Final Lab – ROM and Decoder Component and Verification

Christopher Scrosati

May 10, 2023

Introduction:

Read Only Memory is often used when a memory object does not need to be written to often, or when something is not meant to be written to after production. The different types of ROM include Flash Memory, EEPROM, EPROM, MROM and PROM. For this project I decided to do a 5-bit, 4-word NAND MROM memory module with a 2:4 Decoder for word selection.

Design:

I started with a dot diagram of the ROM module I wanted to make.

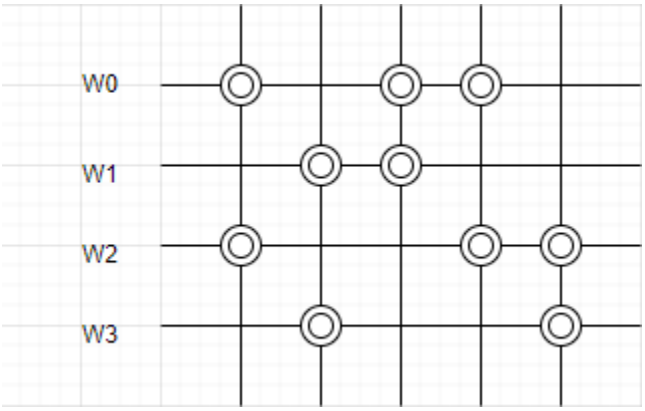

Figure 1: Dot Diagram

Each dot represents a connected NMOS transistor in the lattice structure of the memory module. From this I created a full memory structure schematic

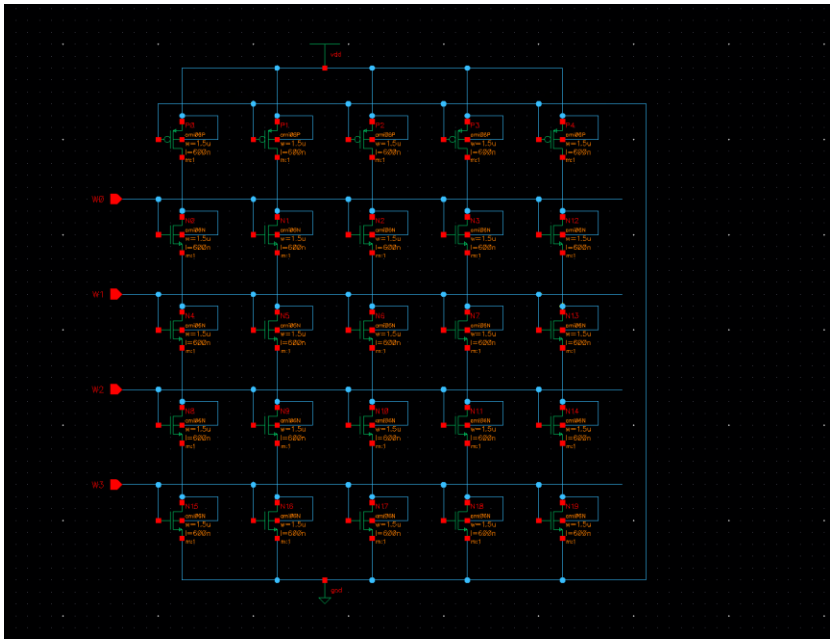

Figure 2: ROM Structure

I then removed and shorted connections where the transistors were absent from the dot diagram. The word lines from 0 to 4 are the binary representation of the numbers: 22, 12, 19, and 9. These numbers each represent a letter in the alphabet, spelling “VLSI”.

