ECEN 351 Final Project: Register File

Scott Madeux

Lab Partners: Brett Sinden and Patrick Hopman

4/11/2019

INTRODUCTION

This project involved designing and laying out a 1 bit register file with 8 memory locations. The three major parts of this project were to design a 3 to 8 decoder, a block of a 1-bit registers, and an 8 to 1 multiplexer. Below is a rough outline of our circuit.

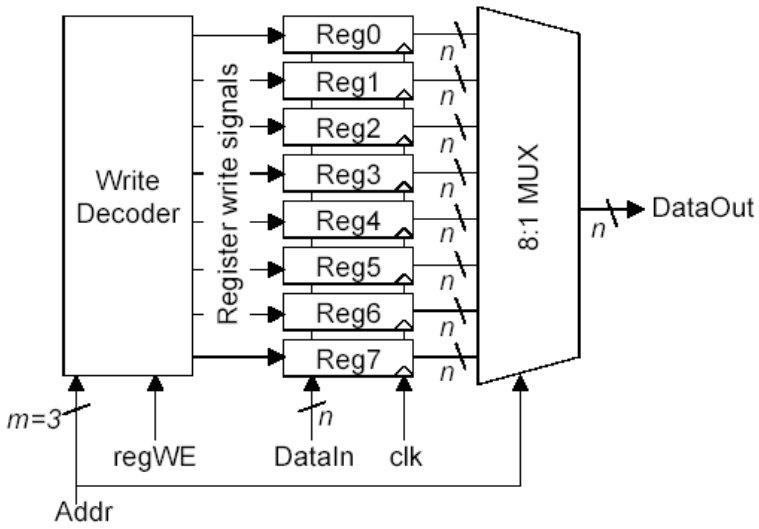

Figure 1: 1-bit register file with 8 memory locations

Patrick, Brett, and I each took one of the three major parts of the circuit mentioned about and created a transistor level circuit in LTspice. After we had that we each created a layout of that circuit in glade using the inverters, MUXes, etc. in standard cell library provided to us. The last step was to take the parts we had created individually and connect them together to form our register file.

SIMULATION IN LTSPICE

We chose to do our simulation of the register file in LTspice. Here is what our different parts of the circuit looked like and how we tested the whole thing out:

3 TO 8 DECODER

Brett created the schematic for the 3 to 8 decoder that takes in a 3-bit binary value and activates the correct register that should have data loaded into it.

