Design and Layout of CMOS Memory in Cadence

The Ohio State University, Electrical and Computer Engineering, Mixed Signal VLSI

Youngsoo Kang

Joseph Chiocca

Qian Lam Hao

Matthew Wiechec

Derek Hong Lew

Sami Arman

Table of Contents

Background

Memory is a digital circuit capable of retaining information over time. In computer architecture, separation is made between volatile memory and non-volatile storage. Volatile is the property of digital memory to lose retained information when power is lost. Storage, or non-volatile mediums, are designed to retain information after power is removed. Random Access Memory (RAM) is a volatile digital space where computers store variables, data, images, etc required for operation of programs. Without RAM computers would need to store all active files in non-volatile mediums which are much slower. It should be noted that memory is around 100,000 times as fast as storage. [1] In this project we will focus on volatile memory, specifically SRAM.

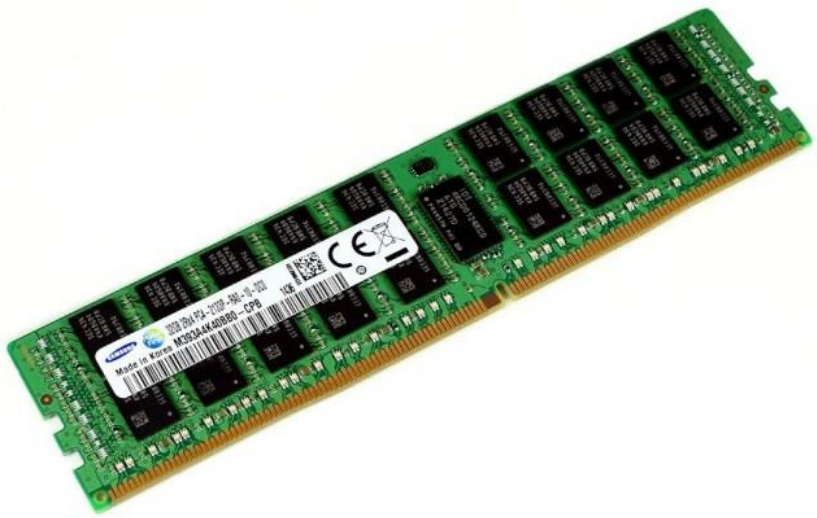


Figure 1: Typical Dynamic Random Access Memory (DRAM) modules used in modern computers [2]

Memory is composed of many cells, each of which can store a value, either zero (0) or one (1). Typically, binary memory employs latches to retain these values. They can be set to an input value or reset to their default state. This is the basis of memory. Each latch is constructed with transistors, and there are two major design categories for latches and other logic structures; static and dynamic. Static logic uses pullup (PUN) and pulldown networks (DUN) to hold a logical state. Dynamic logic uses transistors like a series of gates allowing -- or impeding -- the flow of current. Because static logic is primary controlled by voltage it is faster than the current operated dynamic logic. Static logic is also simpler and more reliable than dynamic logic. Although Static logic is faster it requires more transistors, thus the prevalent type of logic used in modern ram is dynamic logic.

There are ways to mitigate the disadvantages of dynamic logic, however, for simplicity we will be designing static memory. Due to its robustness and speed SRAM is employed in many embedded systems. An example of SRAM in an embedded platform is Microchip's AVR series of microcontrollers.

Even simple computers use thousands, millions, and trillions of memory cells to store data. It would be impossible for the Central Processing Unit (CPU) to access each memory cell individually. Each cell is assigned an address, which allows the cell to be indexed. This is called addressing, and it simplifies the CPU/memory interface. Without addressing the interface would have to equal the number of cells in memory, making large memory values impossible.

Introduction

The design created for this project is a 1024 bit SRAM module. Below is a block diagram of the system:

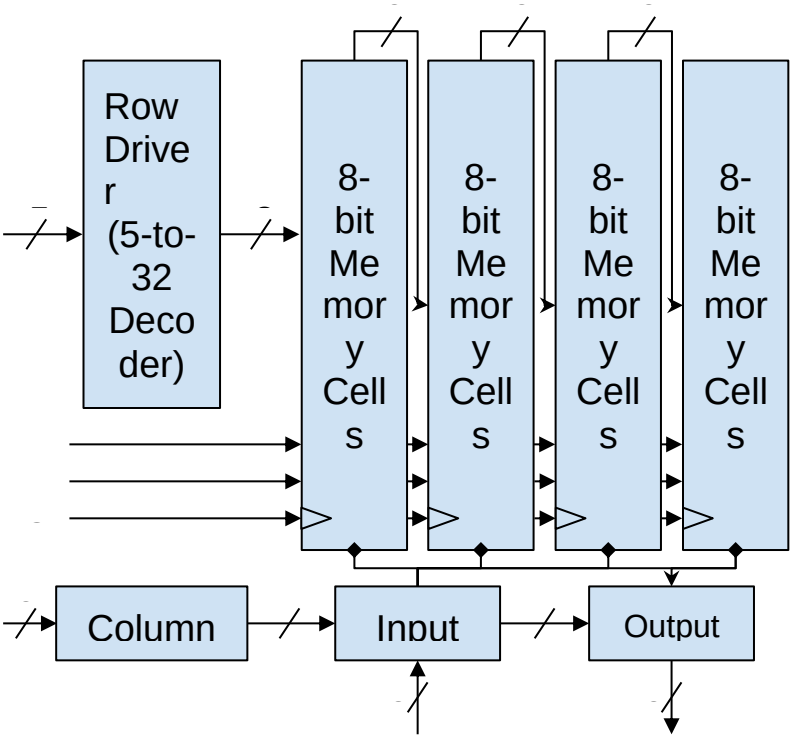


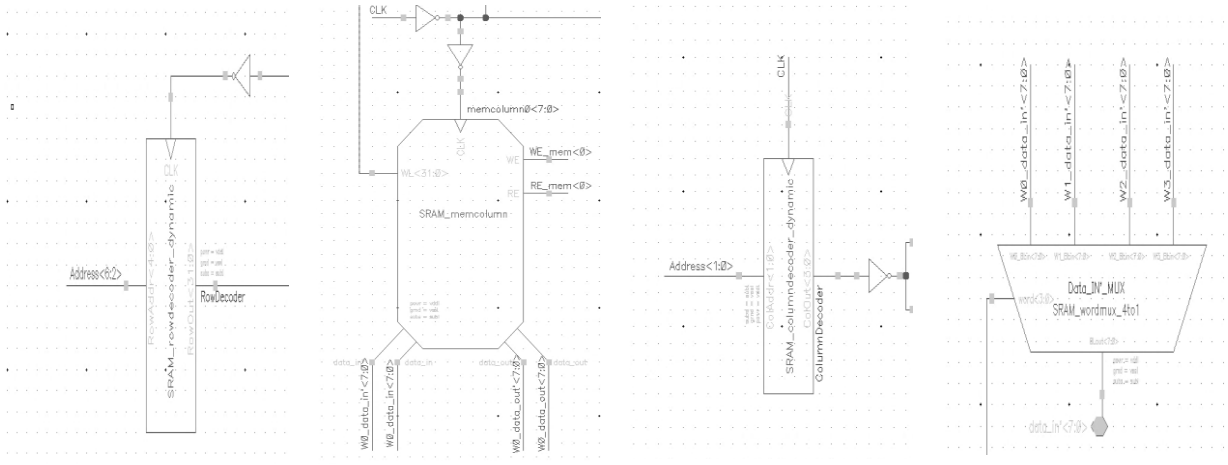
Figure 2: Block diagram of 1024 bit SRAM module. Note the complement of the input and output column drivers are omitted.

For the sake of simplicity the column and output column driver complement are omitted from this diagram. They are used to check the metastability of the memory system. A short operation description will be given here, following sections will explain

at length the operation of each block. To write data Write Enable (WE) is pulled high, then a row is selected by sending a 5-bit number to the row driver. Using a 2-bit number column 0 through 3 is then selected. Data is sent to the input column driver. On the next rising edge the data on the 8-bit input is stored at the current row/column selected. To read data WE is low while Read Enable (RE) is high. The row is selected by sending a 5-bit number to the row driver, likewise a 2-bit number is used to select the column. On the next rising edge the data in the selected location appears on the output column driver.

Top Level Block Schematics

The palette of standard blocks used in the top level design are shown below in the table:



Row Select	8-Bit Memory Cell	Column Select	Column Driver

Figure 3: Top Level Modules

Below is a logic gate level schematic of the inner workings of each top level block. In Figure 4 the inner workings of the row select is shown. It can be observed that each input (the five center pins) connect to an inverted and non-inverted line which propagates vertically. Intersections are made horizontally with the gates of NMOS transistors to create multiple input AND gates.