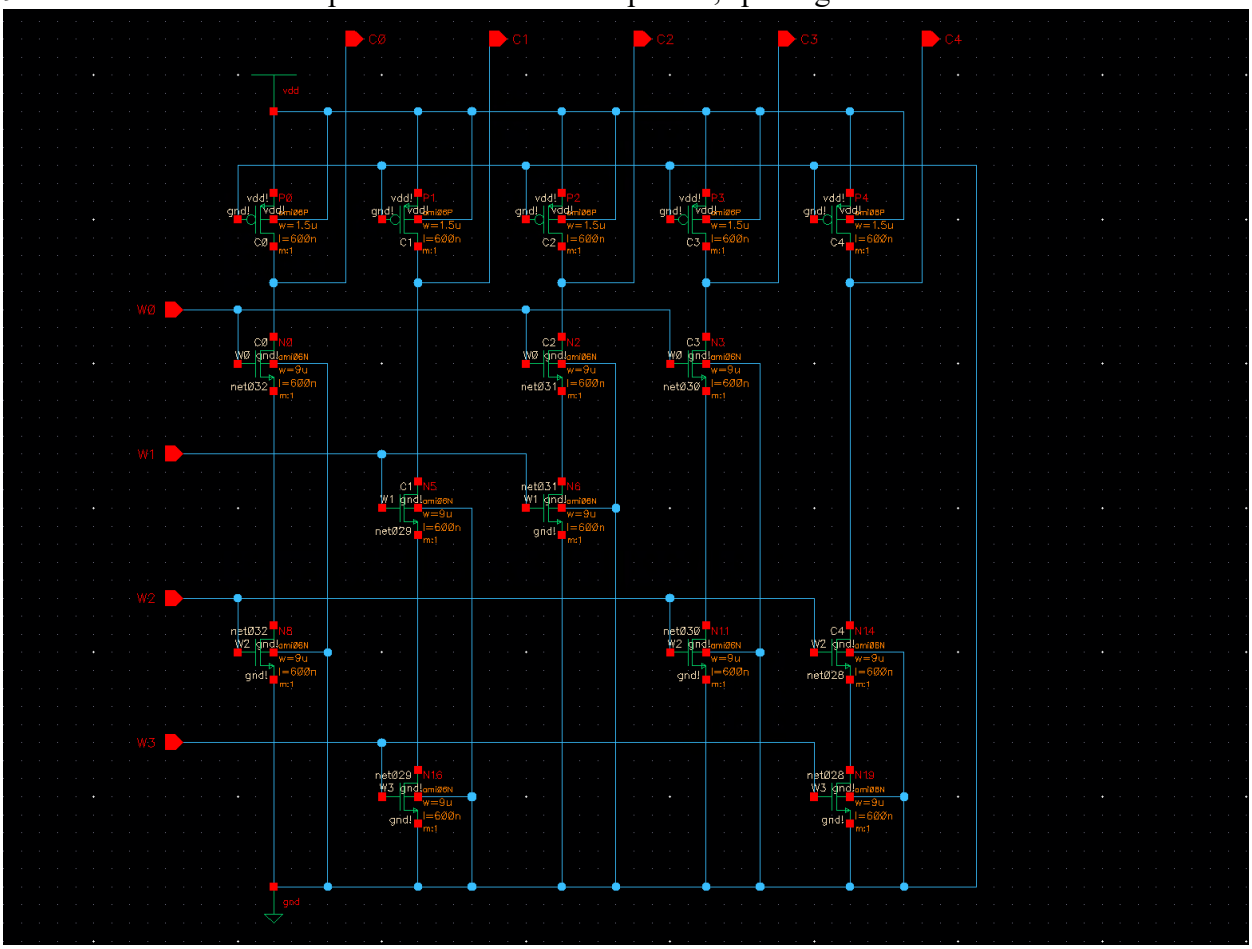

Figure 3: Final ROM Structure

I then created a layout for this design using the same methodology. This shows the scale of the NMOS transistors when compared to the PMOS in a pseudo NMOS configuration.

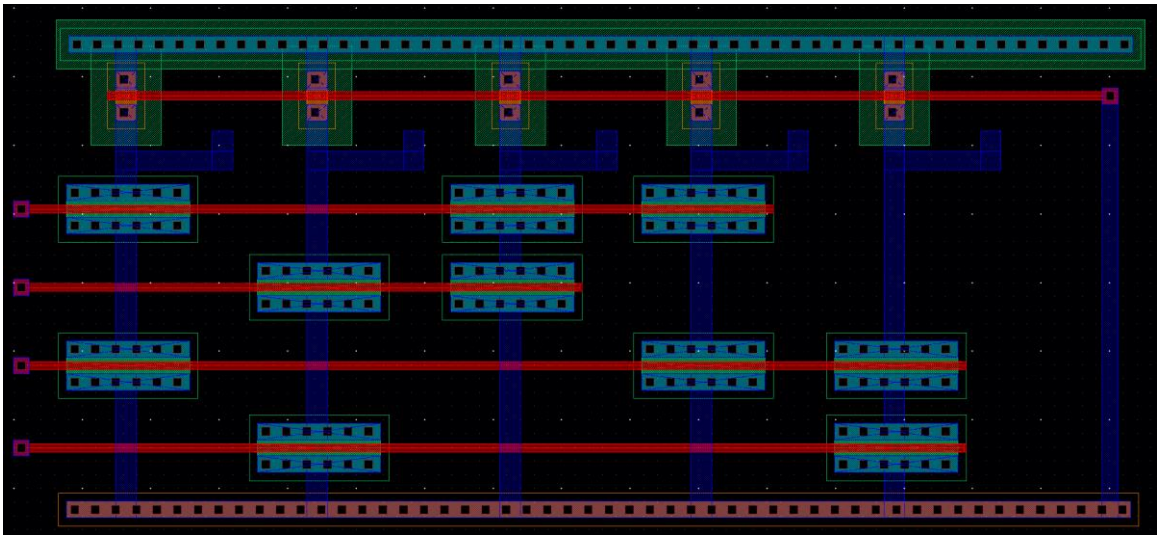

Figure 4: ROM Layout

I created an extracted view and successfully ran LVS, returning a match. The next step was to create a symbol and simulate the ROM to ensure proper functionality.

