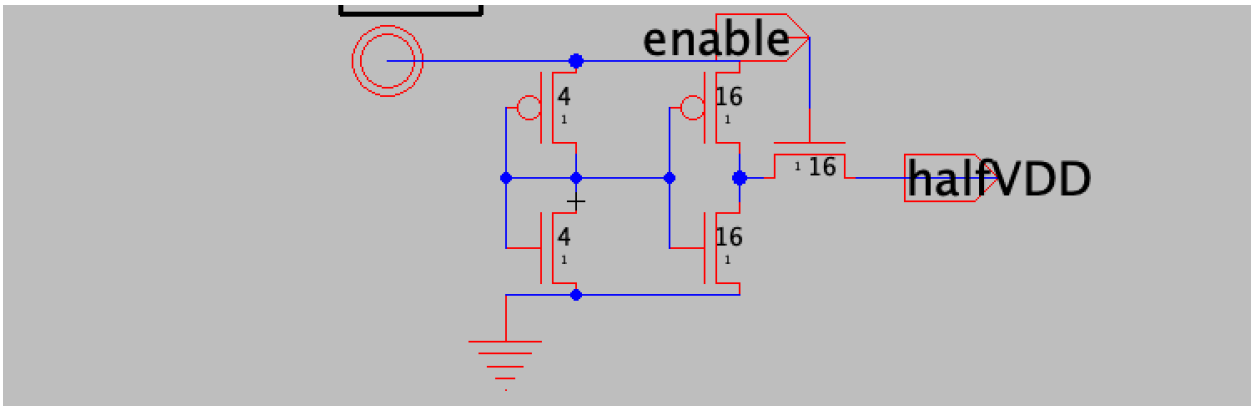
To write a 0, the transistor must be wide enough that Pullup Ratio < 1.8, just as in the lecture notes. To write a 1, The transistor must be wide, as pulling the node up to the voltage past short circuit voltage is the nmos pullup. In this case, it is 0.125, satisfying both.

To read a 0, the cell ratio of access/nmos must be greater than 2.3, which 8 is. To read a 1, the cell ratio of pmos/access must be less than 0.22, which 0.125 is.

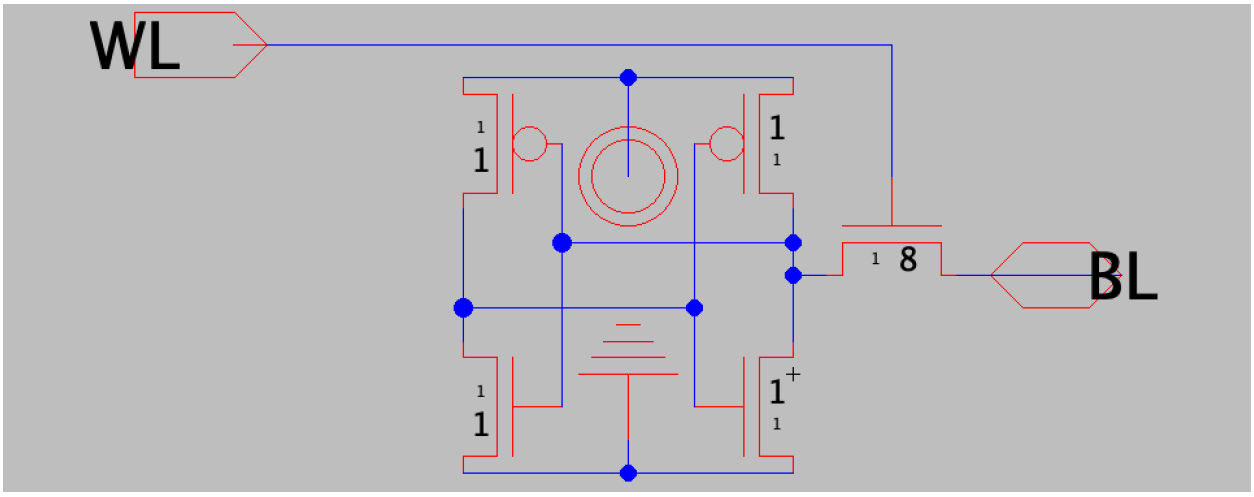



nor
nor
in
proj3_clk
c
d
c
d
2
8
V
probec
probed
1.5
1.0
0.5
0.0
-0.5
1.950
2.000
2.050
2.100
time
ns
V
probec
probed
1.5
1.0
0.5
0.0
-0.5
2.95
3.00
3.05
3.10
time
ns