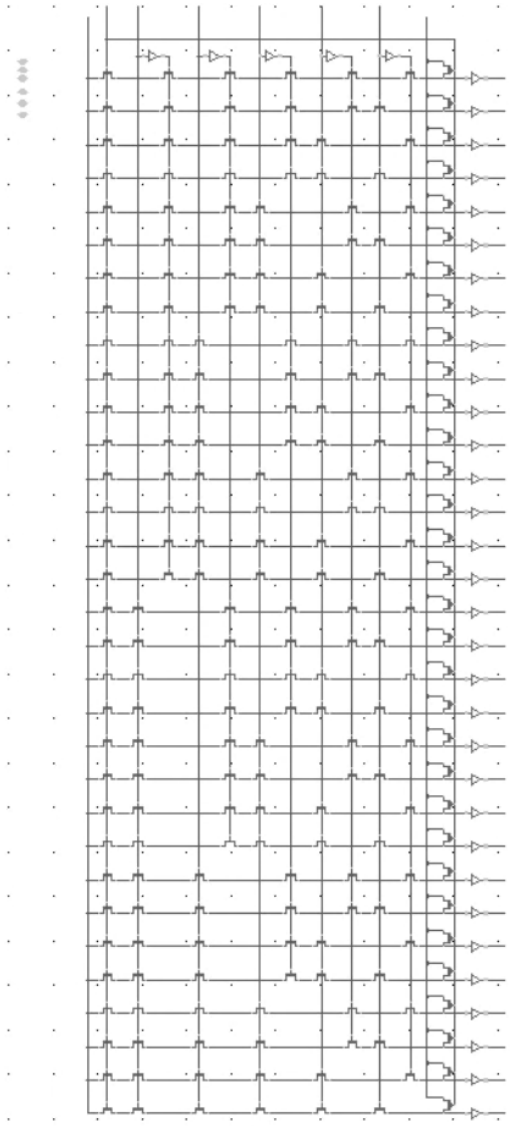


Figure 4: Row Select Internal Diagram

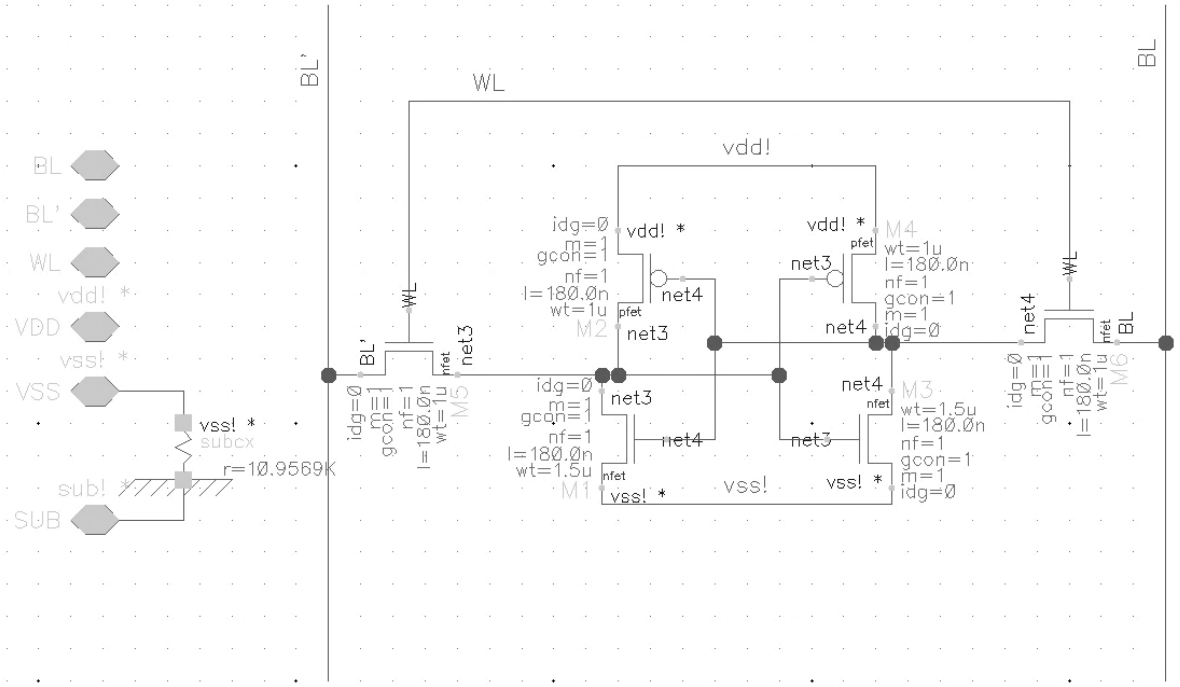


Figure 5: Schematic View of the Precharge Internals

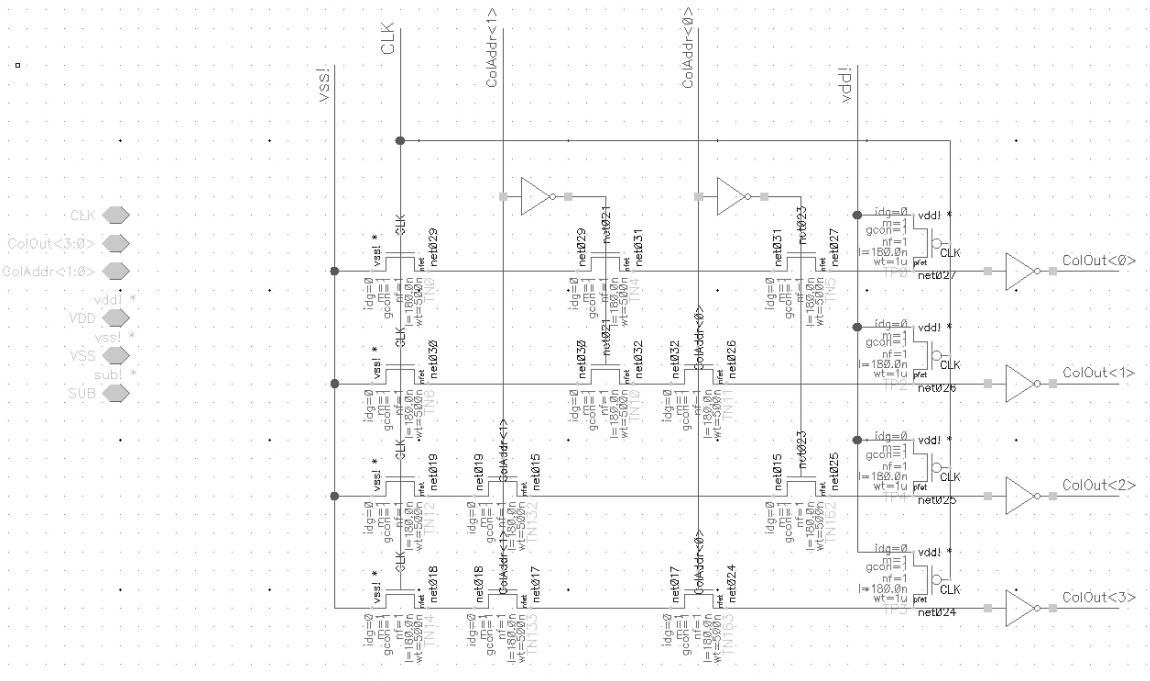


Figure 6: Column Driver Schematic View

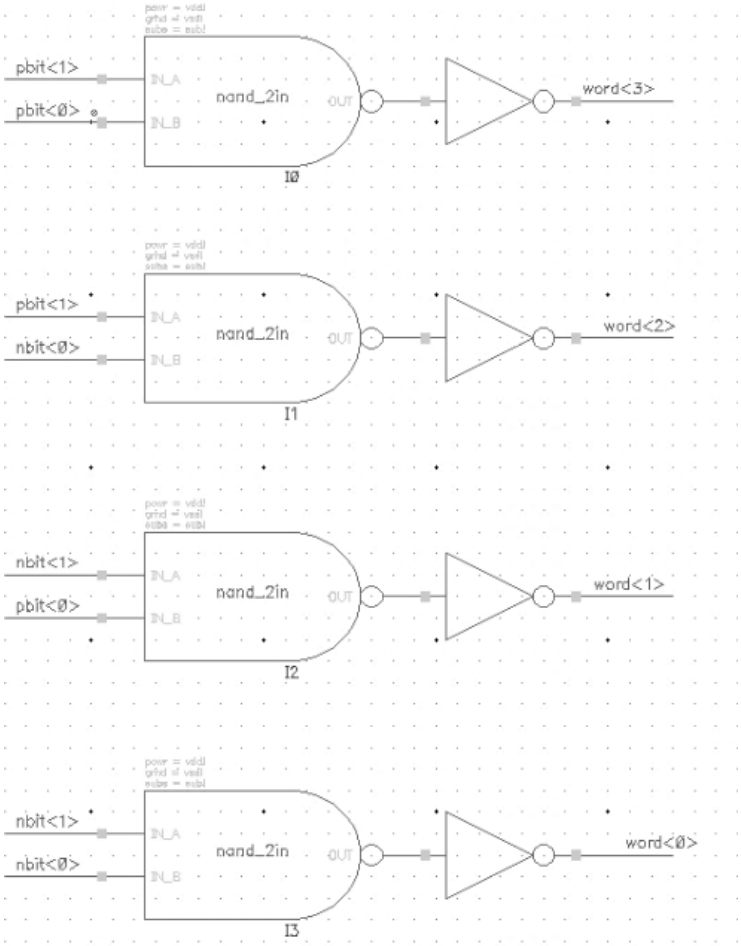


Figure 7: Column Select Internals

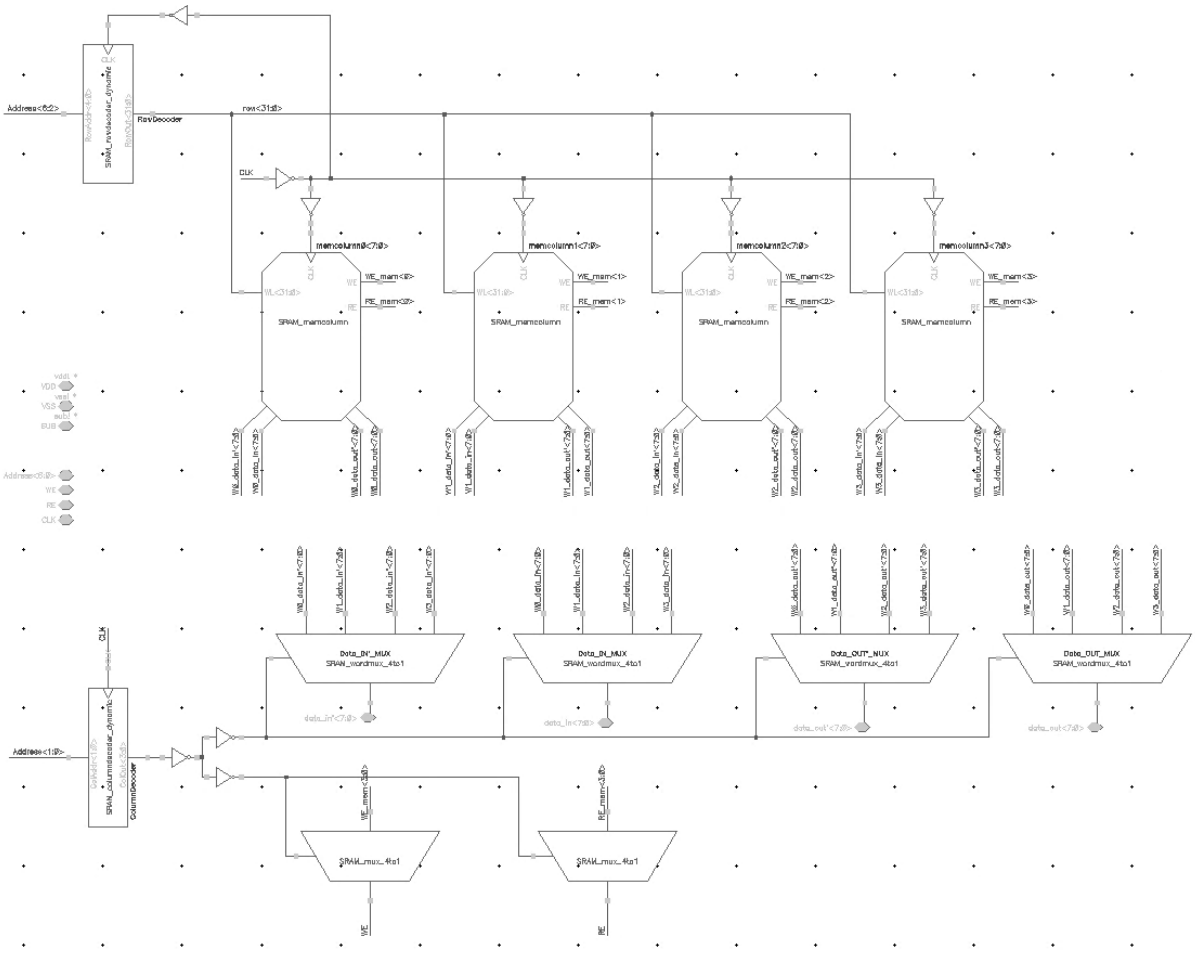


Figure 8: 32x32 Schematic

Figure 8 shows the entirety of the memory circuit and how each component is connected.

Low Level Block Schematics

Below are the logic gate level schematics which show the inner workings of each low level block. These logic units are used extensively in the design of this