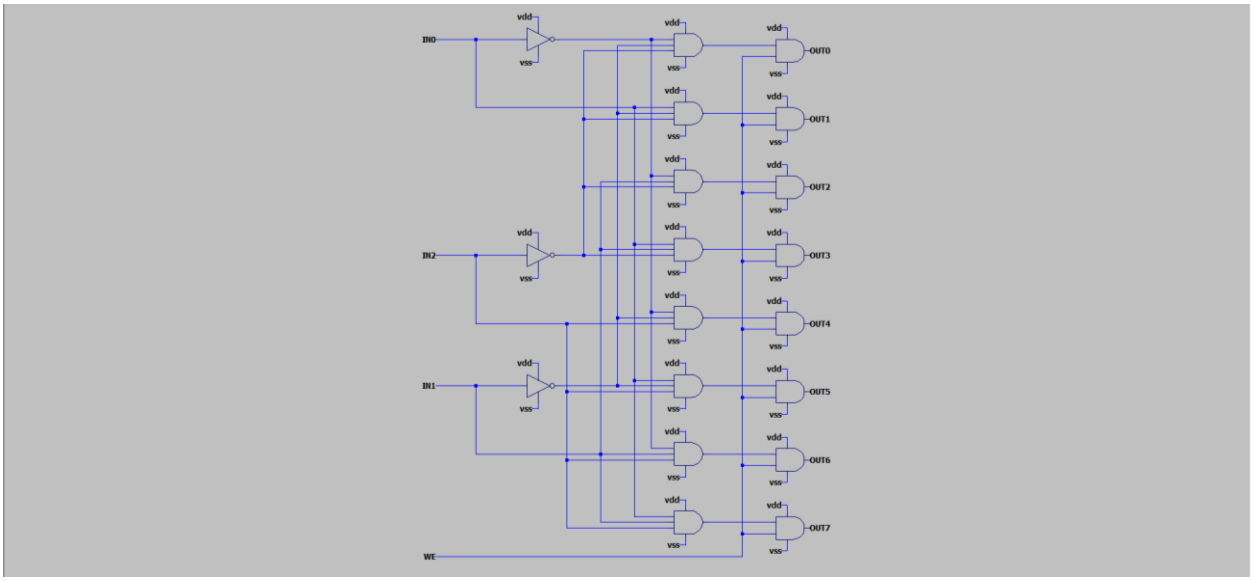

Figure 2: 3 to 8 Decoder Schematic

REGISTER BLOCK

I designed the block of 8 1-bit registers. In order to do this, I had to create a 1-bit register by connecting a 2 to 1 MUX to a flip-flop as pictured below.

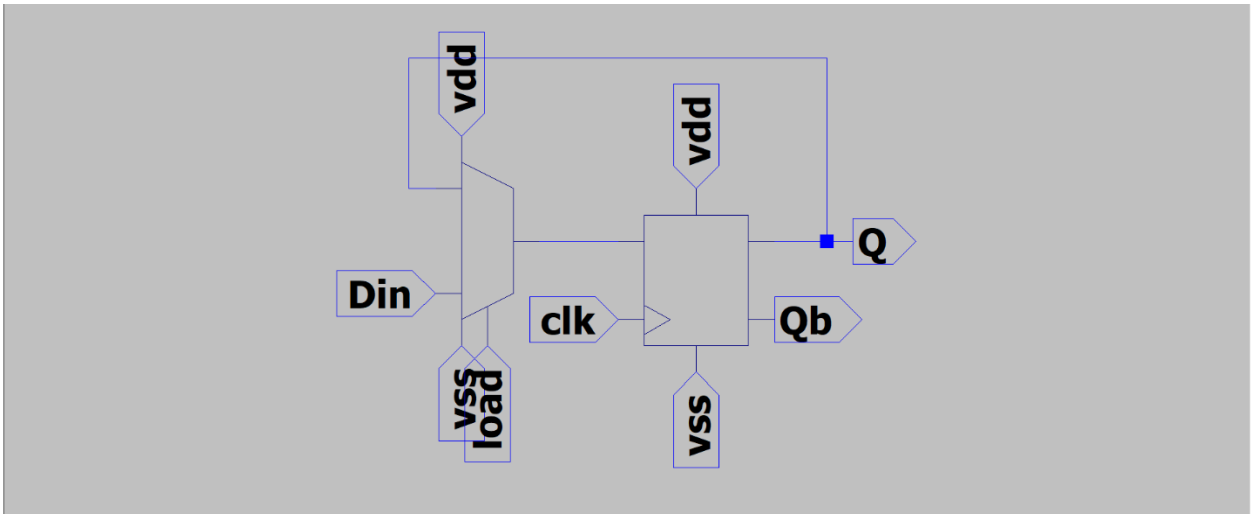

Figure 3: 1-bit register using a 2 to 1 MUX and a flip-flop

Next, I put 8 of those 1-bit registers together connecting all of the Din inputs to the same input and having separate load inputs. I also added buffers for the clock and data input signals. Since buffers I used for this were going to have to drive all 8 registers, I made them 2 times as large as my smallest inverter.