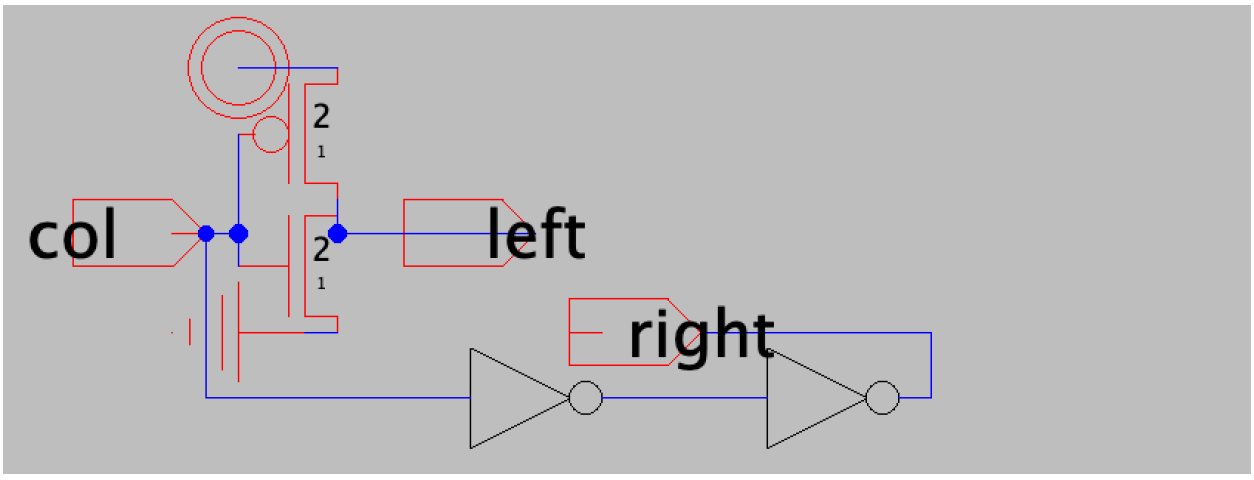
Clock



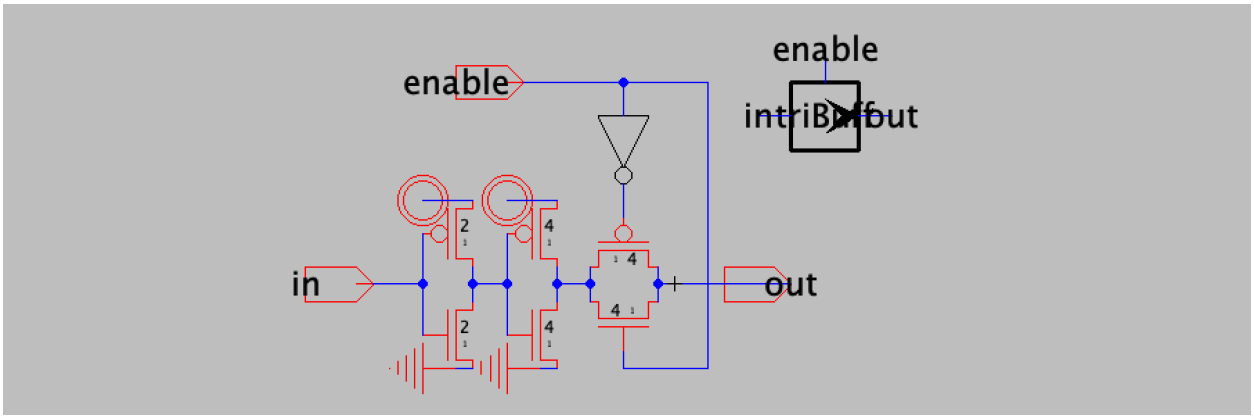
Reference Half VDD



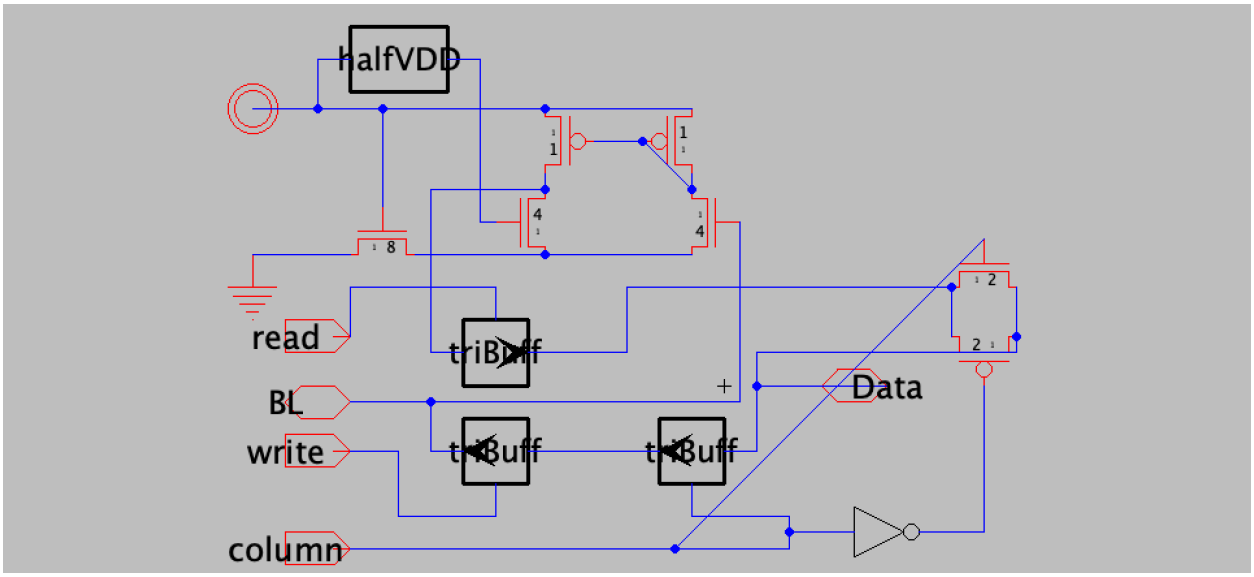
5t cell



Column decoder (unused; replaced with vdd signal in test)



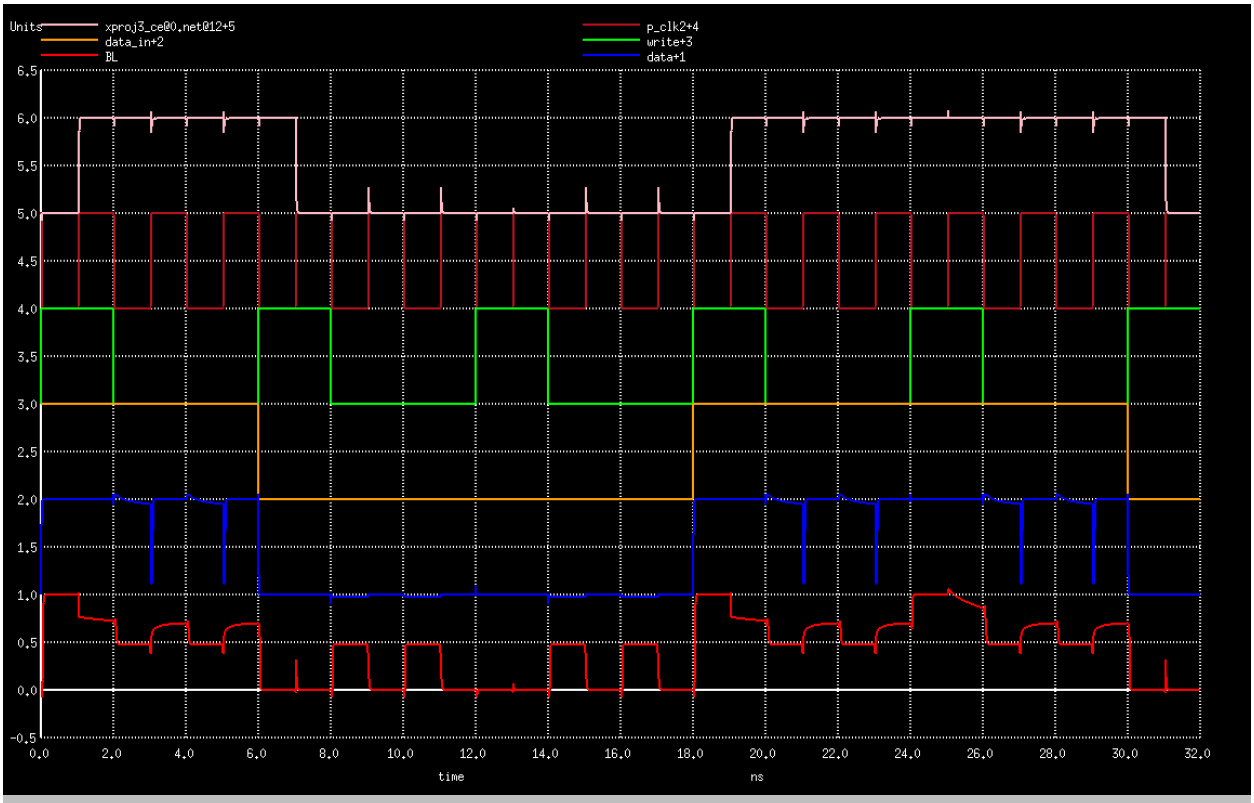
Tristate buffer

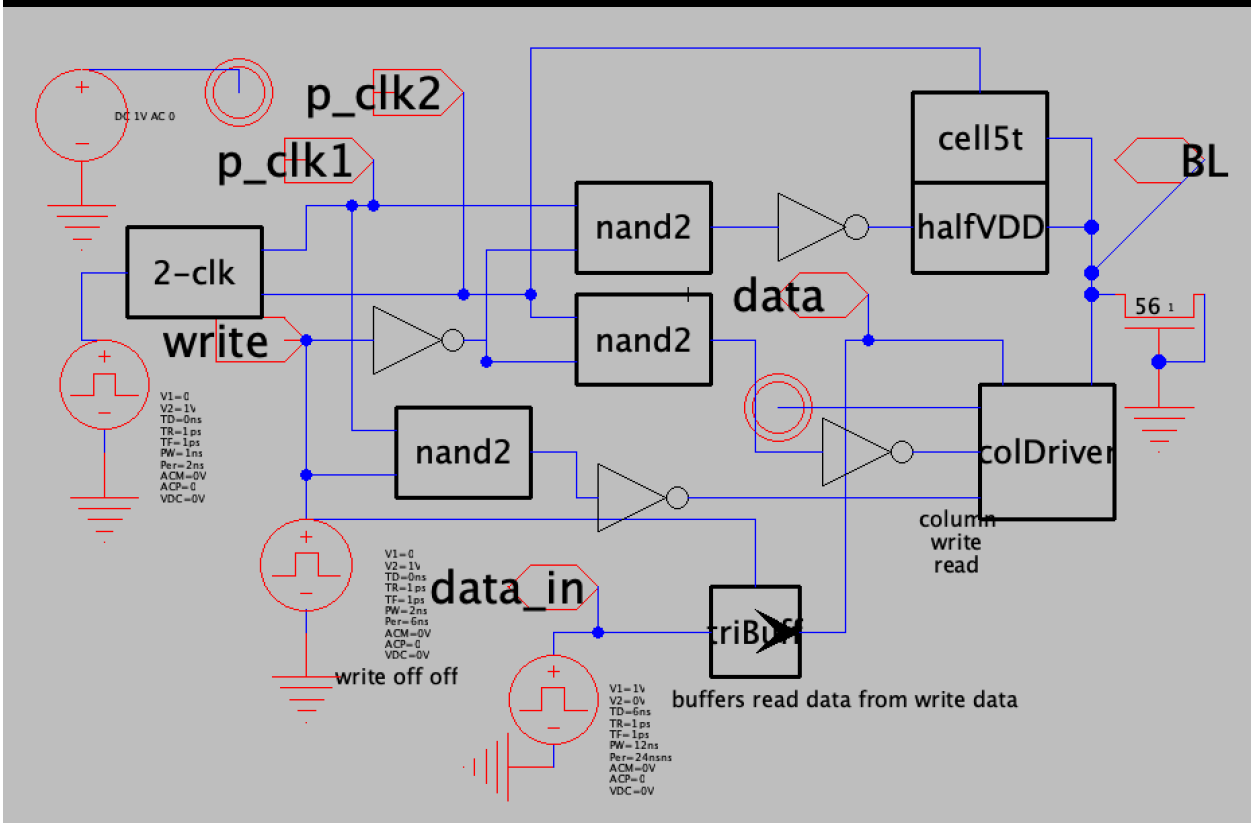


Column driver. Column is driven by a mux that selects which word on the word row. I split write and read because the slide example does not seem to lend itself well to clocked operation. The invariant that read and write are never high together must be maintained. In the whole test, I use NAND and inverters to simulate AND gates, Read is fed by clk2 and !write, write is fed by write and clk1. Thus the invariant is maintained.