memory unit. The SRAM memory cells low level blocks consisted of a several different circuit designs. The circuit designs include a read and write schematic, a decoder, a precharge, a 4 to 1 word multiplexer, and the 32 x 32 module.

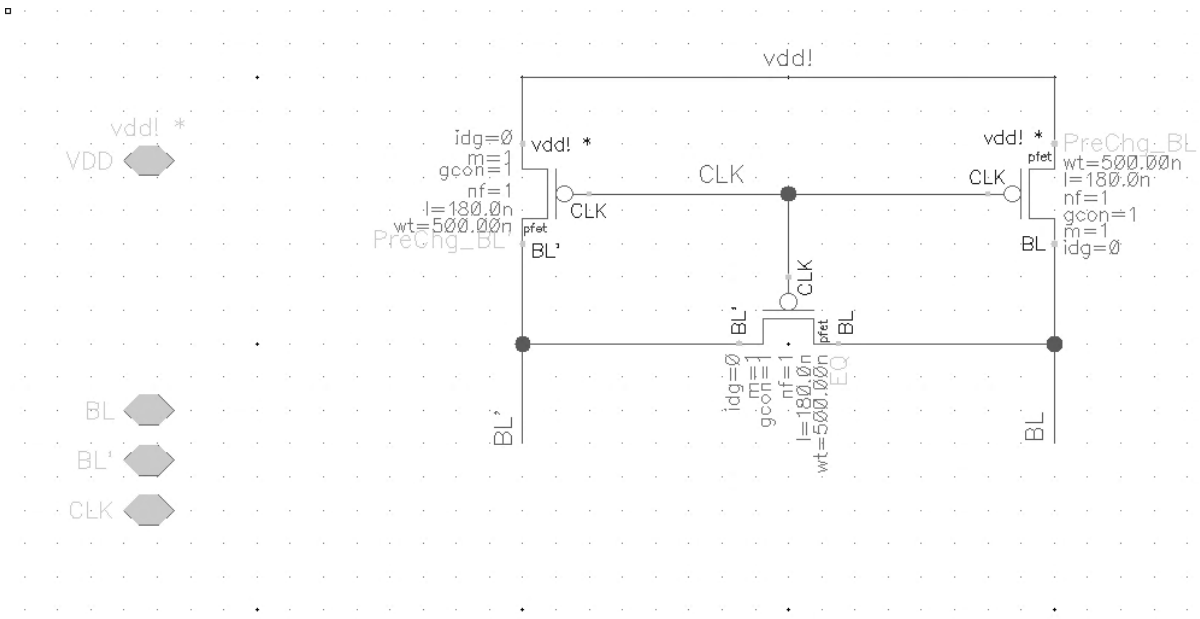


Figure 9: Precharge Schematic

The precharge schematic is given in figure 13. The precharge circuit helps to mitigate the delay caused by the gate capacitances on the BL and $\underline{BL}$ line.

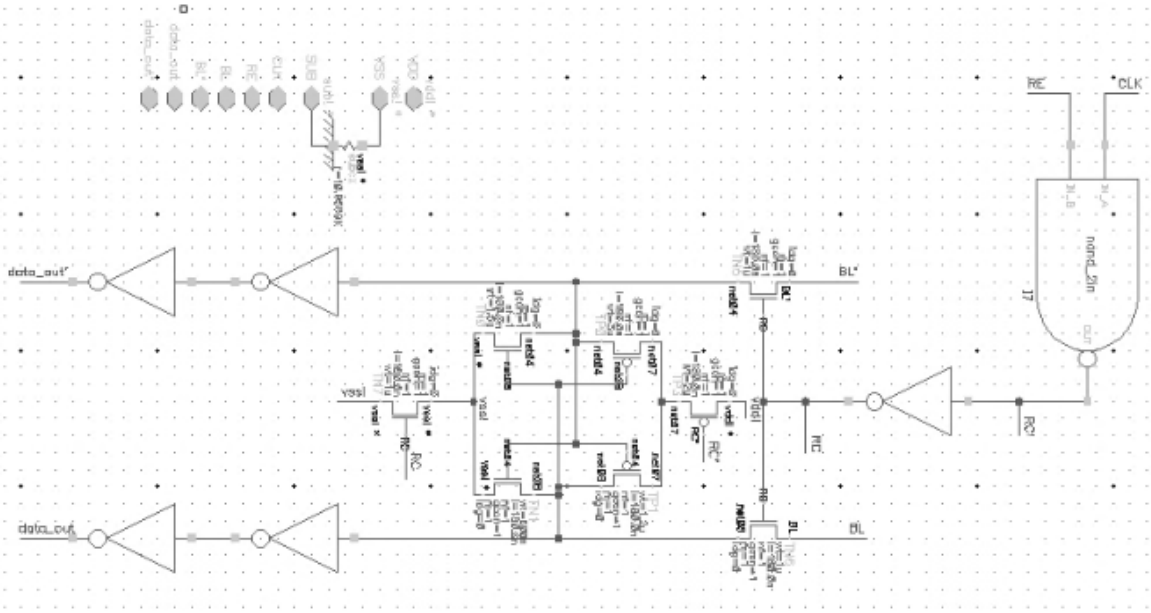


Figure 10: Read Schematic

The read circuit's purpose is to transfer the given memory cell value (High or Low) and send the high or low signal to the data output.