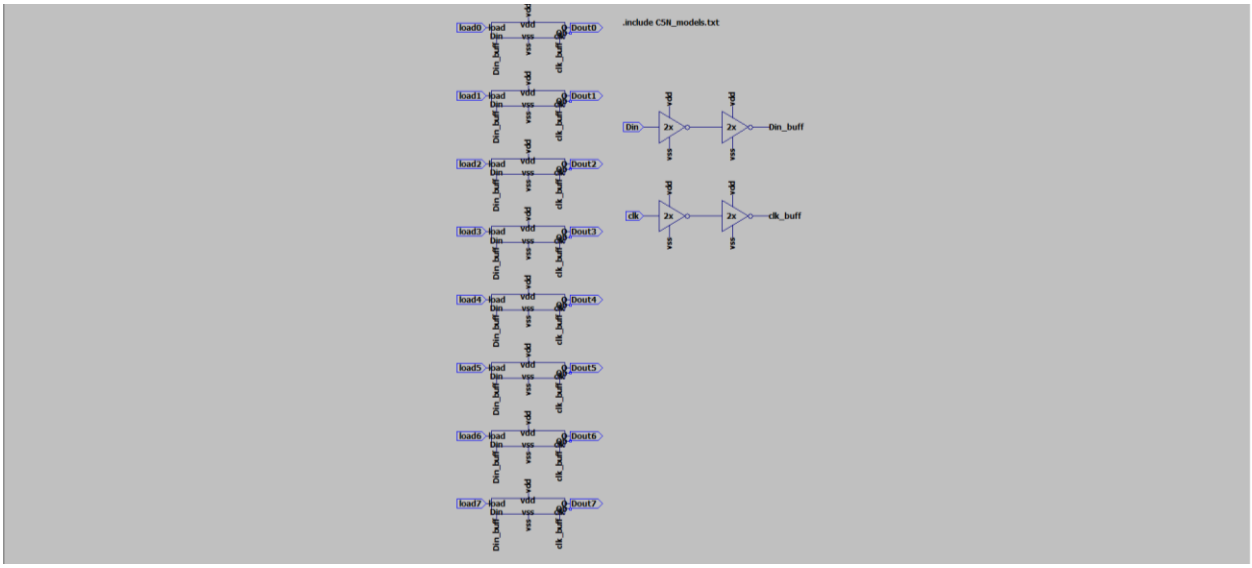

Figure 4: 8 1-bit registers with clk and Din input signals buffered

8 TO 1 MULTIPLEXER

Patrick created an 8 to 1 multiplexer by first creating both a 4 to 1 and 2 to 1 from transmission gates and inverters, then combining the two as pictured below. It's difficult to see on the layout, but the sel0 and sel1 inputs are going into both 4 to 1 multiplexers.

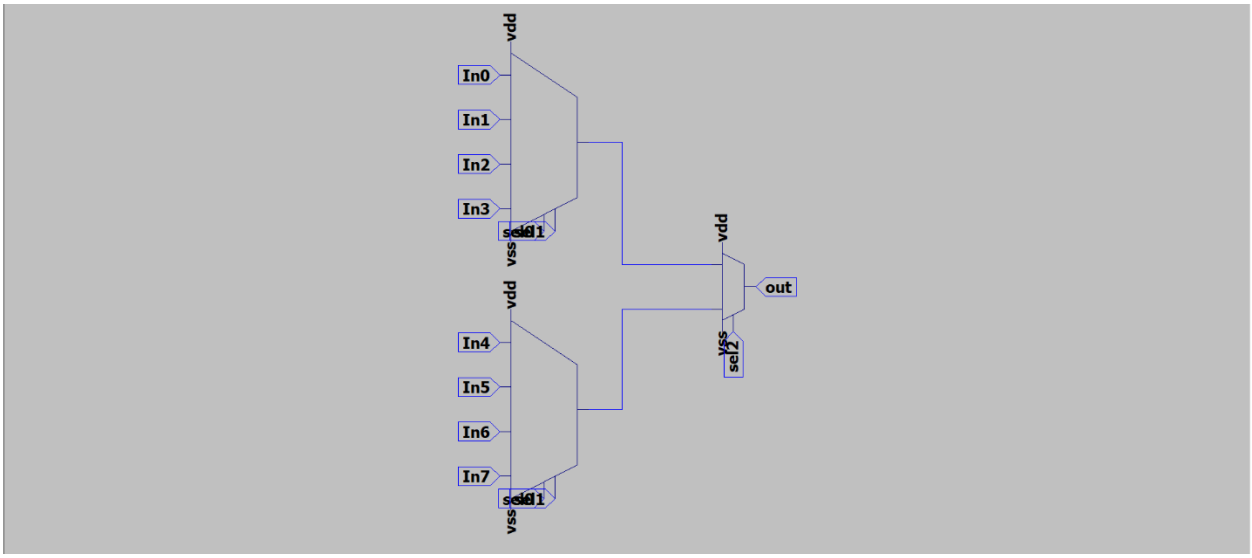

Figure 5: 8 to 1 MUX using two 4 to 1 MUXes and a 2 to 1 MUX

REGISTER FILE

Here is the LTspice schematic of the entire register file:

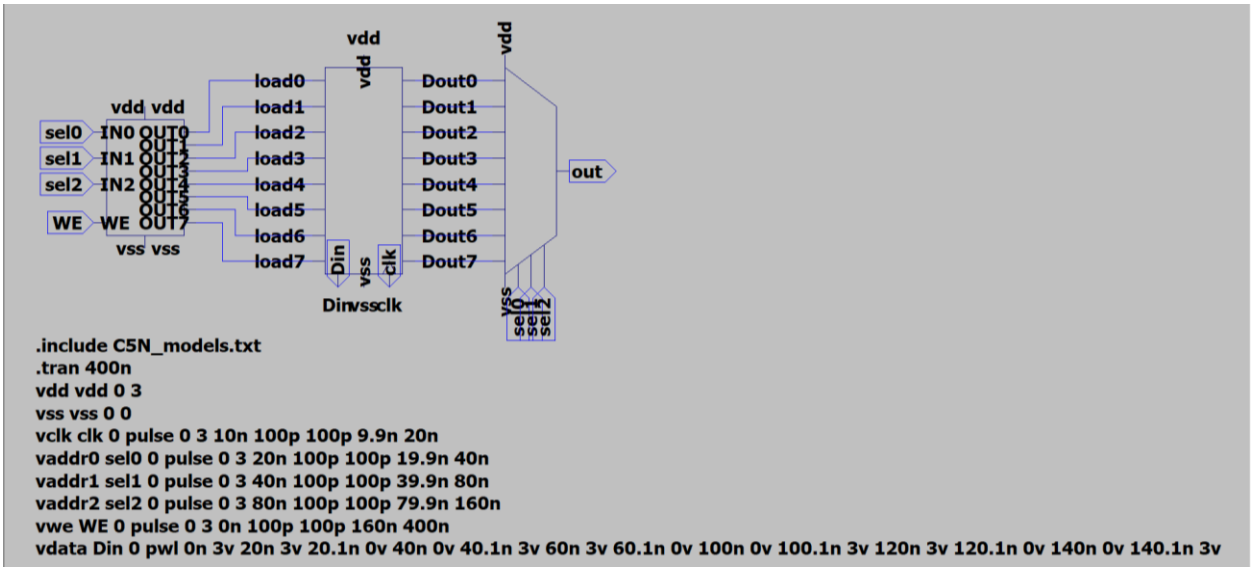

Figure 6: Register file schematic with LTspice directives for testing

Now comes the hard part of figuring out if everything actually works. To do this, I ran wrote either a 1 or a zero to each address in the register file and then turned off the write enable and stepped through each address again to make sure that the output matched what I had just loaded into the registers. On the screenshot below, the green line is the address that is being selected. In order to better visualize the 3 address lines, I used this equation in LTspice, which gave me the actual address number that was being selected:

$$Address = \left(\frac{V(sel0)}{3} \times 1\right) + \left(\frac{V(sel1)}{3} \times 2\right) + \left(\frac{V(sel2)}{3} \times 4\right)$$

This equation converts the high and low signals of the three address lines into a base 10 number. I made a composite of two images to make things easier to see. The red line is the data being loaded into the registers and the blue line is the data being read out. 3 volts is a high signal and 0 volts is a low.

Figure 7: LTspice waveform of register file test

Register #	Data
0	1
1	0
2	1
3	0
4	0
5	1
6	0
7	1

Table 1: Data loaded into each register for test

Table 1 above shows the data that was loaded into the registers for the test. We can clearly see from the blue output line in Figure 7 that we are getting the same values we loaded in when we step through the addresses.

LAYOUT IN GLADE

3 TO 8 DECODER

Brett created the layout for the 3 to 8 decoder in Glade. I've included a screenshot of the entire thing plus a close up shot.