To test the whole thing, w1 r1 r1, w0 r0 r0, w0 r0 r0, w1 r1 r1, w1 r1 r1.
This covers nondestructive reading, and all 4 cases of writing a new value while an old value exists.

Issues: tristate buffer buffers floating voltages
solution: use transmission gate -> Input voltage raised to VDD successfully.
Access Transistor was too small; now it works.



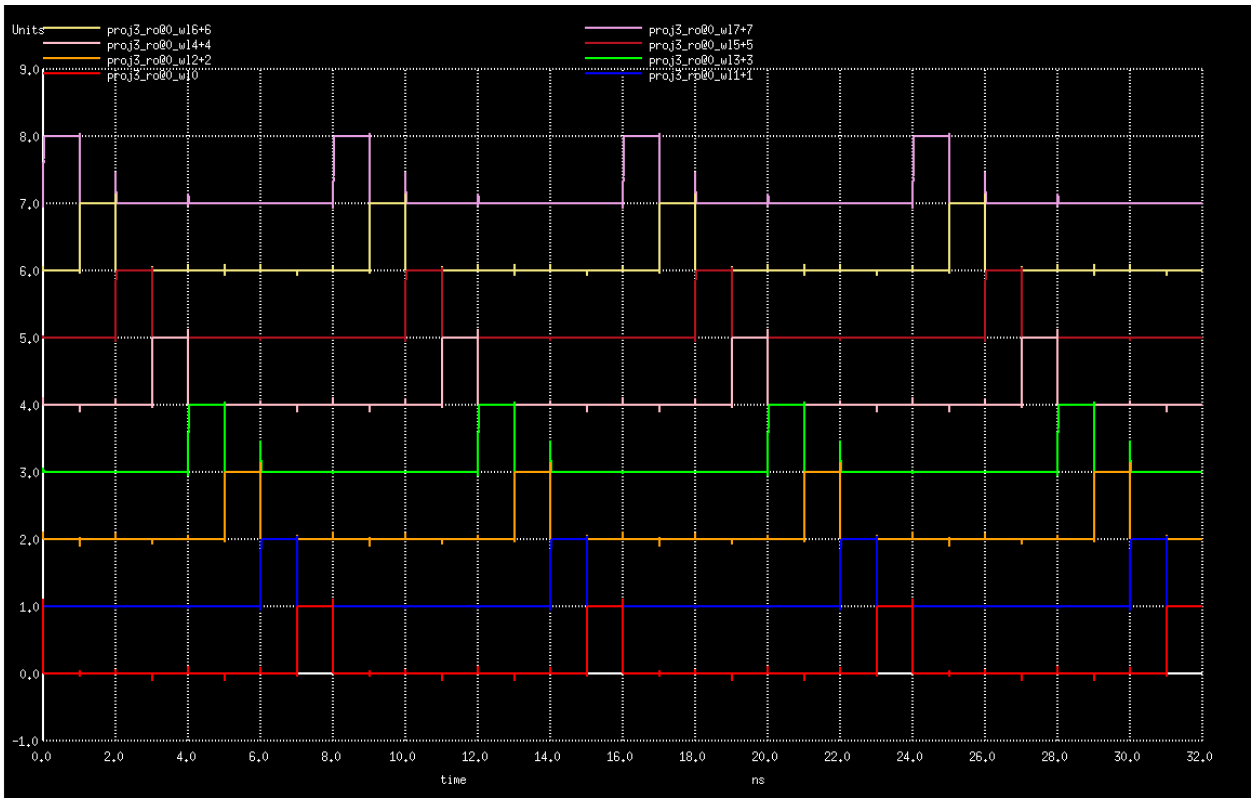


The top line tracks the inside of the RAM cell, for debugging. As we see from data line, there are no read upsets. NB in bottom picture, write and read on colDriver should be flipped.

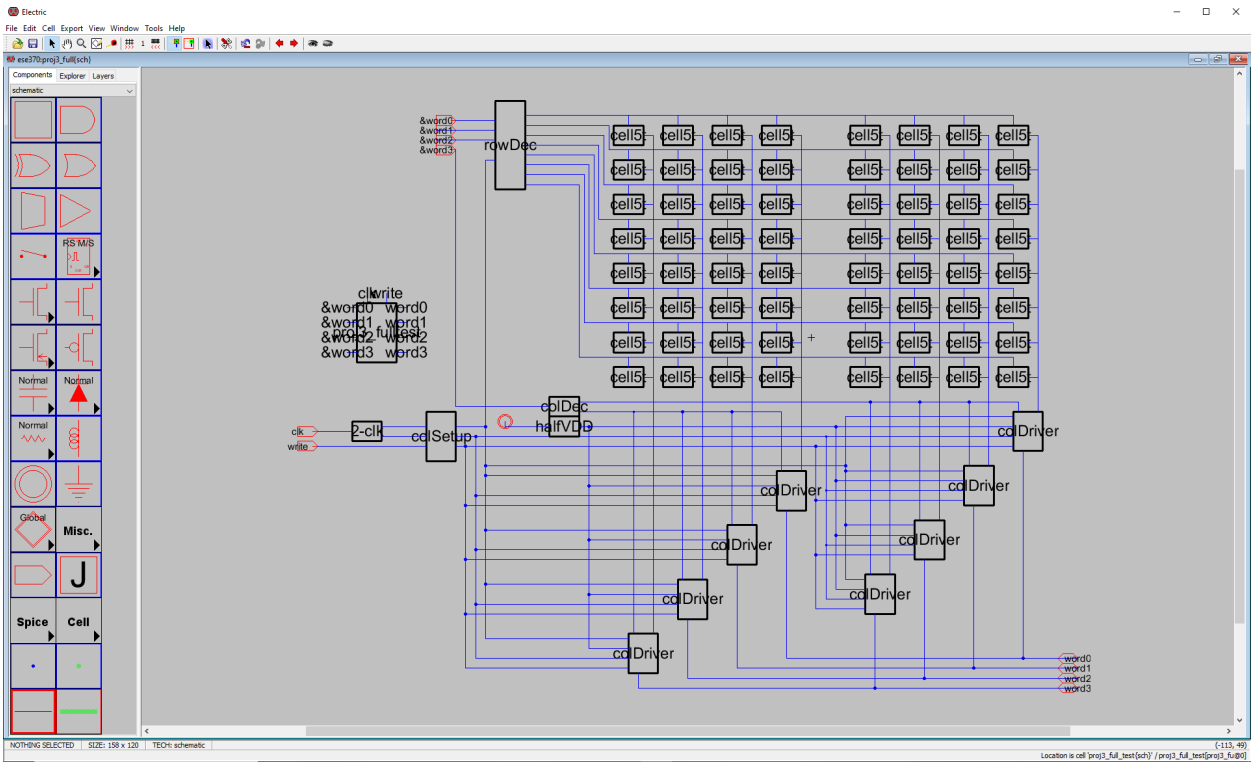
Full design write timing:
Cannot write faster than it takes to charge/discharge the bitline, which only happens when clk2 is active so if the time to charge/discharge is t, it takes period of 2t and this is the clock frequency. I will verify this by increasing the base clock speed until there are writing errors.