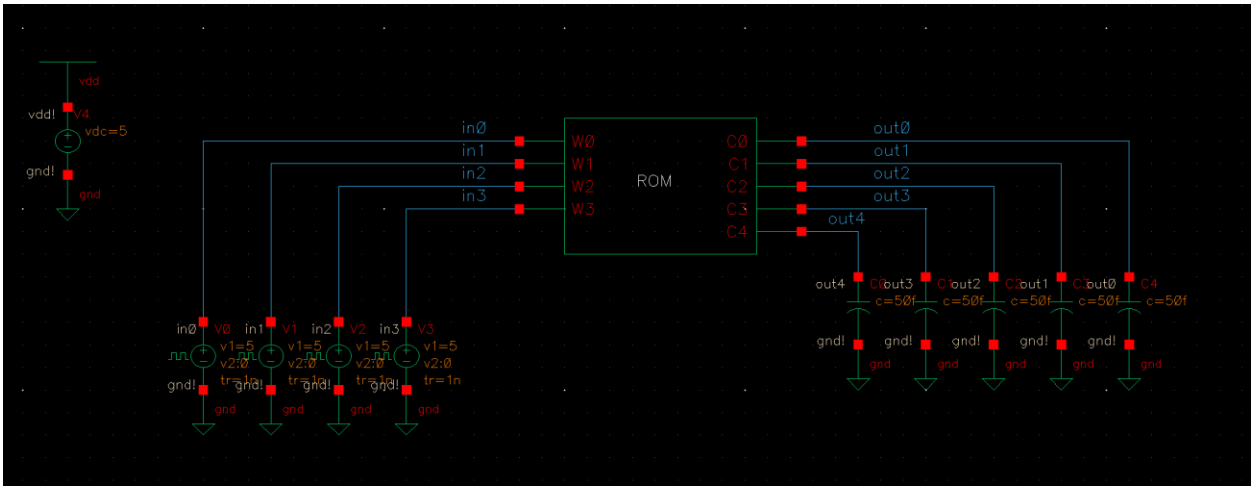

Figure 5: NAND ROM Simulation

I drove each word line manually using delayed vdc objects. The word lines should all be high, and the active word goes low for a read.

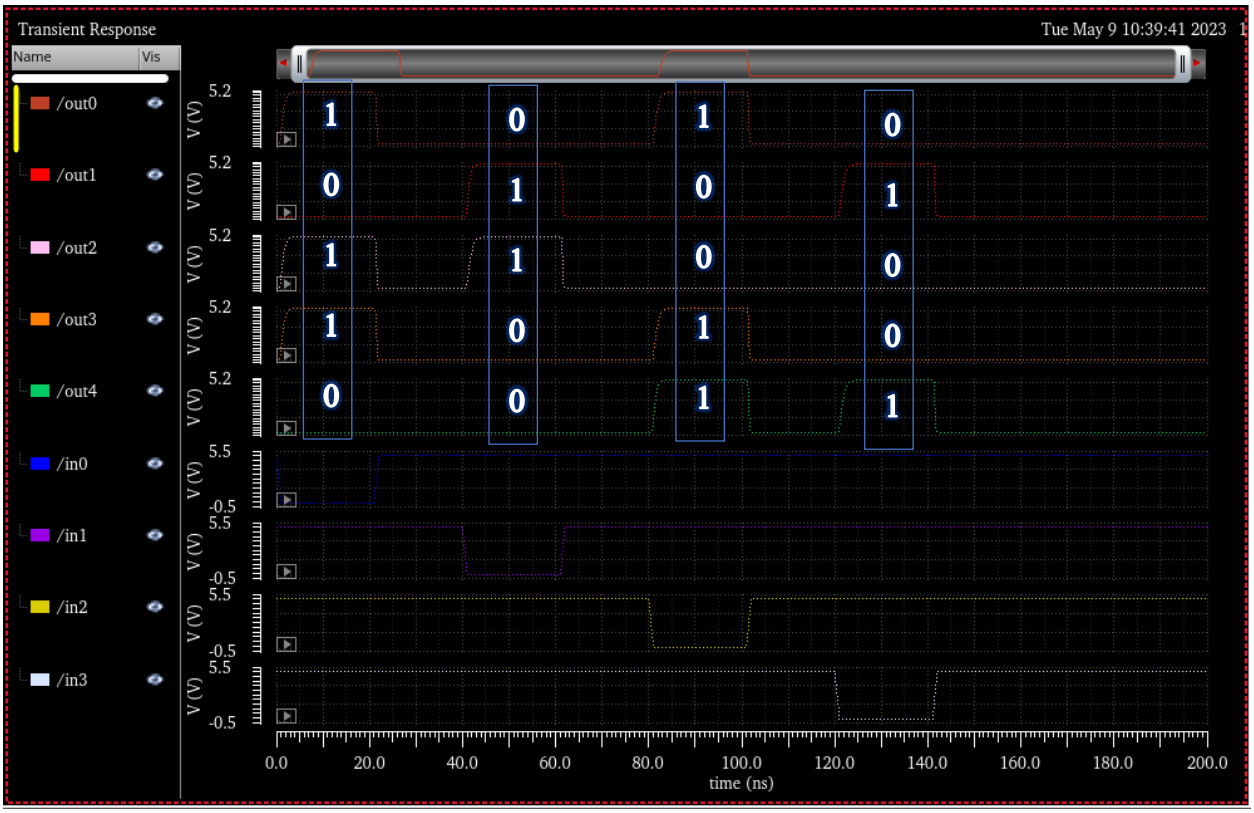

Figure 6: NAND ROM Waves

This is the expected output given the inputs are driven one at a time.

Next, I needed to create a decoder for the NAND ROM, using 2 select lines I should be able to select one of the 4 words, preventing a multiply driven instance. Since I needed the output to be inverted for the word lines to work properly, the 2:4 decoder was made using only NAND gates and 2 inverters.