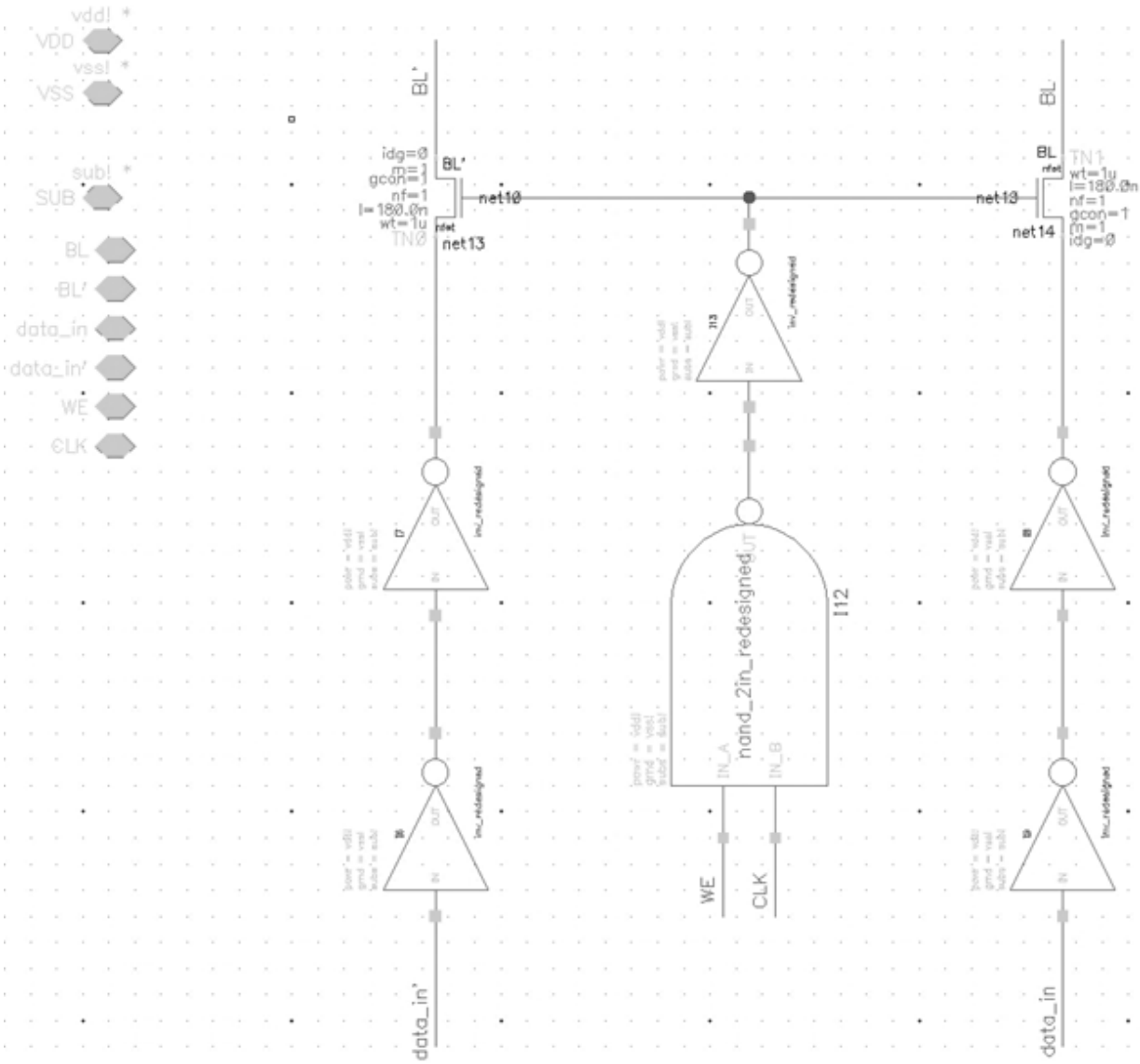


Figure 11: Write Schematic Internals

The Write circuit is designed to set a memory cell's value based on the given input data. The write circuit's purpose is to change memory cells values.