0	8
1	9
2	A
3	B
4	C
5	D
6	E
7	F

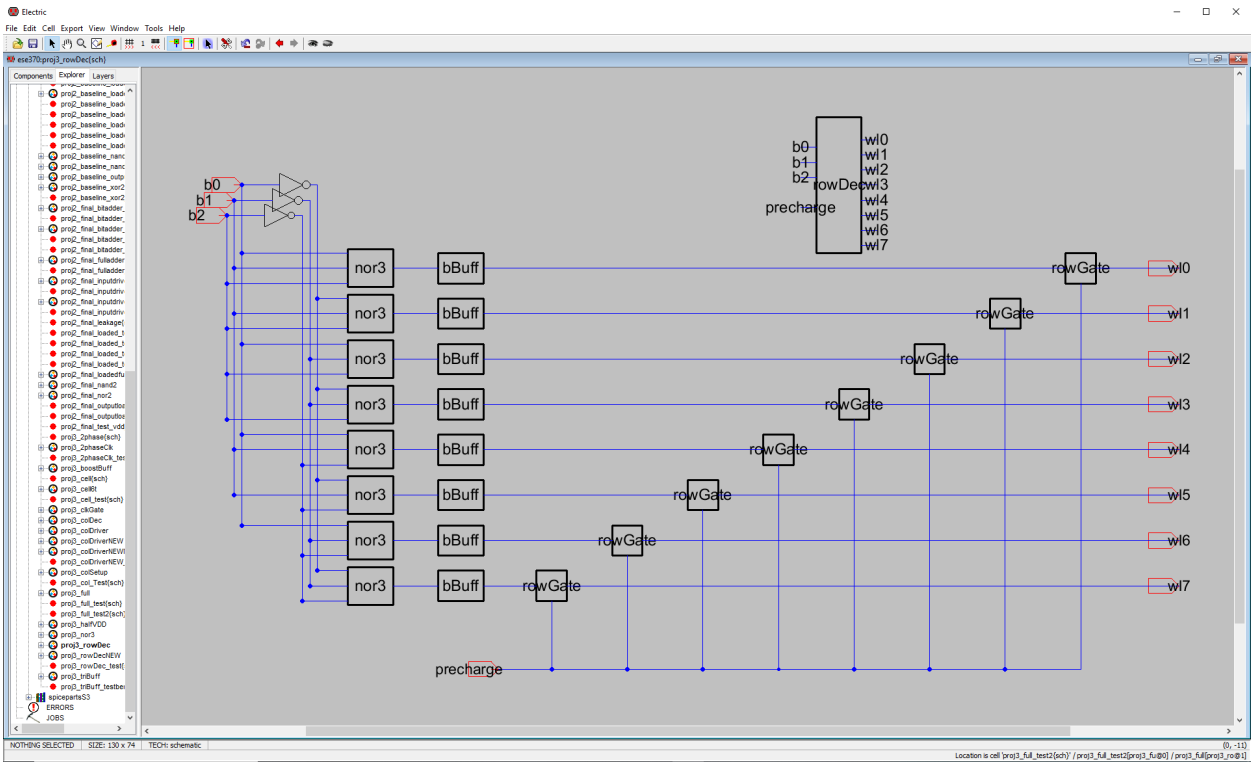
Ideas: 4 bit word, precharge bit lines on inactive word column to prevent charge sharing from overwriting data



Row decoder working; needs to be gated on read precharge

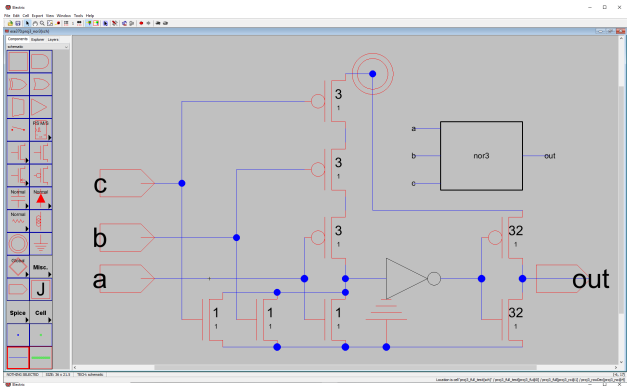


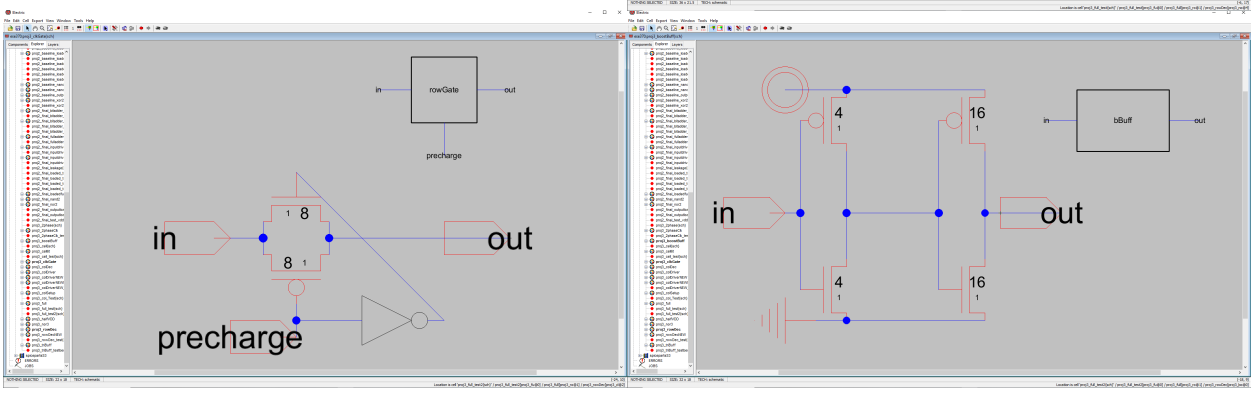
This is my overall memory cell structure.



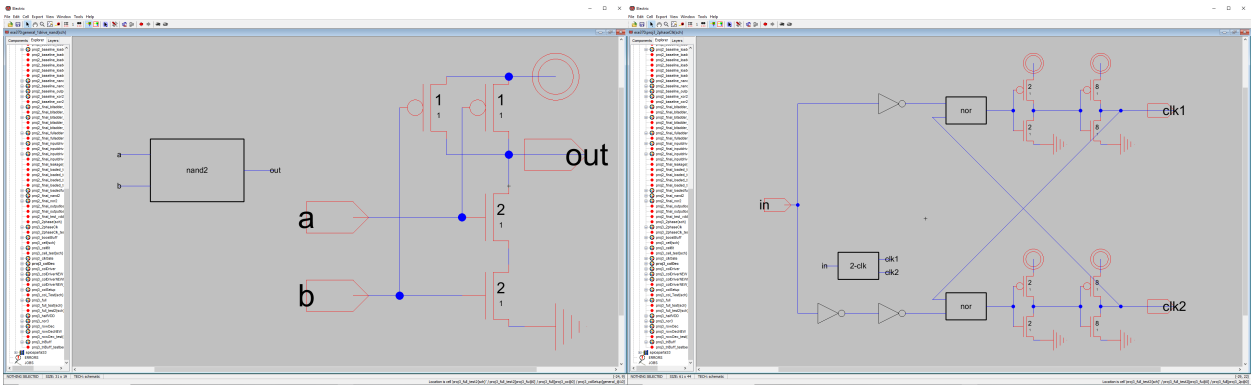
The MSB of the memory location is used to determine which column of words to select through the column decoder, while the 3 other bits go through the row decoder to activate Word Lines.

Additionally, since the word lines must not be active during precharging, that signal is also used to gate all the word lines in the row decoder.