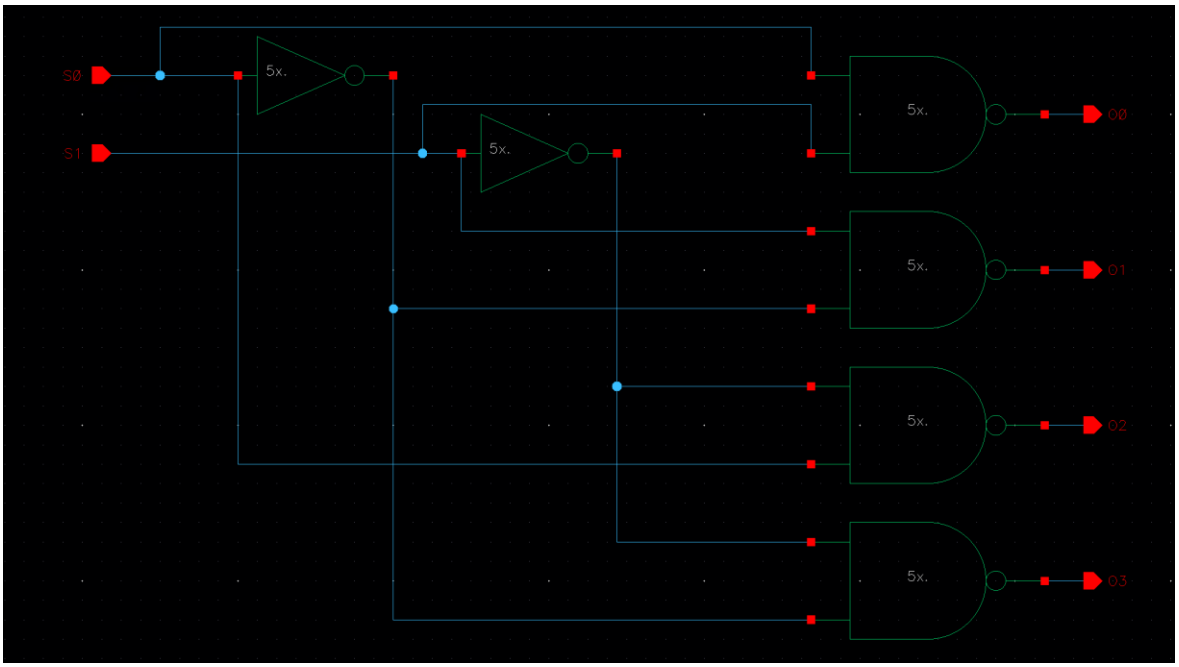

Figure 7: 2:4 Decoder Schematic

For the decoder the following was mapped:

S1	S0	Output
0	0	0
0	1	1
1	0	2
1	1	3

I thought it would be interesting to use the components we made in previous labs to create this component, resulting in the following layout.

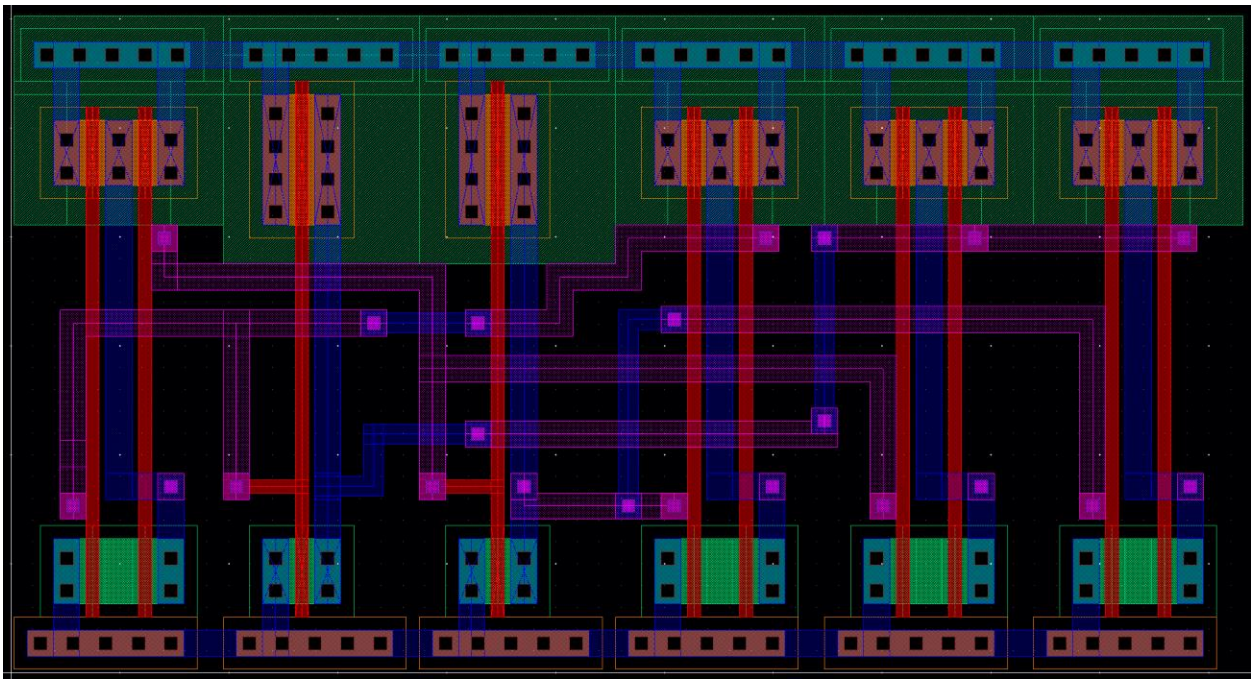

Figure 8: 2:4 Decoder Layout

All the contacts exist in the metal 2 layer, with some traces needing to run under using metal 1. This was the cleanest I could make the design from pre-made cells, while still fitting within the 10-lambda height requirement of previous labs. I left metal 3 open for later connections. Then I simulated the inputs.