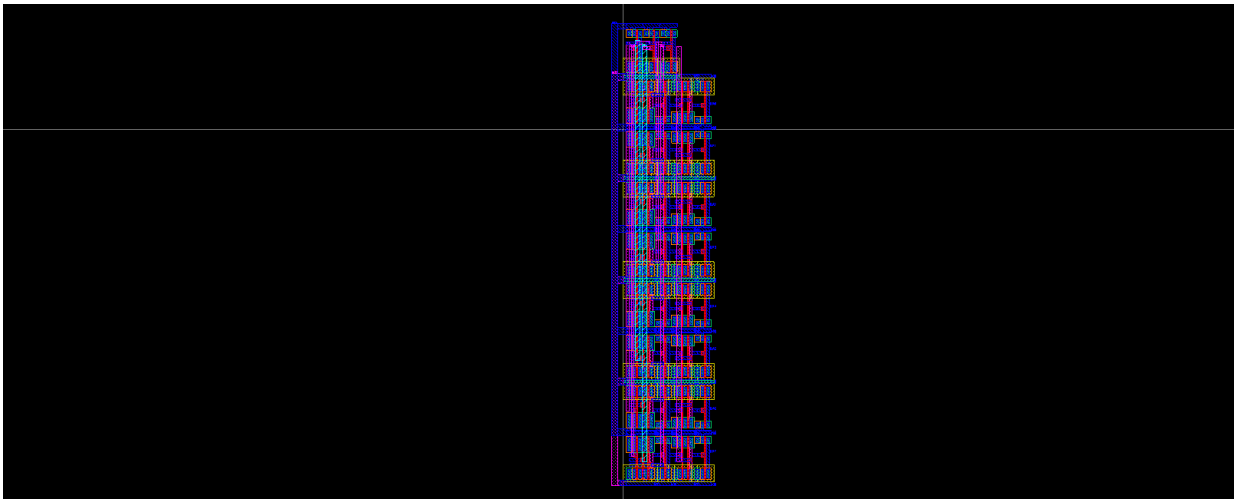

Figure 8: 3 to 8 decoder layout in Glade

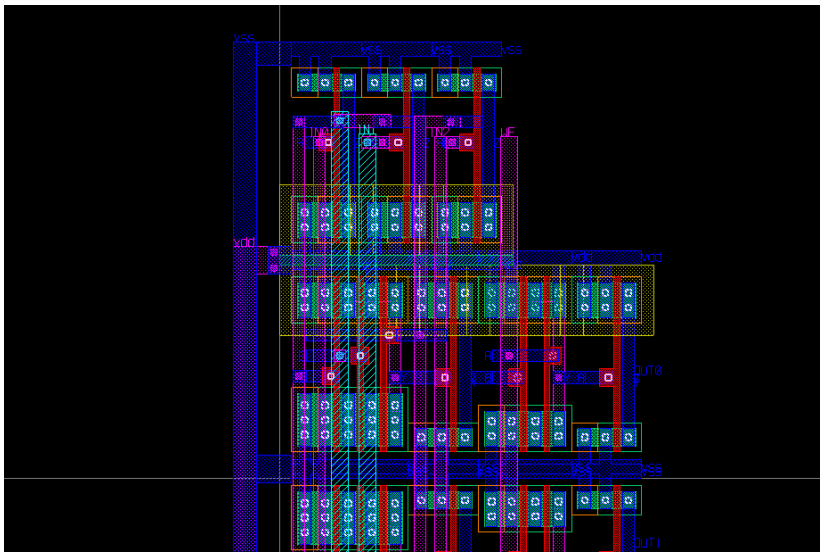

Figure 9: Closeup of the 3 to 8 decoder layout

REGISTER BLOCK

I created the layout for the block of 8 registers used in the register file. Below is the layout for one of those registers.

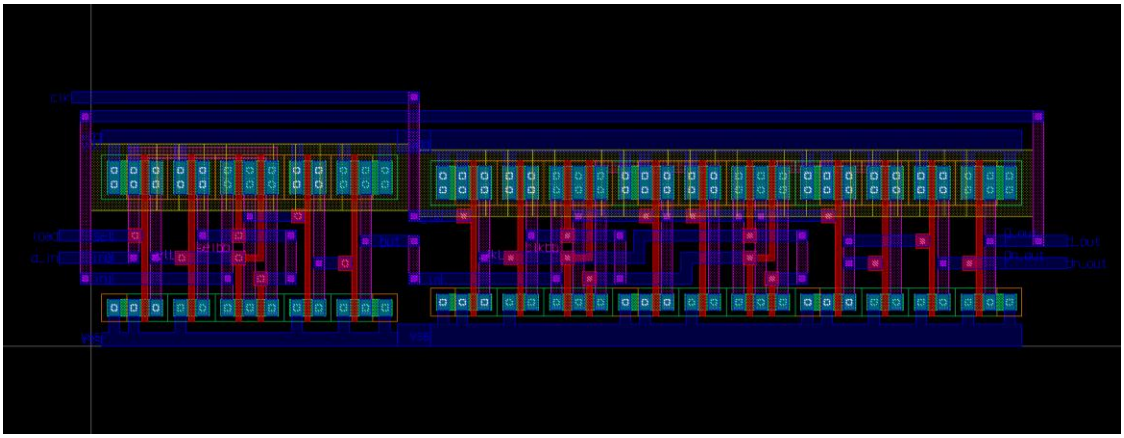

Figure 10: 1-bit register file layout in Glade

This layout was pretty simple to create I used a 2 to 1 MUX along with a flip-flop and connected them the same way they were connected in my LTspice schematic shown in Figure 3. The only difficulty I did have was that the labels for the data inputs on the MUX were switched so I had to connect “In1” to the output of the flip-flop instead of “In0” as shown in the LTspice schematic.