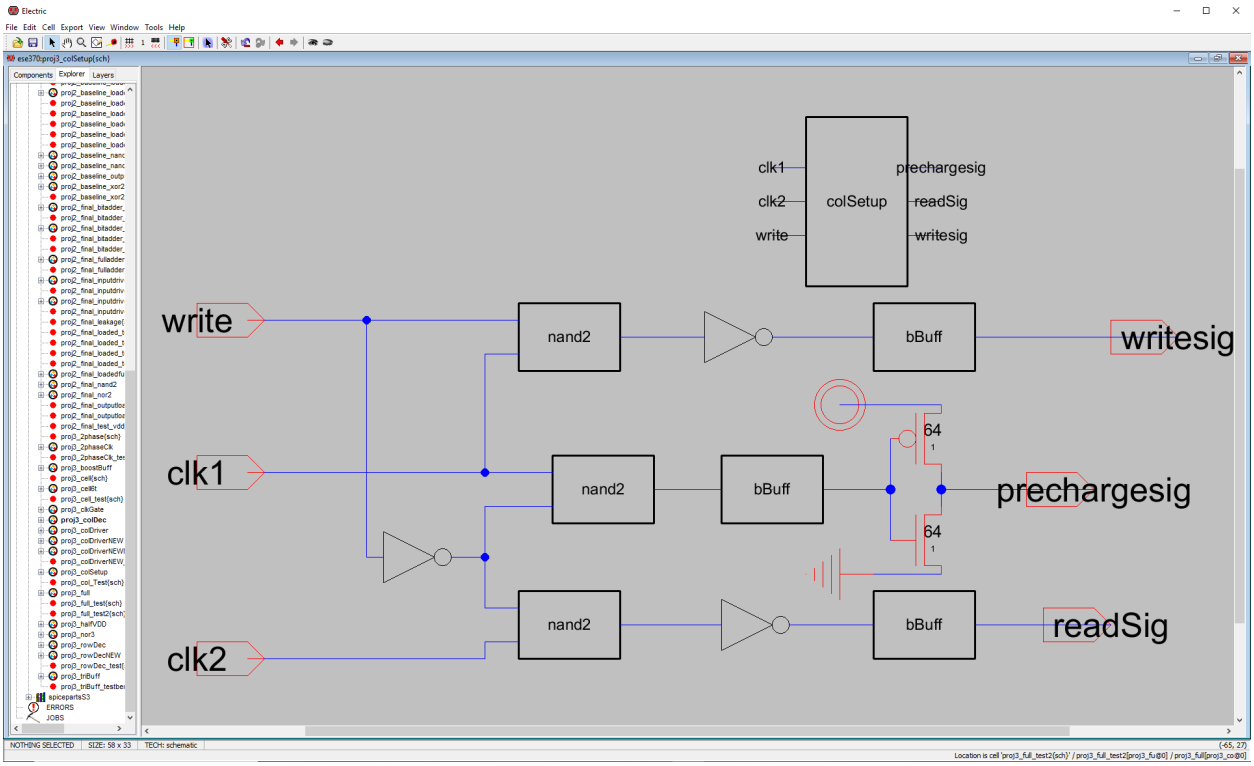




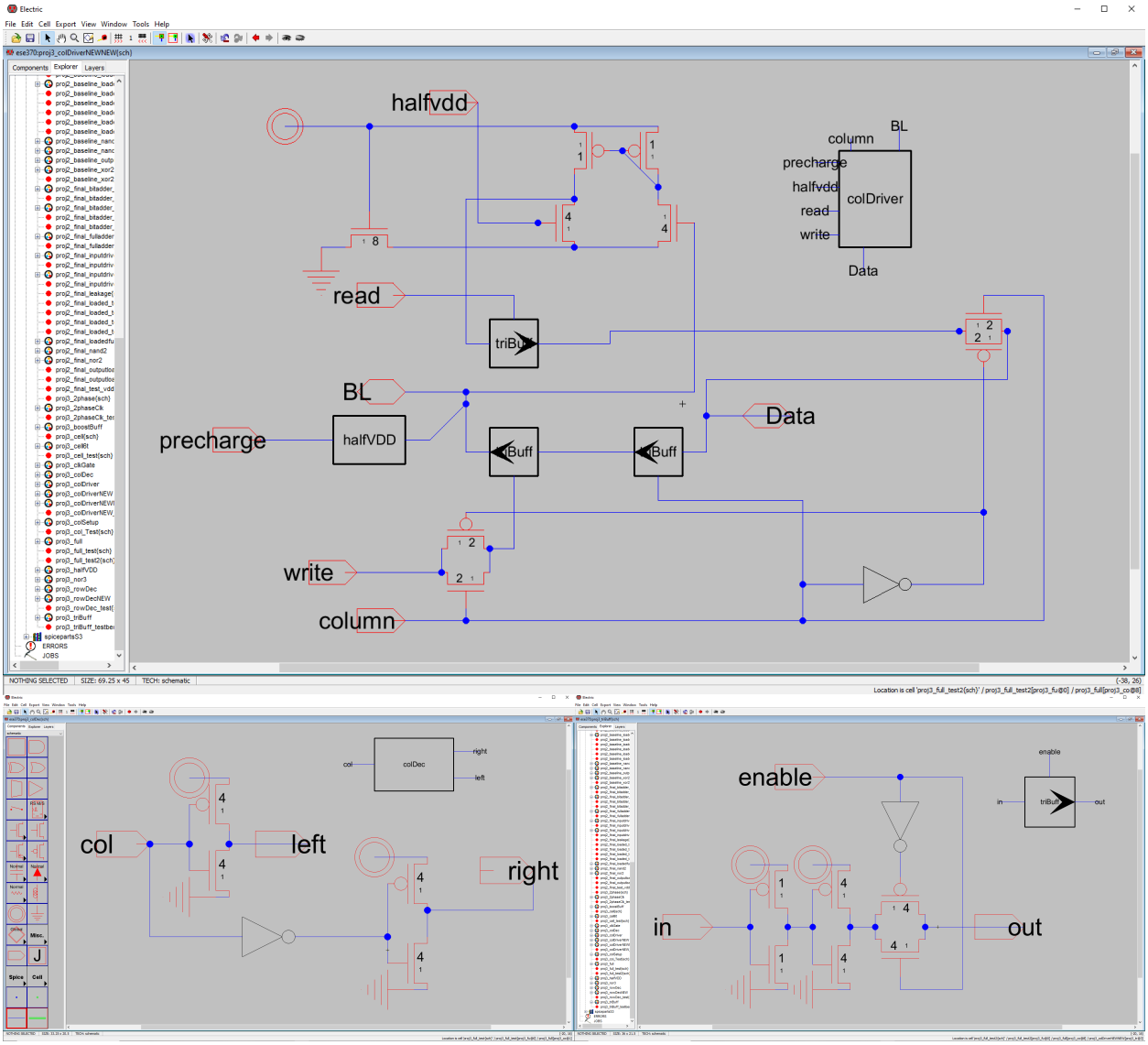
Since my row gates are transmission gates and there are 8 access transistors on the other side to power, I use 2 sequential sized inverters to boost the drive.



The clock signal is fed into the 2phase clock, and both of those clock signals and the write/! read signal go through the column setup circuitry that translates those signals into precharge, write, and read signals.

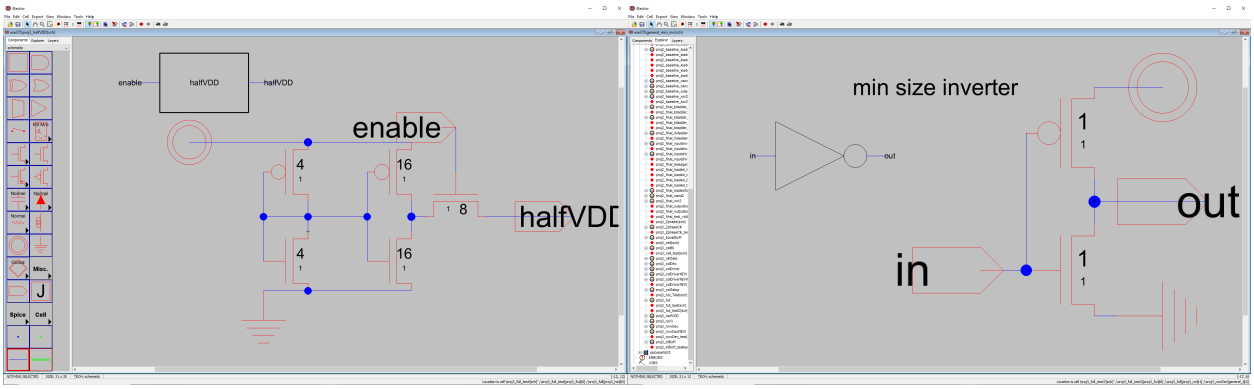


Since precharge had 208C_0 of capacitance and the read and write signals had at least 64C_0 (write also has additional elmore delay), a min sized inverter does not charge the row decoder gates fast enough, so I used the current boosting buffer and a huge inverter to get more oomph from the signal. Unfortunately, after this change, ngspice refuses to converge :(. The signal conversions work on the same principle from the milestone, just neater. Also, since these signals are shared between the column drivers, there is no need for each to get their own.