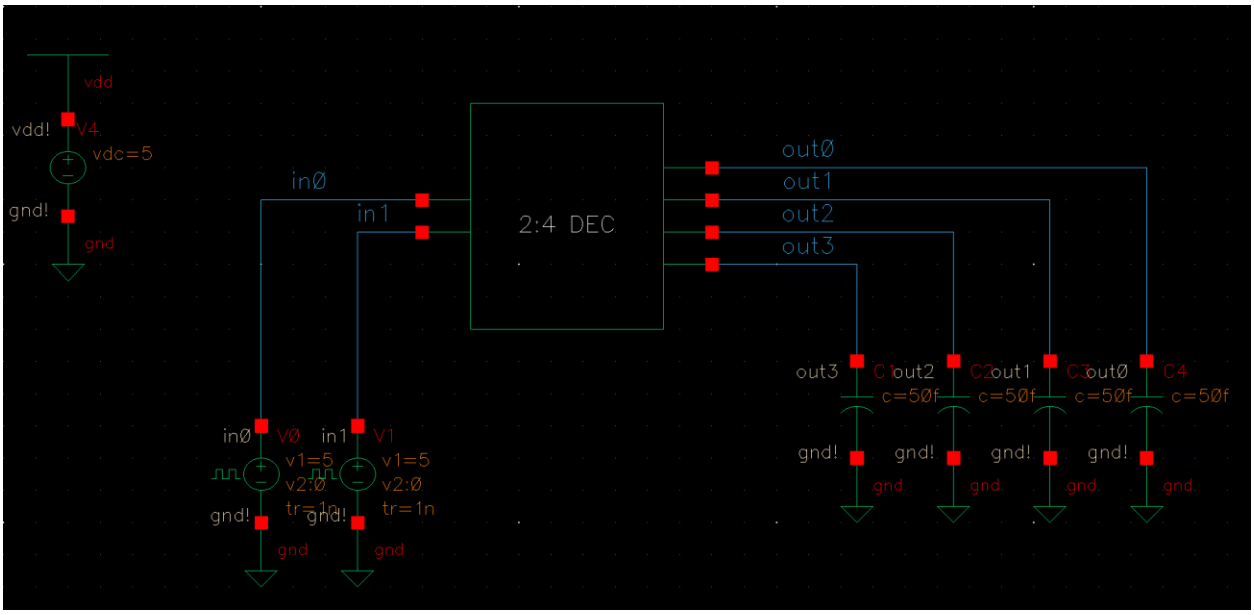

Figure: 9: 2:4 Decoder Simulation

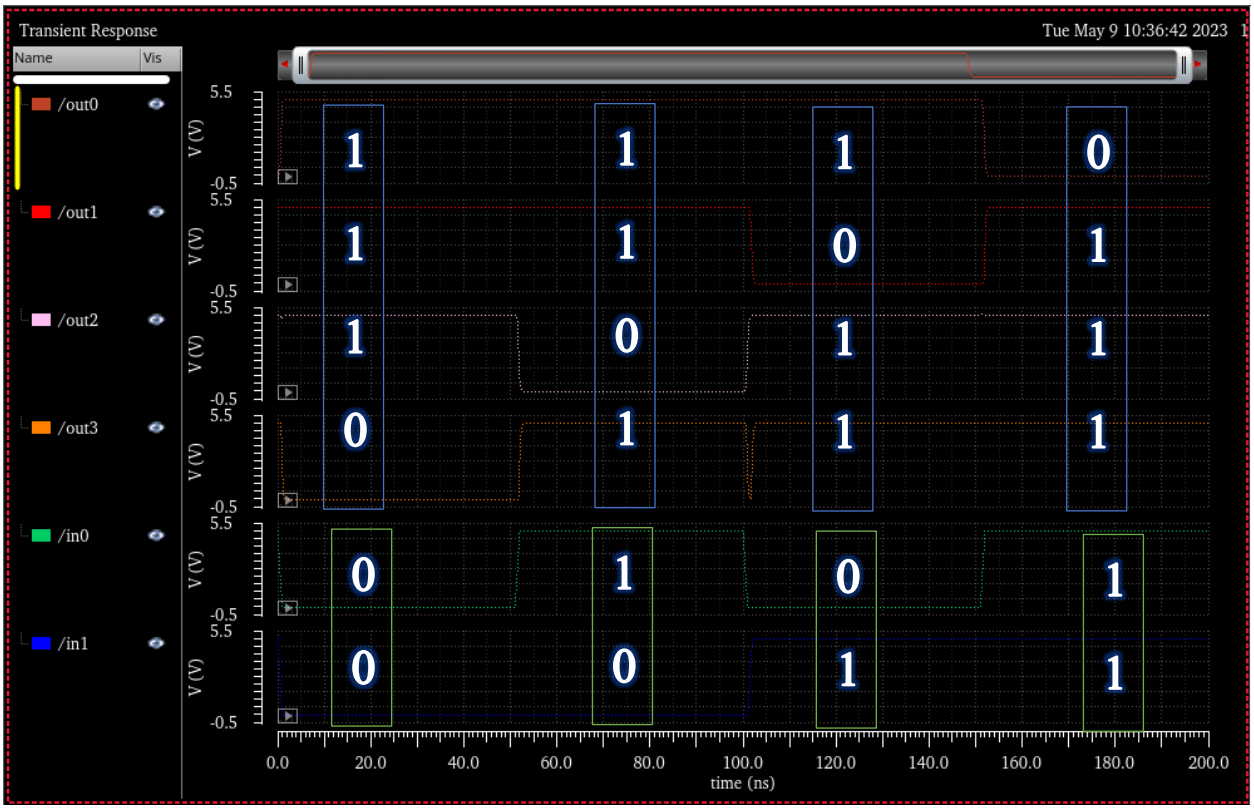

Figure 10: 2:4 Decoder Waves

The waves generated by this schematic show correct operation, with some minor glitching. The last portion to complete was a combined testing of the two components.

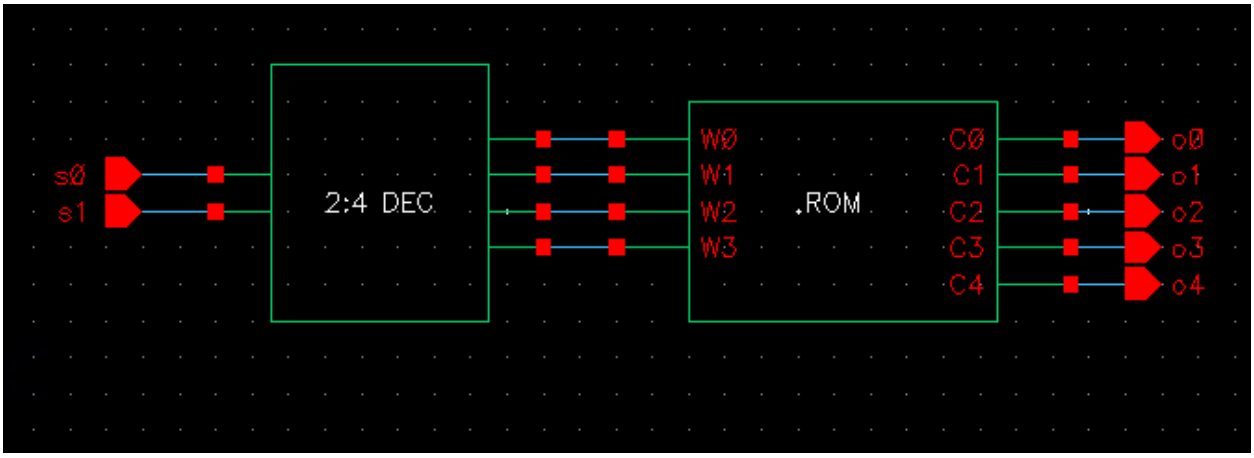

Figure 11: ROM With Decoder Schematic