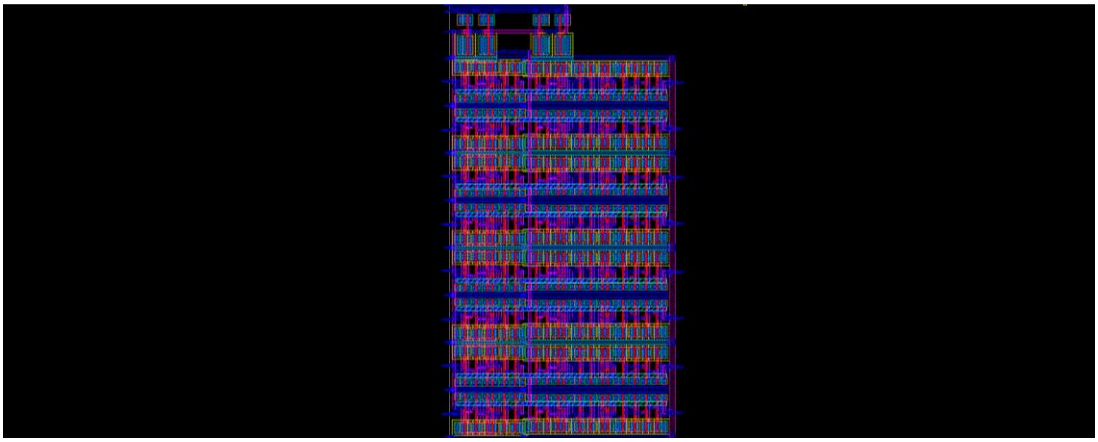

Figure 11: Block of 8 1-bit registers with buffered clock and data lines

It’s very difficult to see anything in this screenshot of the entire block of registers so the screenshot below is zoomed in to show the clock and data line buffer as well as one of the registers.

The 3 to 8 decoder is both DRC and LVS clean.