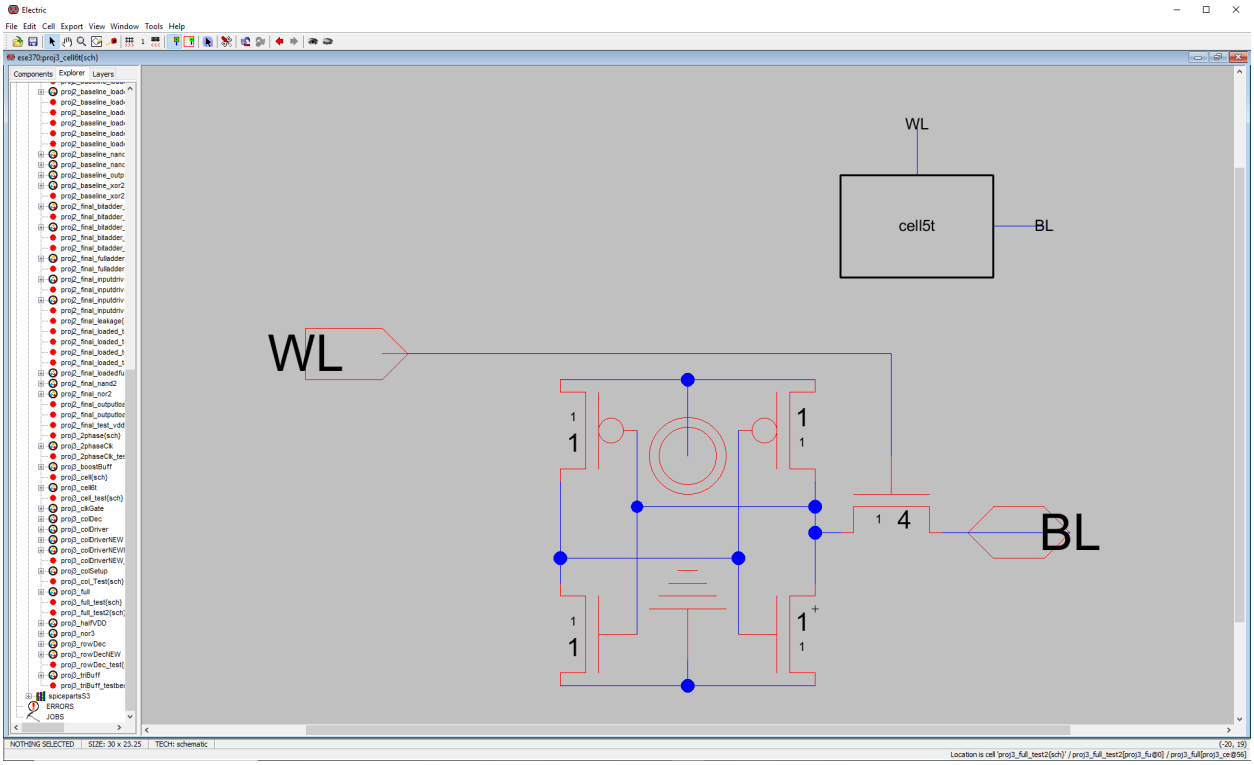


The column decoder selects either the left column of words or the right column of words, and the column activate signal is fed to all the column drivers to ensure that the wrong column does not activate. It is used to gate several gates and a buffer. The rightmost gate is because a tristate buffer would be able to send through floating node voltages to the data line even when read is off, the leftmost gate is because of a similar reason but with the bitline and the write signal. The tristate buffer ensures proper staging so the value on a write is the data and not a floating voltage.

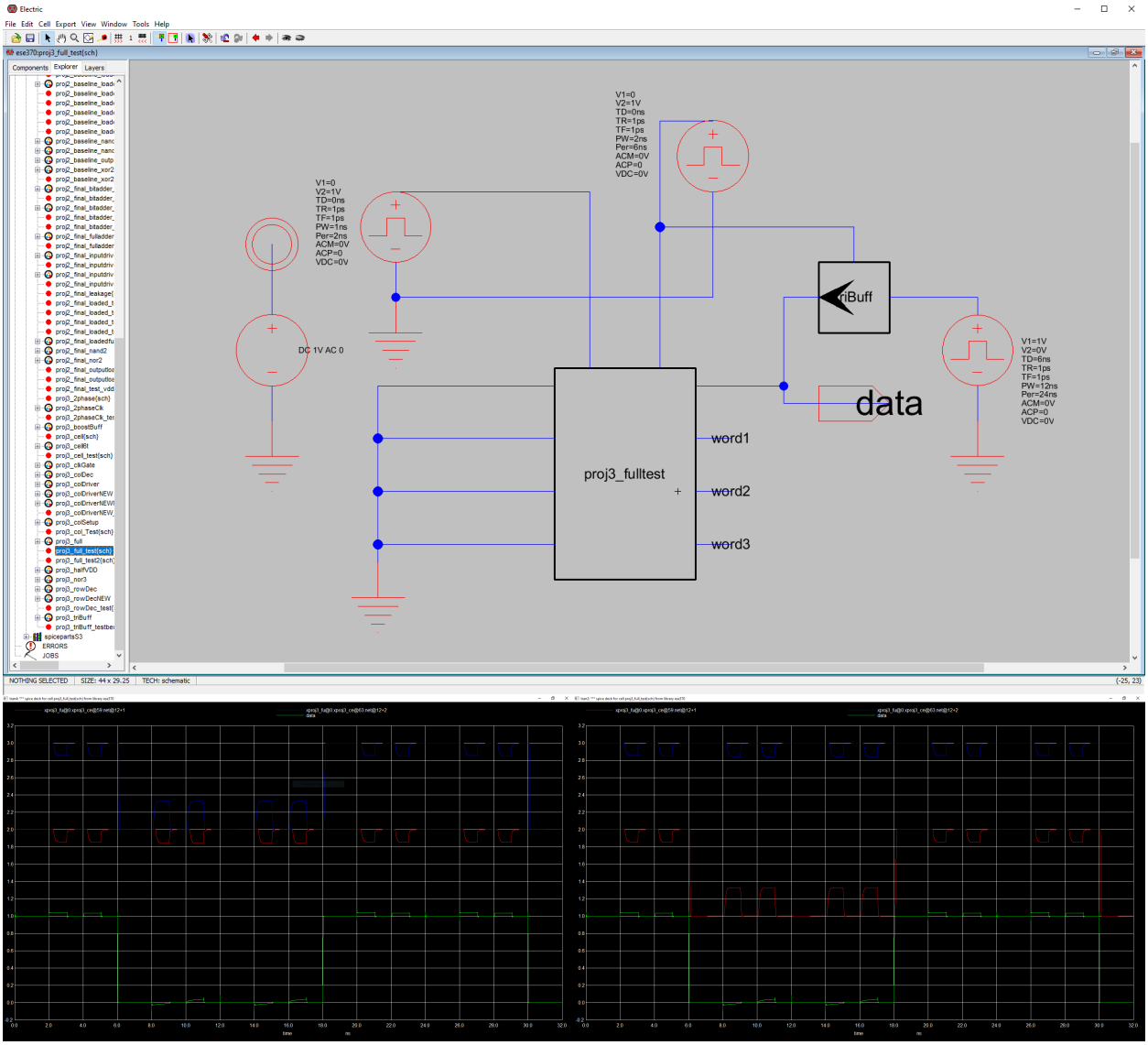
The precharge enables the driver's halfVDD to precharge the bit line, the write drives the data onto the bit line, and the read signal allows the output of the diffamp to be written onto the data line, but only when the column is active.



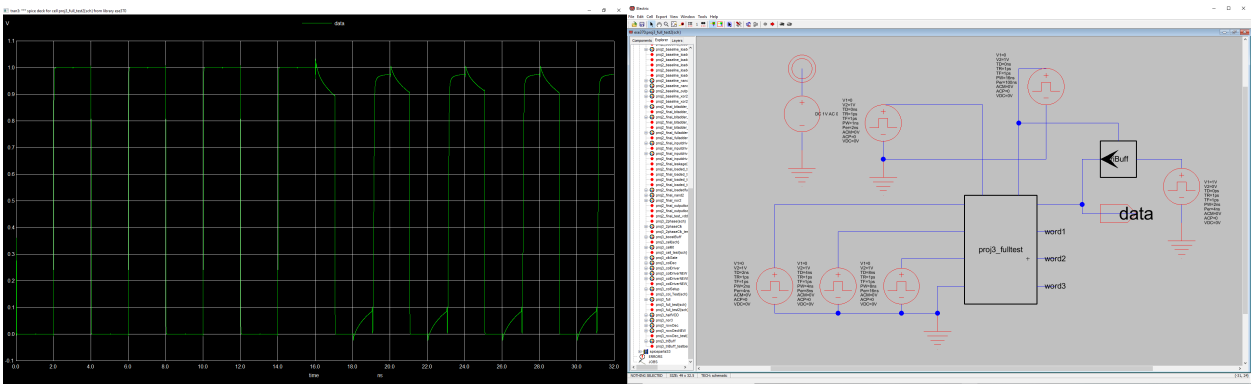
The halfVDD device is the reference design, and the minimum inverter is a minimum sized CMOS inverter. As the former is also used by the diffamp, it is enabled.