Since the metal 3 layer was left unused in both components, I decided to use it to connect the two as well as for their inputs and outputs.

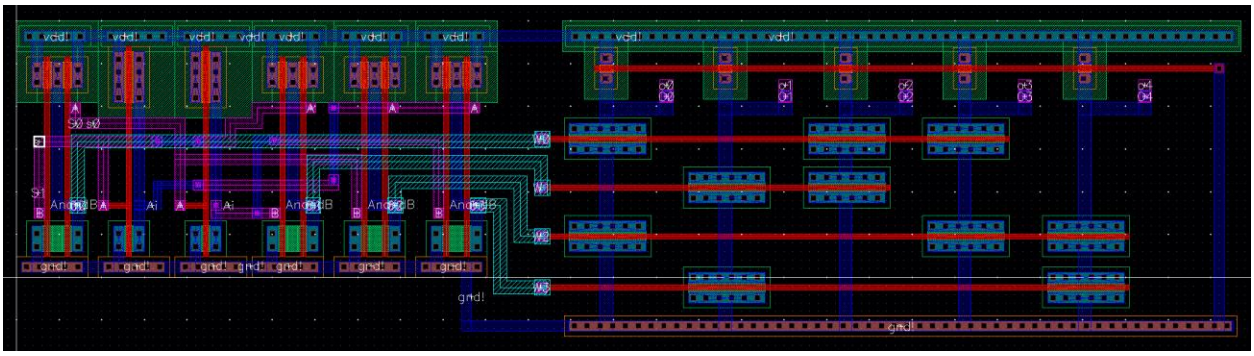

Figure 12: ROM with Decoder Layout

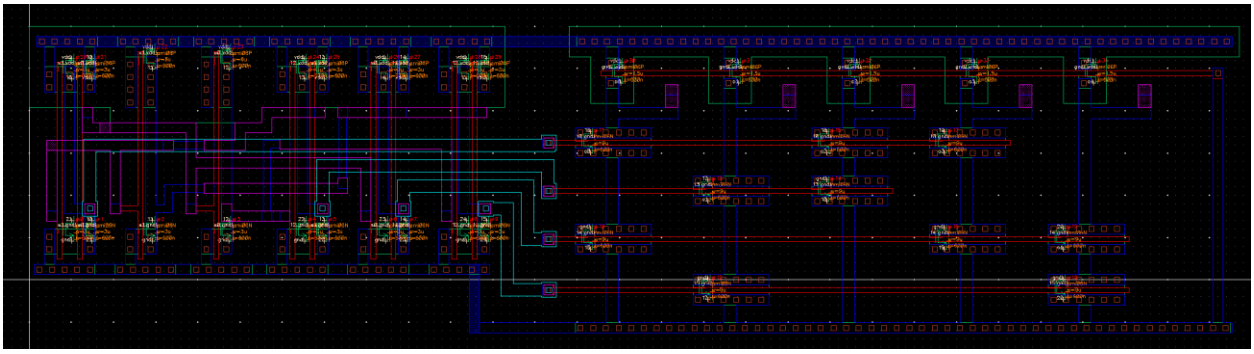

Figure 13: ROM with Decoder Extracted

After the LVS Completed I created a symbol and wired it up for testing.