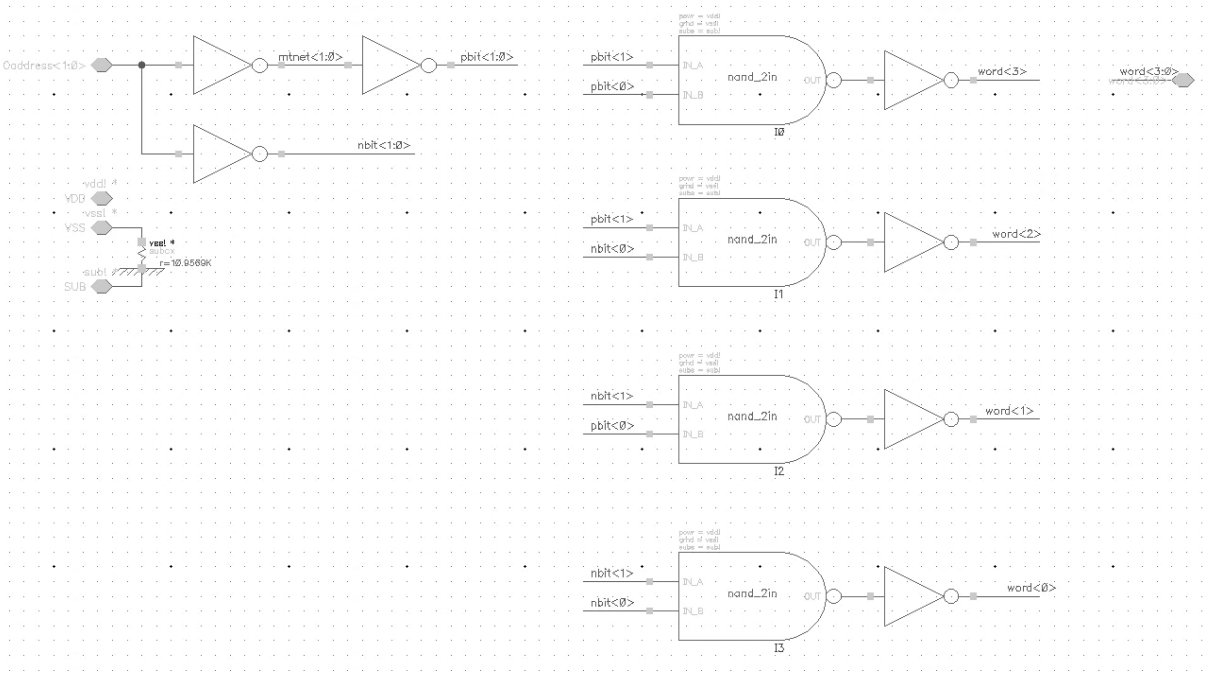


Figure 12: Decoder Schematic Internals

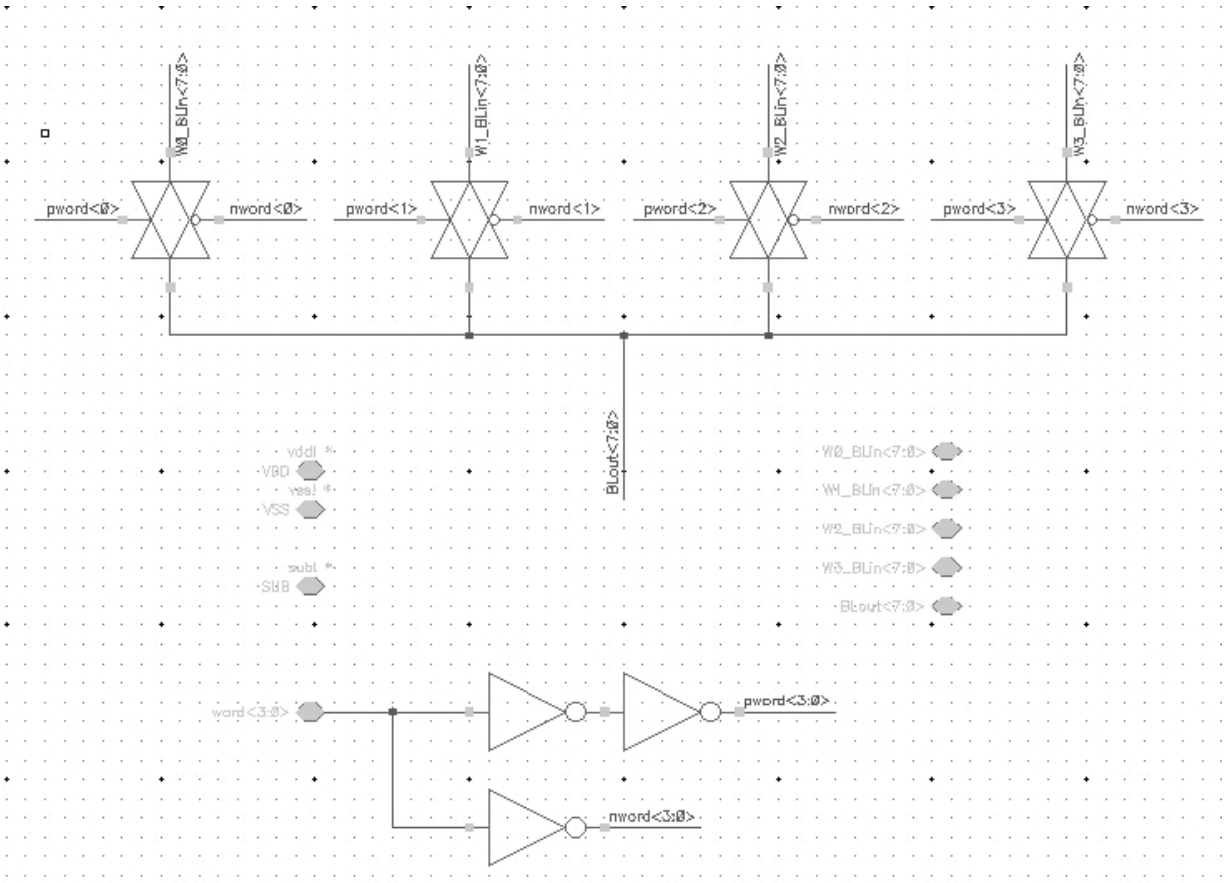


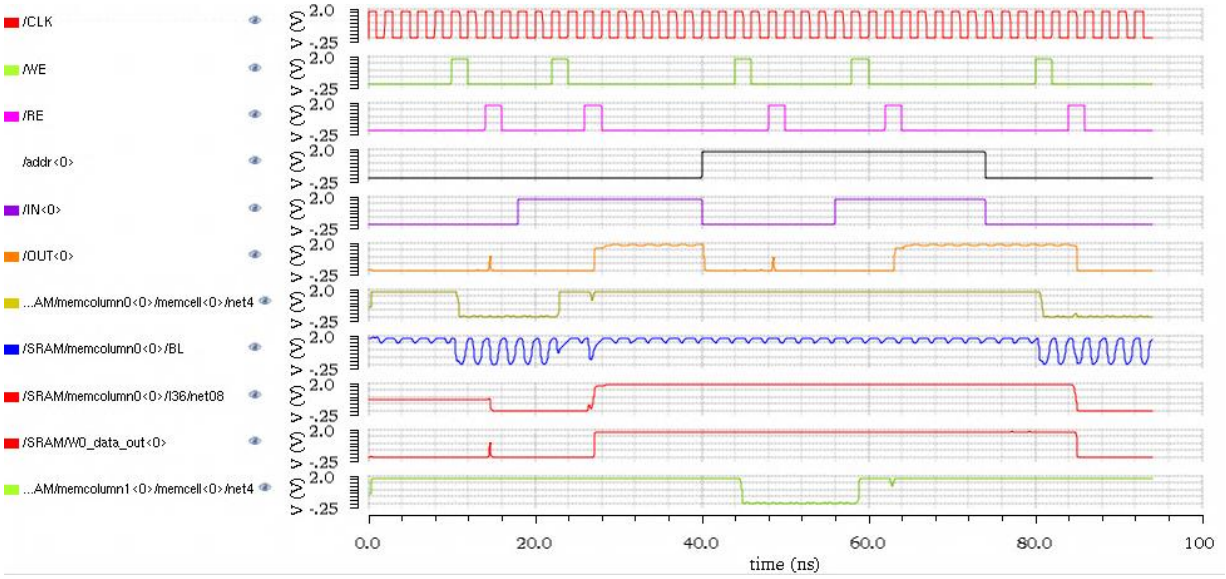
Figure 13: 4 to 1 Word Mux Schematic

The word multiplexor or column driver directs the flow of data to its respective column using a series of transmission gates. The transmission gates are normally open (they do not conduct), however when a voltage is applied to both gates the transmission gate closes (coducts) and data can pass through. By sending a 3-bit number to the word select input a respective transmission gate is chosen.

Pre-Layout Simulation Results

The pre-layout simulation results display how the entire cmos memory circuit works. The simulations show the function and purpose for each circuit.

The function of the write circuit is to transfer the value of the data input value to an address. This address can be either address 0 or address 1. Identifying the address value, identifies what memory cell will be altered. The read circuit's purpose is to read the memory cell's value and output that data value until a different memory cell is chosen or the read circuit goes high again and reads a different value. Displayed below is the full simulation for the pre-layout CMOS memory circuit.

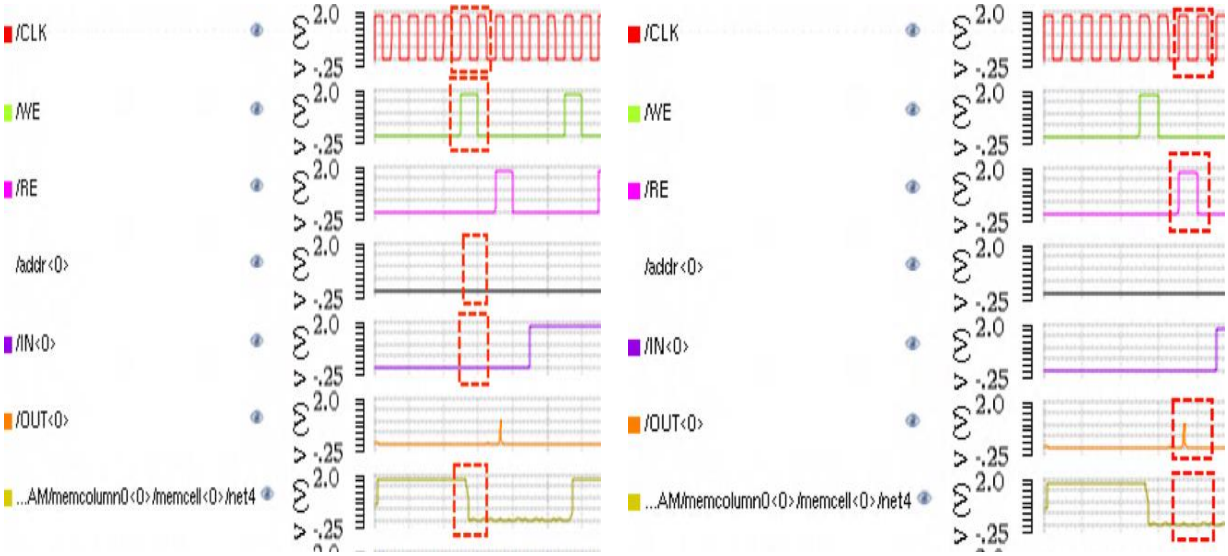


-/CLK = Clock
-/WE = Write
-/RE = Read
-/addr<0> = Address assignment
-/IN<0> = Data Input
-/OUT<0> = Date Output
-memcolumn0<0> = memory cell 0
-memcolumn1<0> = memory cell 1

Figure 14: Pre-Layout Complete 1024 (32 X 32) SRAM Results

An example of how the CMOS memory works can be seen by looking at the different sections of the simulation. The image below shows that at the rising edge of /WE and clock, addr<0> = 0 and /IN<0> = 0. This results in memory cell 0 transitioning from 1 to 0. What occurs next in the simulation is an activation of the read circuit. At the rising edge of the clock and /RE, memory cell 0x0000 was read and lead to the data output =

0. No change is present at the output because it was already outputting the value 0. These two circuits being used can be seen in the two images below.



Pre-Layout Write Circuit High *Pre-Layout Read Circuit High*

Figure 15: Read & Write circuits

The output value can change without the read circuit being high. This change can occur when the memory cell is switched. When this occurs, the output value will change into the new input value placed. An example is displayed below. The simulation shows /addr<0> = 1, /IN<0> = 0, and a delay where /OUT<0> transitions from 1 to 0.