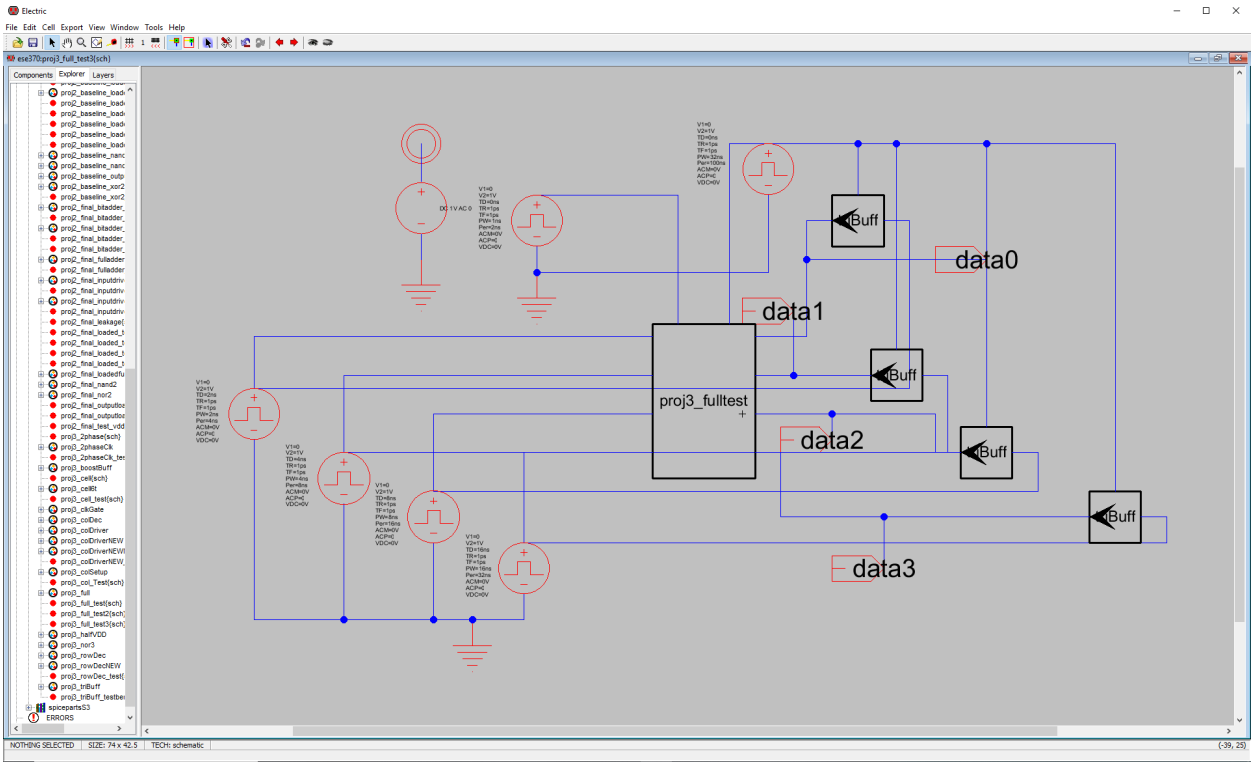
My hand calculations from the milestone should show that the access transistor needs to be larger than width 4, and when testing the single cell, I found this to be true. However, when playing around with the parameters with the full memory, I found that even at width 4, the transistor still functioned properly. In fact, the smaller transistor meant that on reads, the value stored in the cell did not dip as far towards halfVDD.



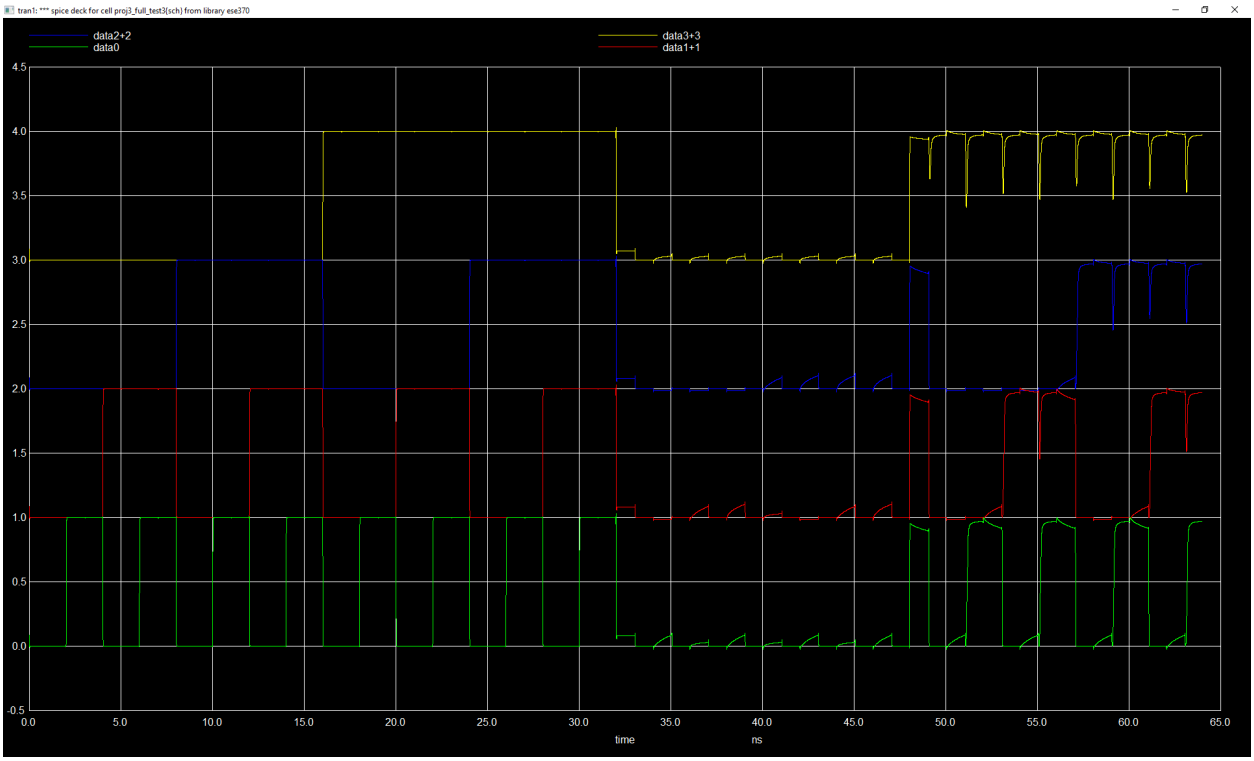
To test the column decoder/selector, I performed the single bitcell verification sequence but this time measured the output data and also the cell storage itself, to ensure that the other corresponding cell in the row was not written to.



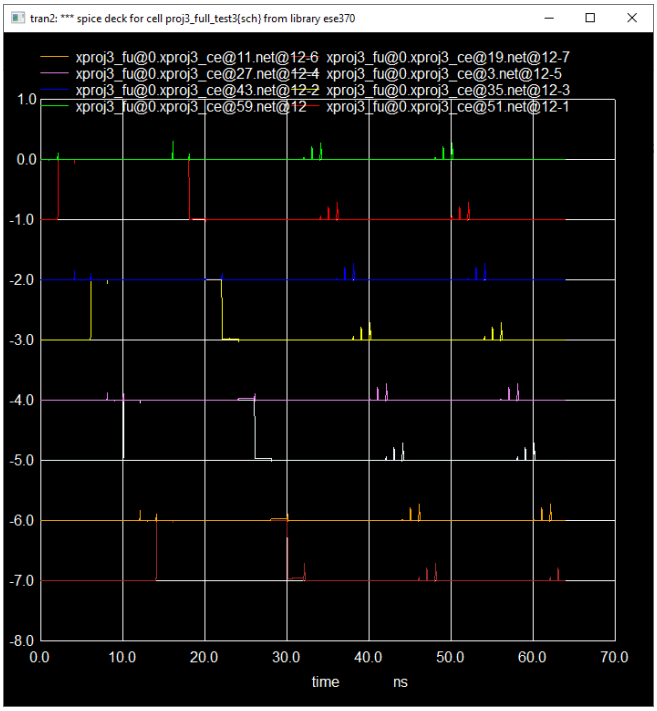
This shows the row selector working; I wrote values to every cell in a column and then read them back.



Here, to verify operation, into each word I write the word number and then read it.



Unfortunately, due to the shared word lines, the word that is not currently selected leaks out all its charge through the access transistor. I should have gone 16x4 rather than 8x8. It's too late now.