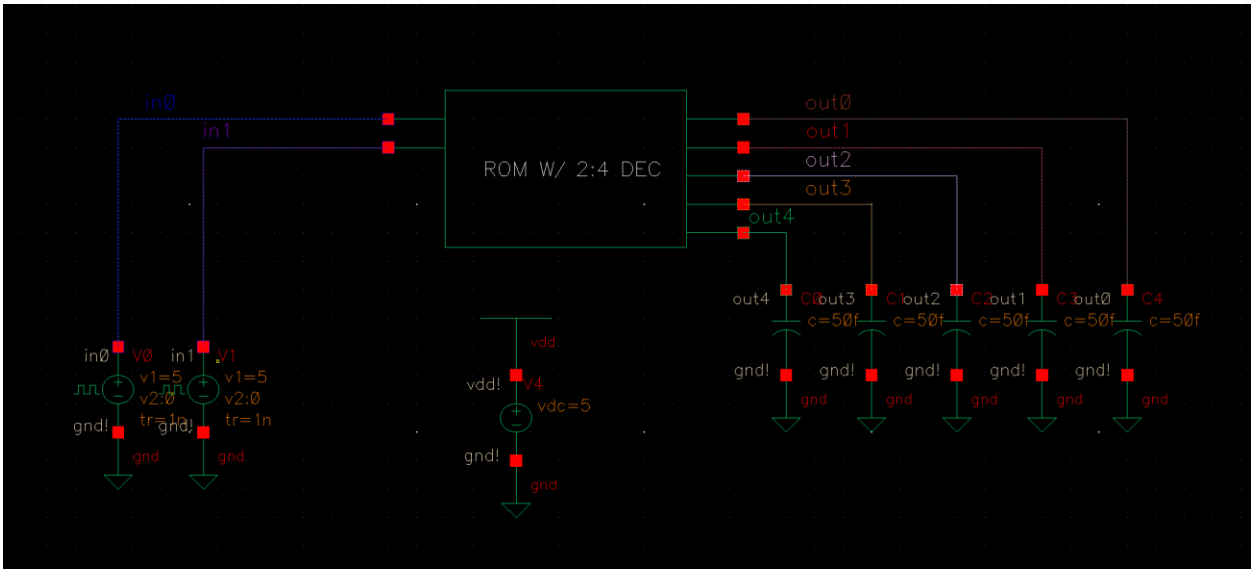

Figure 14: ROM With Decoder Simulation

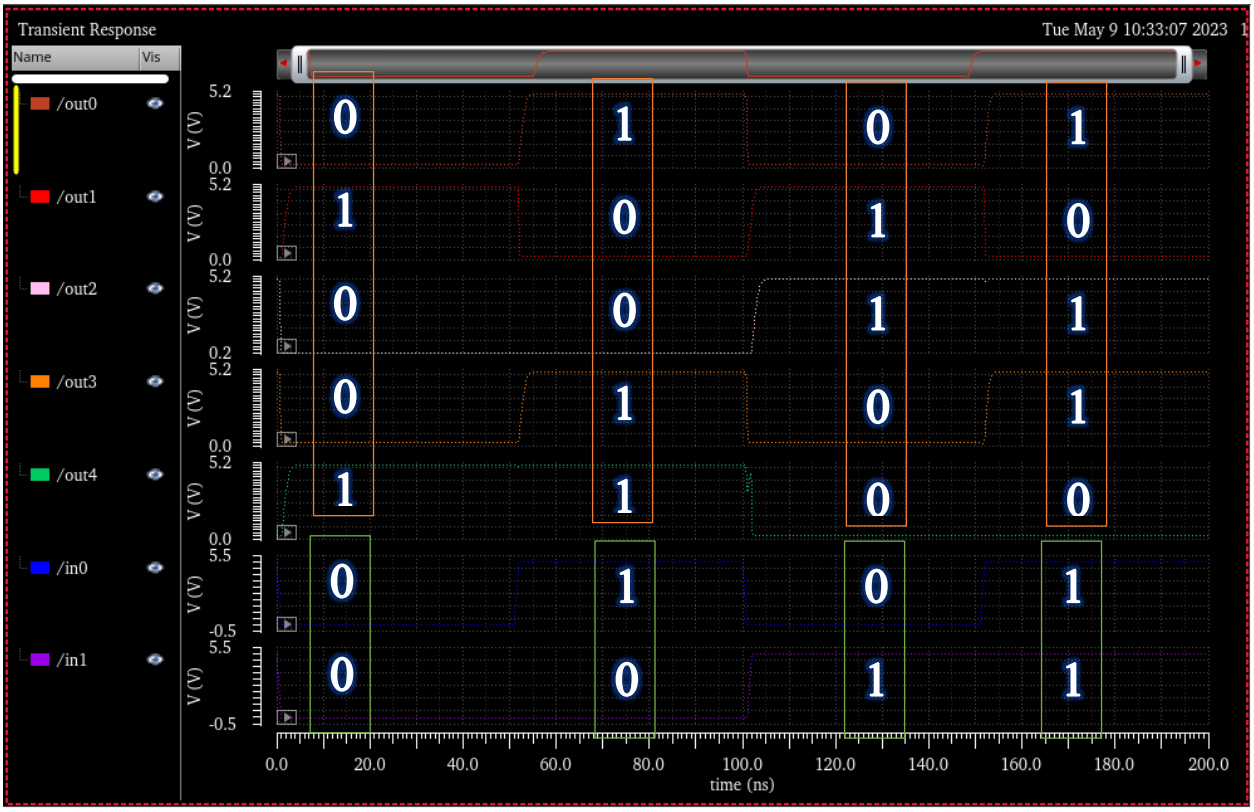

Figure 15 ROM With Decoder Waves

When the input order is reversed it will display “VLSI” in order via a binary representation.

Data and Analysis:

One issue I encountered early on with the design of the RAM module was the lack of a full-zero voltage. This can cause stability issues when integrating in larger systems and is due to the pseudo-NMOS transistors constantly being high. This also meant that in order to get a clean signal I had to size the NMOS transistors 9 times the size of the PMOS transistors.

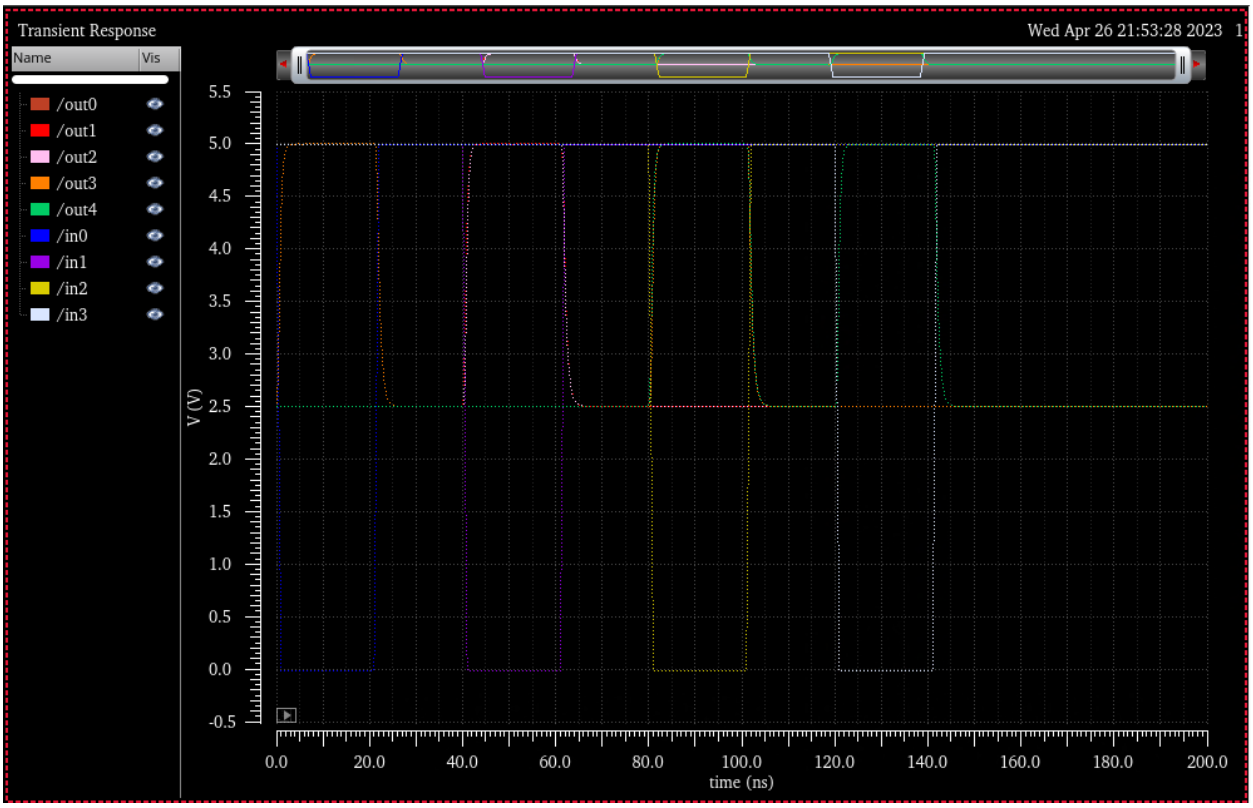

Figure 16: Pseudo NMOS Issues

This implementation of ROM is relatively straightforward and works well enough to be reliable in the long term, although more systems implement EEPROM and other flash memory solutions since they can be easily fixed or updated. This style of ROM would require the hardware be sent back to the manufacturer and a new ROM module would have to be manually installed.

Conclusion:

ROM is a vital tool in computer systems, and the experience I gained researching this topic has been invaluable. Personally, as a classic-video game hardware enthusiast I find it entertaining to look at this implementation of ROM, as well as its place in the computing world. The ROM was relatively easy to design, but the decoder could be better optimized. I am satisfied with the final outcome of the project, as well as the final implementation of both designs.