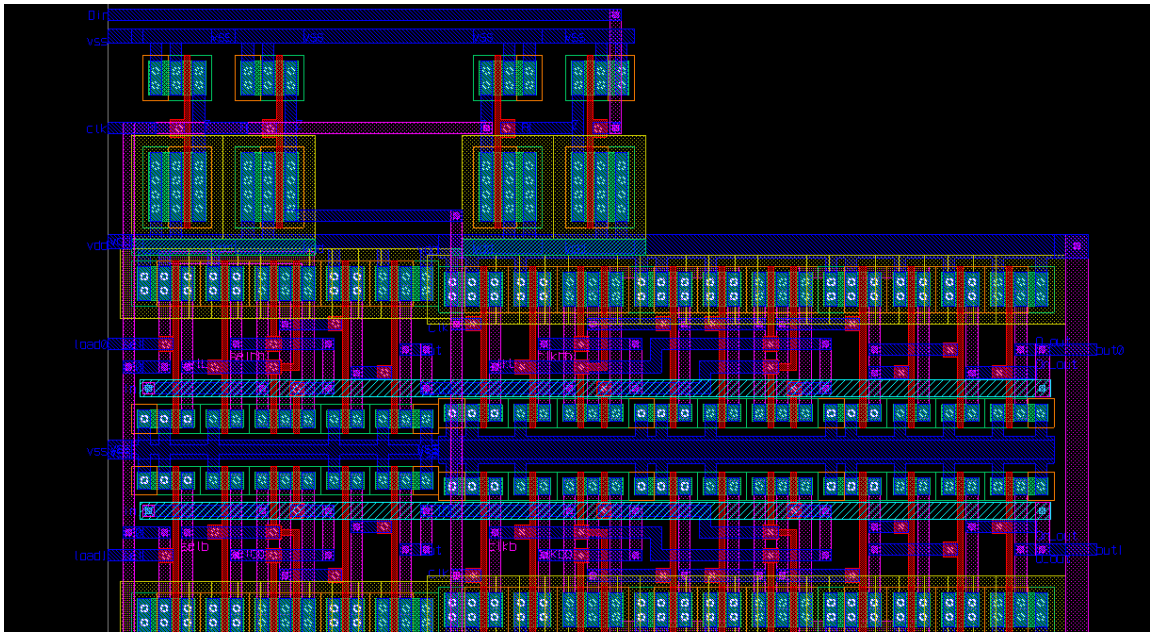

Figure 12: Closeup of the register block layout

The two inverters on the top left form the buffer for the data line and the one on the right forms the buffer for the clock. The register beneath them is the register corresponding to address 0.

The register block is both DRC and LVS clean.

8 TO 1 MULTIPLEXER

Here is the 8 to 1 MUX layout that Patrick created using two 4 to 1 MUXes and a 2 to 1 MUX.

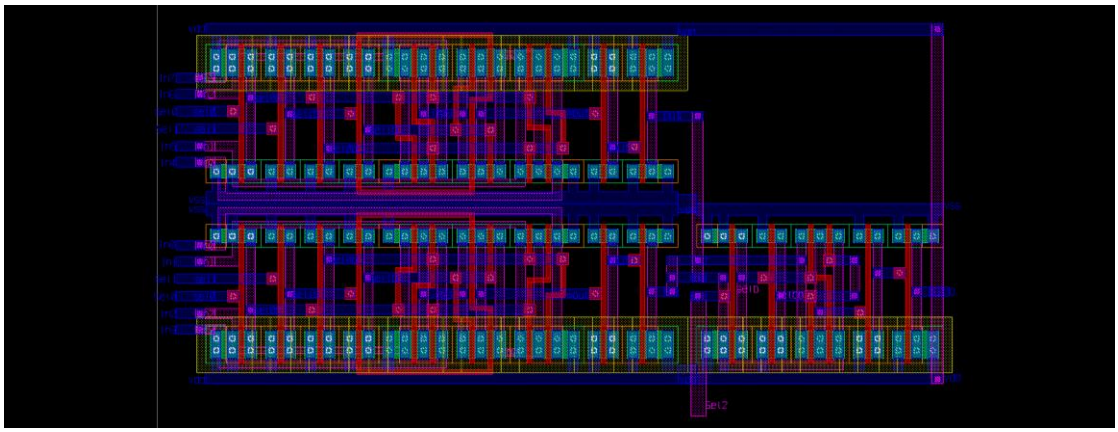

Figure 13: Glade schematic of 8 to 1 multiplexer

The 8 to 1 multiplexer is both DRC and LVS clean.