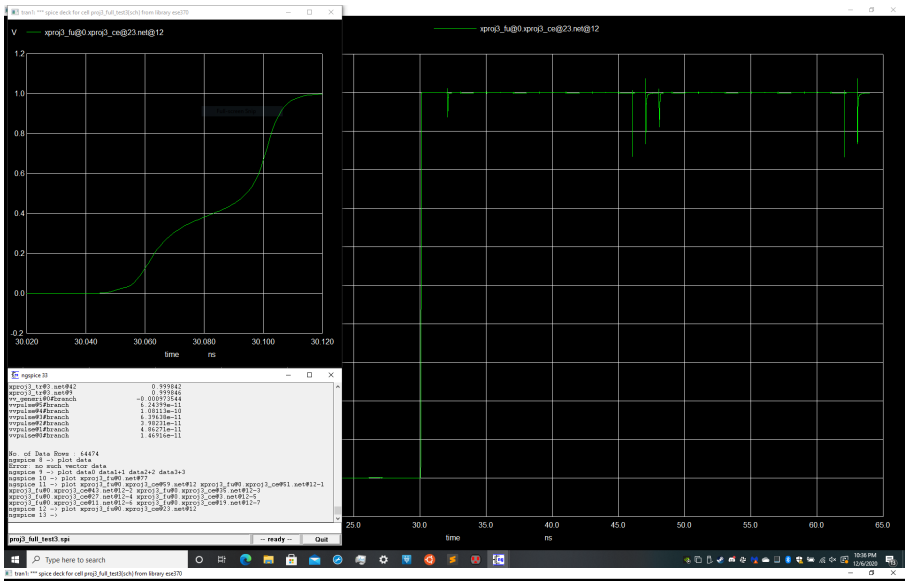


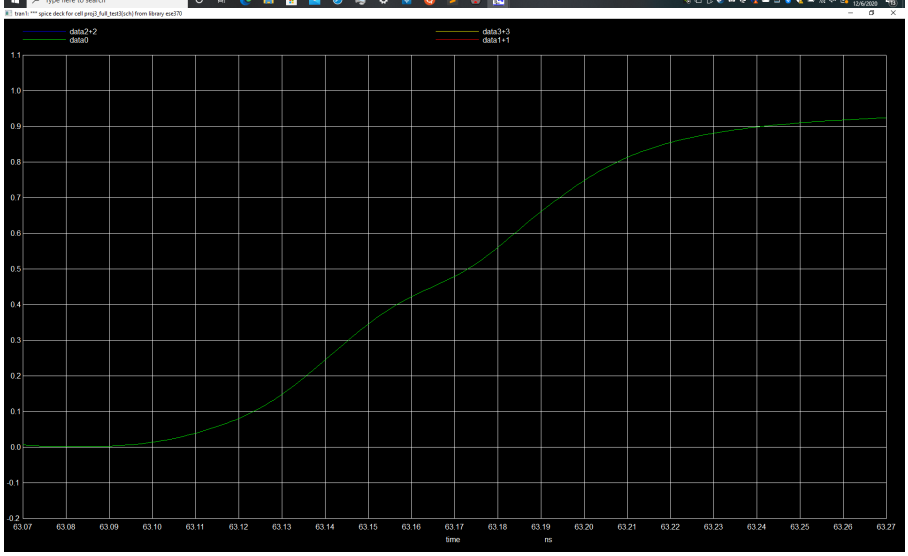
Critical Path: Write
Write signal flips on, in column setup, clock 1 and write signal then gets fed into the boost buffer before being sent to all the drivers. There, it is gated by the column signal before it writes to the bit line.
Critical Path: Precharge
Write signal turns off, clock 1 goes through nand, boost buffer, big inverter, simulatenously turn off all the row gates while enabling the precharging of the bit lines
Critical Path: Read
Write turns off and clock 2 goes through boost buffer and turns on the tristate buffer which passes the diffamp output through a transmission gate to the data line.

Worst case through row decoder row 7, column right.
From Milestone, write 1 to 0 is hard, reading should be symmetric.

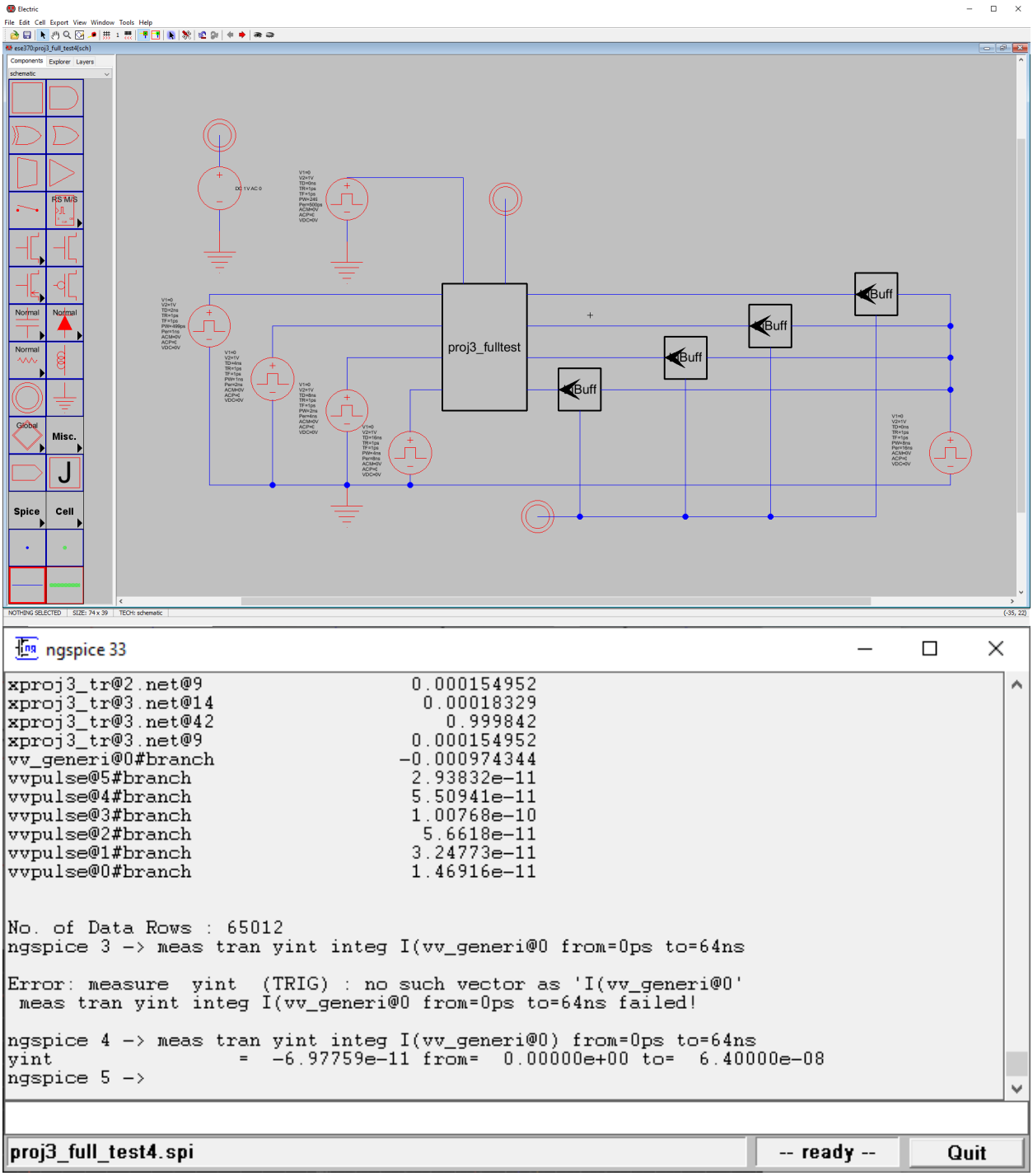
In the graph above, zooming into the lower right green is row 7, column right. It hits .5V at 63.173ns, leading to a delay of 173ps.

Writing to that same cell: at 30ns, it gets written from 0 to 1, it happens at around 30.095ns so a delay of approximately 95ps. This delay only accounts for 50% though so to get a little margin of error, say period of 2x250ps. This means 2 counts in 1 ns and 1 billion ns in 1s so 2gigahertz.





16words*.5ns = 8ns to write everything
16ns for a cycle of 1s and 0s, 64ns for 4 rounds.



6.978e-11 Coulombs * 1V / 64ns = 1.09 miliWatts

Design Metrics	
Memory Cell Area (units of min width)	8
Delay (units of seconds)	1.73E-10
Power (units of watts)	1.09E-03
Figure of Merit	0.000000000000000000000156588528

Additional timing considerations: The row gates must activate before the boost buffer switches to on.

Despite trying resizing transistors, always keeping bitlines precharged when not reading or writing, and everything else I can think of, after solving the above timing constraint, I could not think of a way to overcome the fact that the word line is shared by 2 different words and this causes shenanigans. I need to size the access transistors such that it can charge and discharge the value stored in the cell, but this means that when the word lines activate, the value stored will be forced to change. Thus it was the height of folly to try an 8x8 design with a 5t cell, and I should have used 16x4 instead.

My disappointment is immeasurable, and my day is ruined.