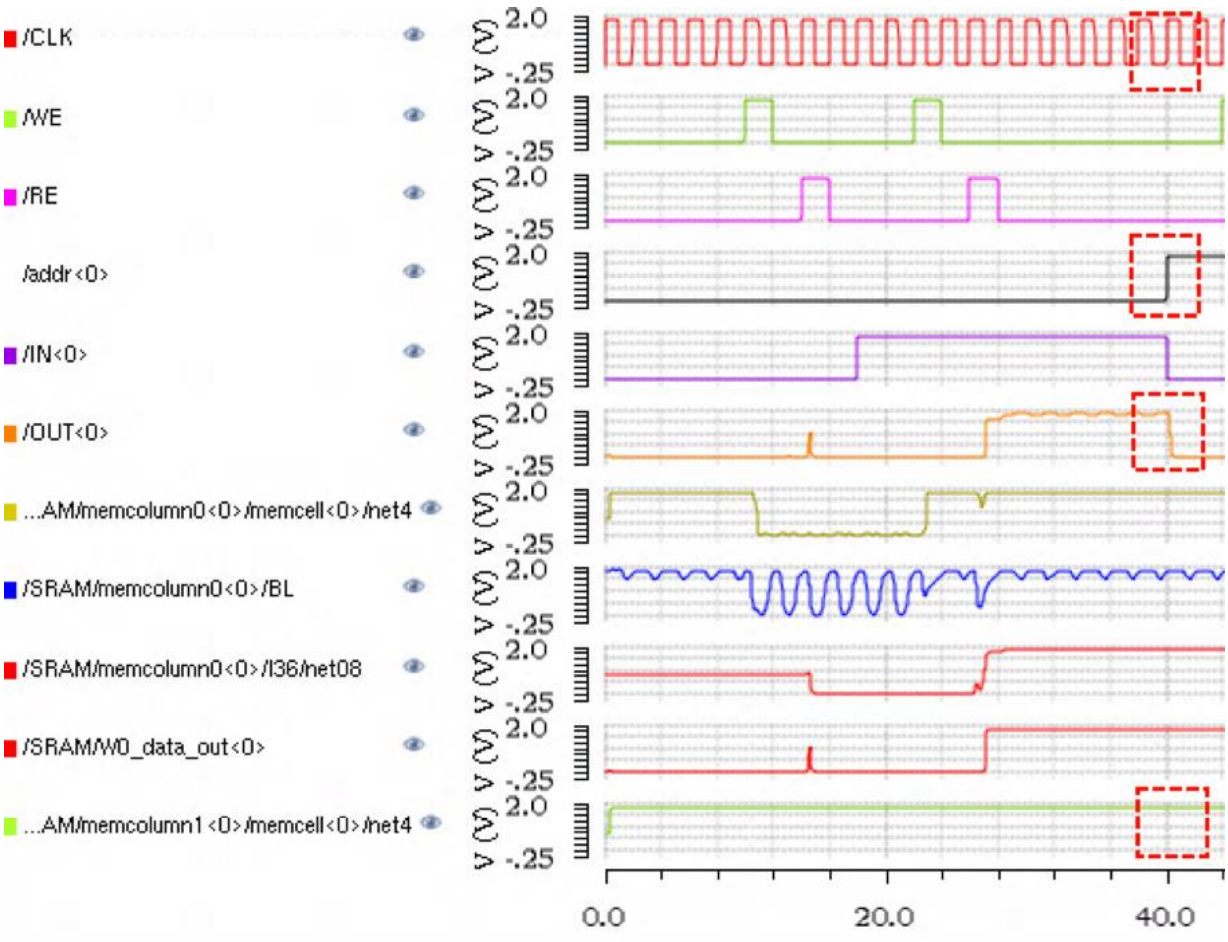


Figure 16: Pre-Layout Output Change without Read Circuit High

Test signals of the row decoder schematic are shown below. The row decoder is used to select which row in the memory array should be selected. Together with the column decoder it allows memory to be addressed in matrix fashion. This allows us to control all 1024 memory cells with only a few wires. Due to its large size only select input/outputs are shown. There are 32 outputs, 5 inputs, BL, $\underline{BL}$, and a clock line. In Figure 22 row address 1 is logical high (1.8V). At the five inputs we have the logical states 0b00010. In decimal this is 2, therefore the 2nd output (grey) is selected. In Figure 23 row address 2 is high, there the binary number at its input is 0b00100. In base ten 0b00100 is 6, therefore output 6 (dark orange) is high.

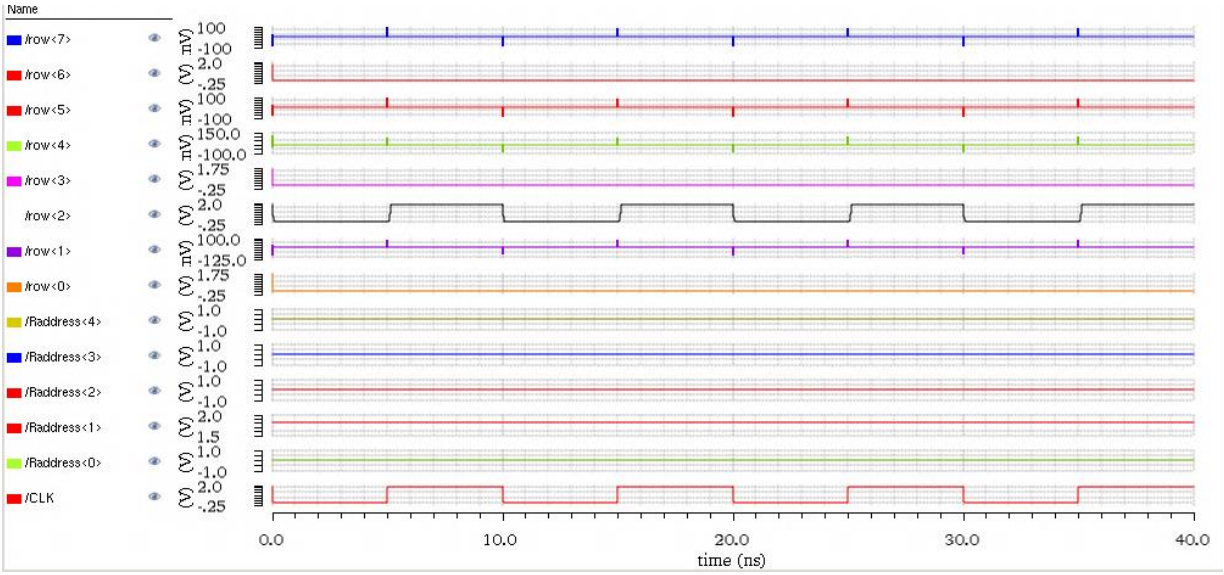


Figure 17: Pre-layout Row Decoder with Row Address 1 High.

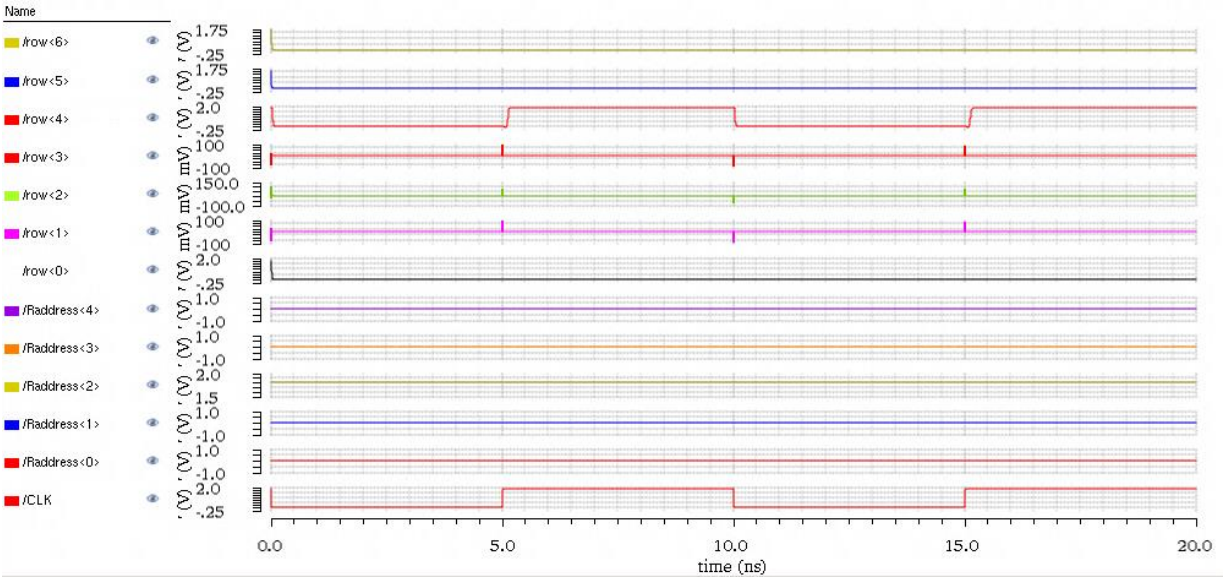


Figure 18: Pre-layout Row Decoder with Row Address 2 High.