REGISTER FILE

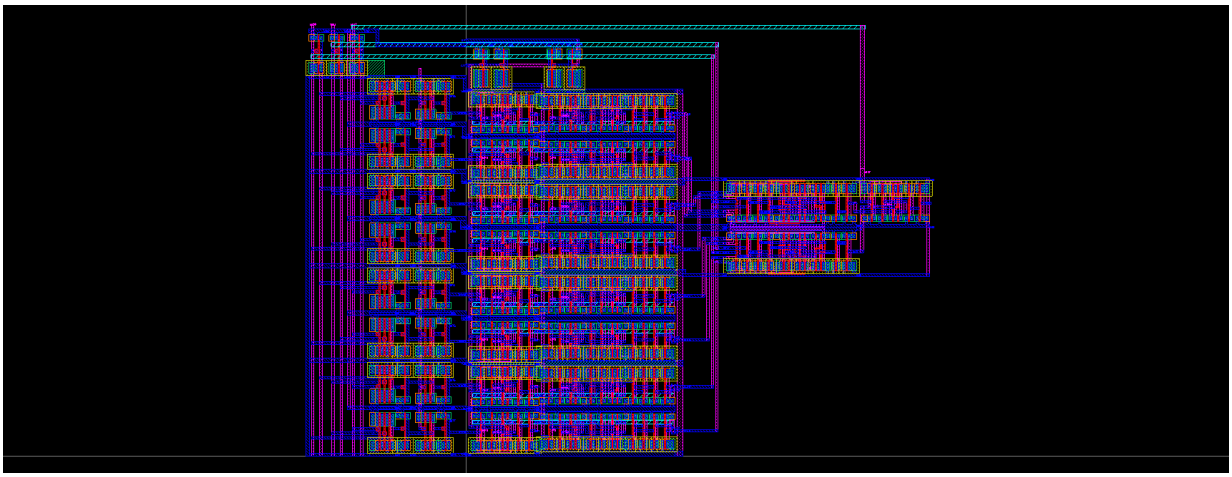

Figure 14: Glade register file layout

Above is the layout for the entire register file. The decoder used in this screenshot is slightly different than the one Brett made. We were having some troubles with it and I ended up making a new version of the decoder. The register file had no DRC errors. I was able to get a clean LVS with only the decoder and the register block connected to each other, but I still got a few errors when I added the 8 to 1 MUX. These errors could be worked out with some more time. Here is a screenshot of part of the LVS error file.

```
register_file.lvs
38
39  ---------------------------------------------------------------------------
40    Netlist errors : register_file_extracted.cdl
41  ---------------------------------------------------------------------------
42  1 NETS do not match:
43  NET "n2" 20 connections
44
45  2 DEVICES do not match:
46  DEVICE P: (inst MM935) [g] n135 :: [s,d,sub] n2, n139, n4
47  DEVICE N: (inst MM698) [g] n17 :: [s,d,sub] n139, n2, n14
48
49  ---------------------------------------------------------------------------
50    Netlist errors : register_file_top.net_flat
51  ---------------------------------------------------------------------------
52  2 NETS do not match:
53  NET "sel0" 14 connections
54  NET "N013" 6 connections
55    P: (inst M20/XX2/XX14/XX2) [g] /XX2/NC_06 :: [s,d,sub] N013, vdd, vdd
56    N: (inst M1/XX2/XX14/XX1) [g] /XX2/XX14/XX1/selb :: [s,d,sub] /XX2/XX14/XX1/pre_buff, N013, vss
57    P: (inst M2/XX2/XX14/XX1) [g] /XX2/XX14/XX1/selbb :: [s,d,sub] N013, /XX2/XX14/XX1/pre_buff, vdd
58    N: (inst M3/XX3/XX2) [g] /XX3/XX2/sel0bb :: [s,d,sub] /XX3/XX2/N001, N013, vss
59    P: (inst M4/XX3/XX2) [g] /XX3/XX2/sel0b :: [s,d,sub] N013, /XX3/XX2/N001, vdd
60    N: (inst M23/XX2/XX14/XX2) [g] /XX2/NC_06 :: [s,d,sub] N013, vss, vss
61
62  2 DEVICES do not match:
63  DEVICE P: (inst M4/XX3/XX2) [g] /XX3/XX2/sel0b :: [s,d,sub] N013, /XX3/XX2/N001, vdd
64  DEVICE N:
65  (inst M3/XX3/XX2) [g] /XX3/XX2/sel0bb :: [s,d,sub] /XX3/XX2/N001, N013, vss
66
67  2 devices and 0 nets written to C:\Users\scott\OneDrive\Documents\School\Winter 2019\ECEN 351 VLSI Design\Final Project\LTspice
    Files\register_file.err
68
69  Gemini completed at 23:36:47 on 11/04/2019
70
```

Figure 15: LVS errors for register file

CONCLUSION

There were definitely some challenges when it came to having multiple people working on the same design. One was that we all have a little different idea of how the design should look and how we like to layout our circuits in glade. It took a little more effort to get our different parts to match up so that things fit together nicely. A second challenge was that there are sometimes slightly different ways to implement a circuit and there can be mix-ups if one person is doing the schematic and another person is doing the layout. We ran into this when doing the 8 to 1 MUX. A version we had created in LTspice was using 3 2 to 1 MUXes in the 4 to 1 MUX while the Glade version was only using transmission gates and inverters to for the 4 to 1 MUX. This led to the two not quite matching up when the LVS was run.

I probably spent a little too much time on this lab, but I definitely learned a lot and really enjoyed laying out a more complicated circuit. I like the idea of working in teams to create a very large circuit. One thing I think would have helped is if we were all a little more experienced and had more time in our day to sit down together and collaborate. I also loved using the standard cells. It added a little bit of abstraction that made things easier to put together. I can see how this would be extremely valuable when designing a very large IC. I can't imagine having to try and layout each individual transistor separately.