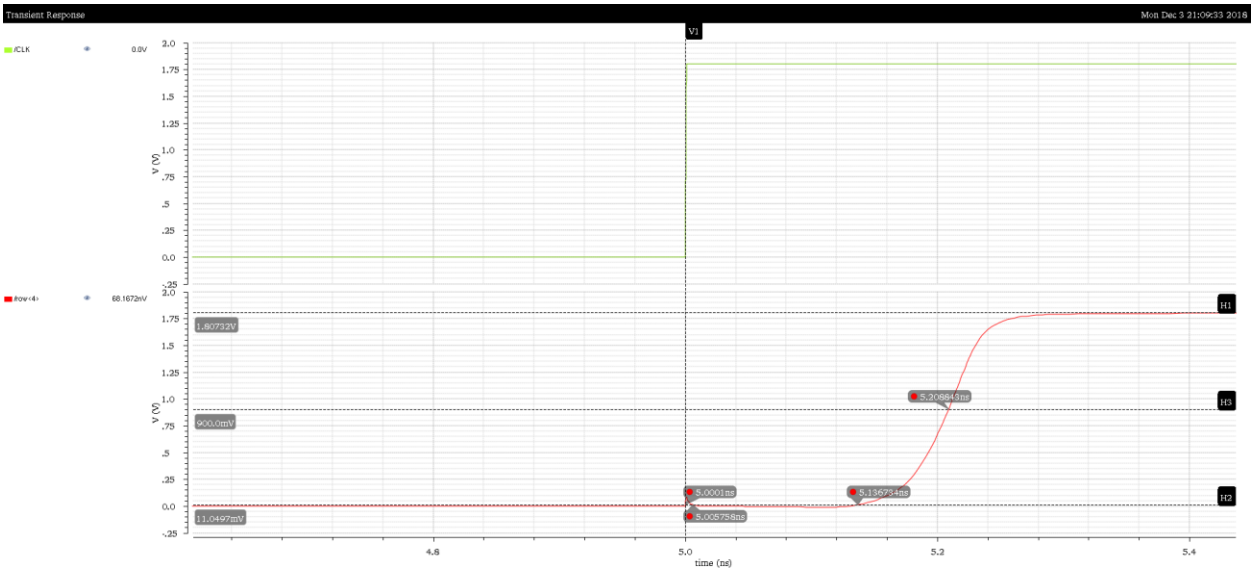


Figure 19: Pre-layout Row Decoder Propagation Delay with Row Address 2 High Propagation Delay = 0.208743ns

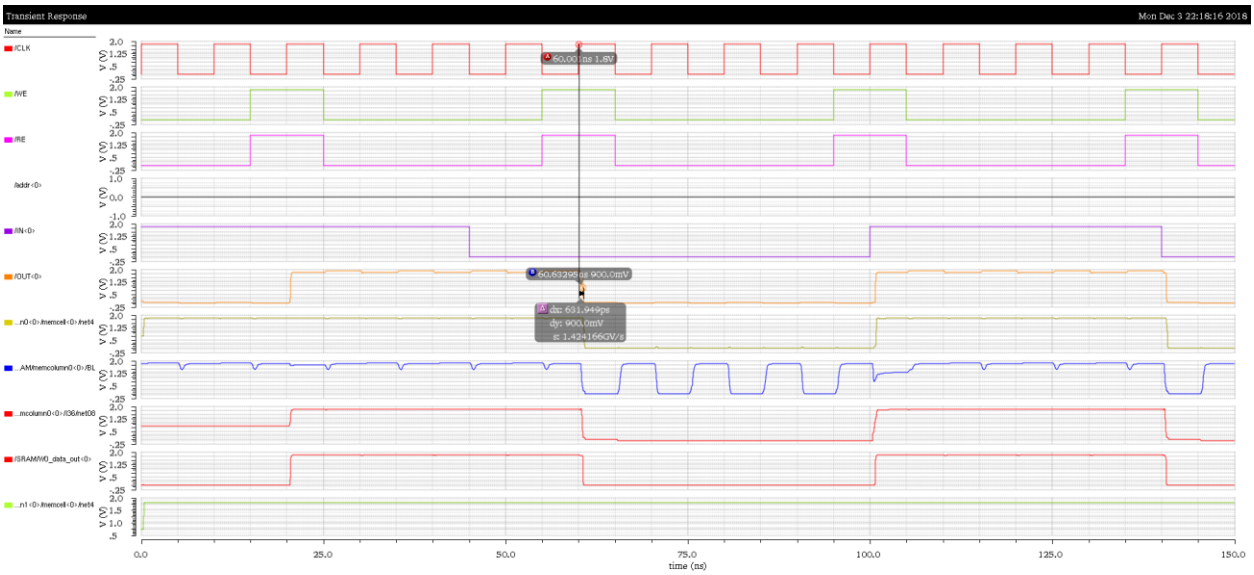


Figure 20: Propagation Delay of SRAM clock in schematic.

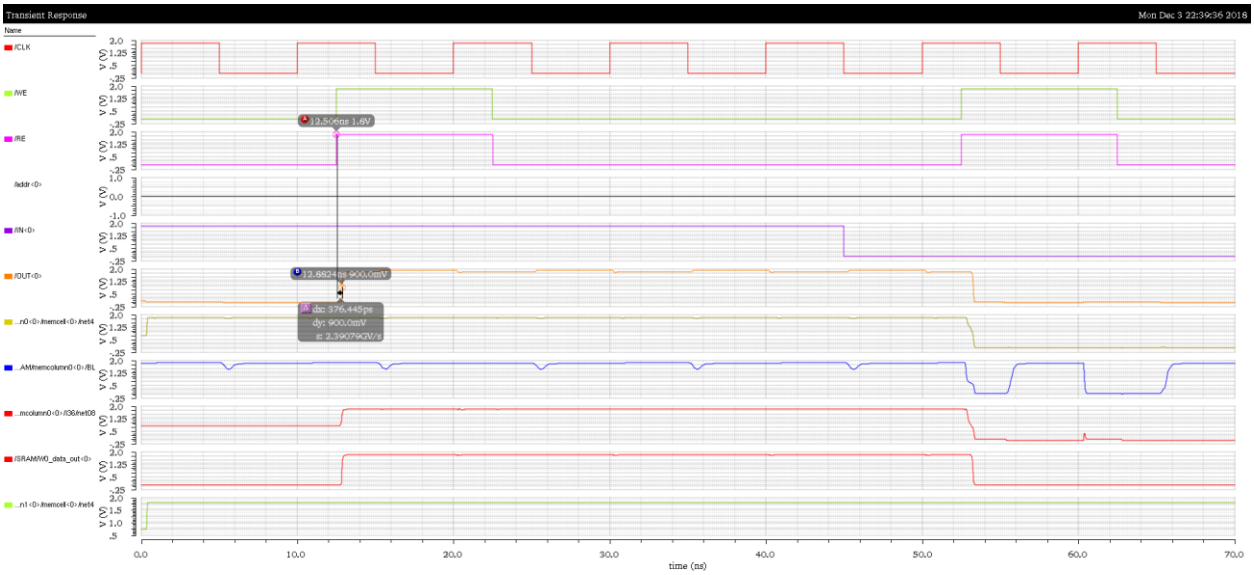


Figure 21: Propagation Delay of SRAM Reading in schematic.

Since the designed SRAM works as a latch, it will activate on a high clock. The propagation delay from clock is 631.95 ps as shown in figure 20, while the propagation delay from reading circuitry is 376.445 ps as shown in figure 21.

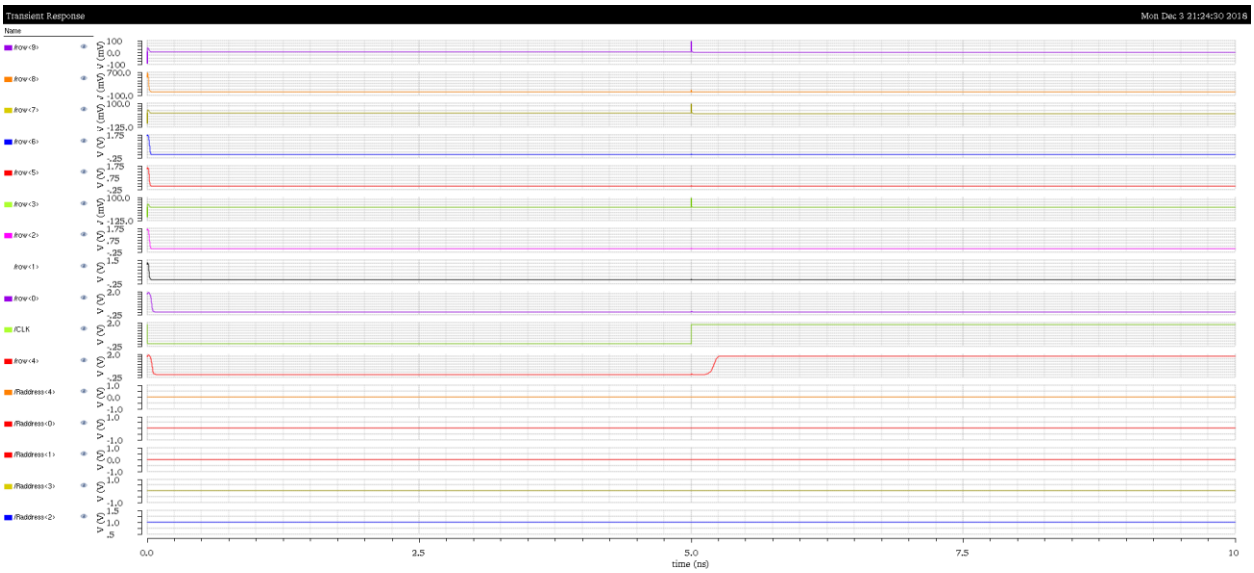


Figure 22: Post-layout Row Decoder with Row Address 2 High.

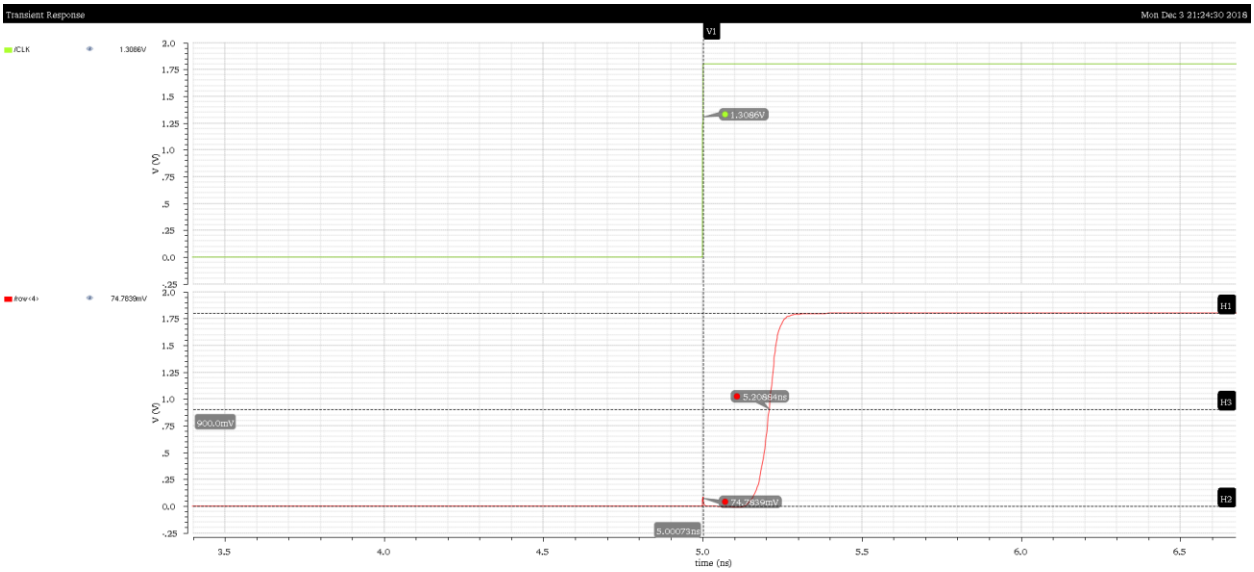
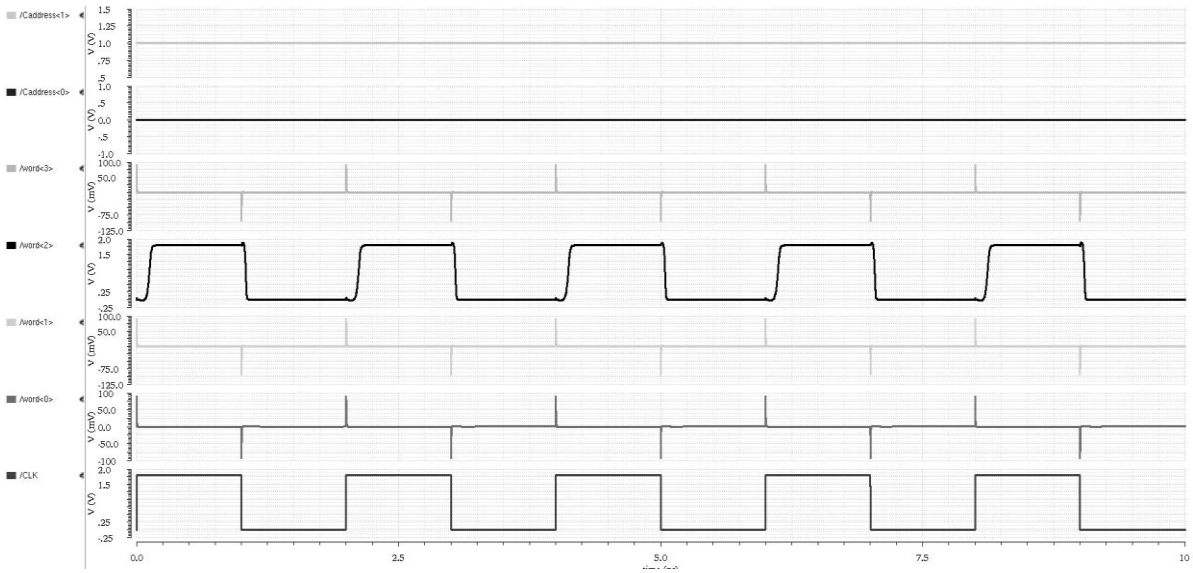


Figure 23: Post-layout Row Decoder Propagation Delay with Row Address 2 High Propagation Delay = 0.208843ns

The Column Decoders' purpose is to output the value from the multiplexer address chosen. Figure 27 displays its full simulation.The two figures below indicate how it works. Figure 28 shows address $(10)_2$ being accessed allowing multiplexer address 2 to go high at the rising edge of the clock.



-/CLK = Clock
-/Caddress<0> = Binary value in place 0
-/Caddress<1> = Binary value in place 1
-/word<0> = multiplexer value in place 0
-/word<1> = multiplexer value in place 1
-/word<2> = multiplexer value in place 2
-/word<3> = multiplexer value in place 3

Figure 24: Pre-Layout Column Decoder Full Simulation

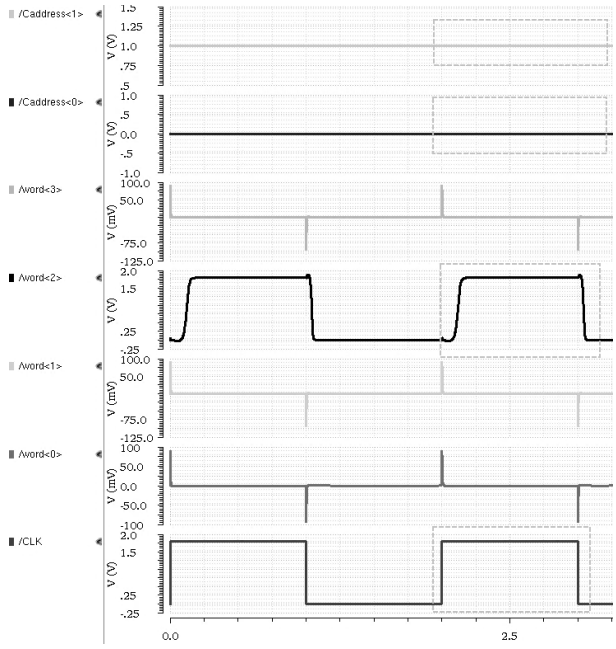


Figure 25: Pre-Layout Column Decoder with Address $(10)_2$

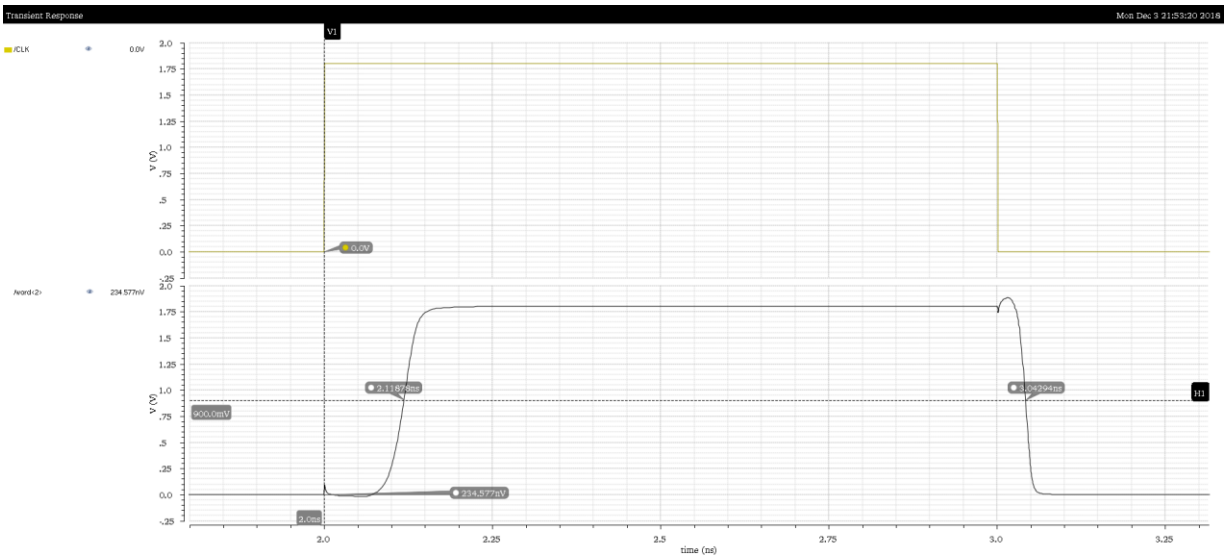


Figure 26: Pre-Layout Column Decoder with Address $(10)_2$ Propagation Delay

Propagation Delay = 0.08086ns

The purpose of the word 4-1 mux is to pass the value to the determined memory column based on the signal received from the column decoder.

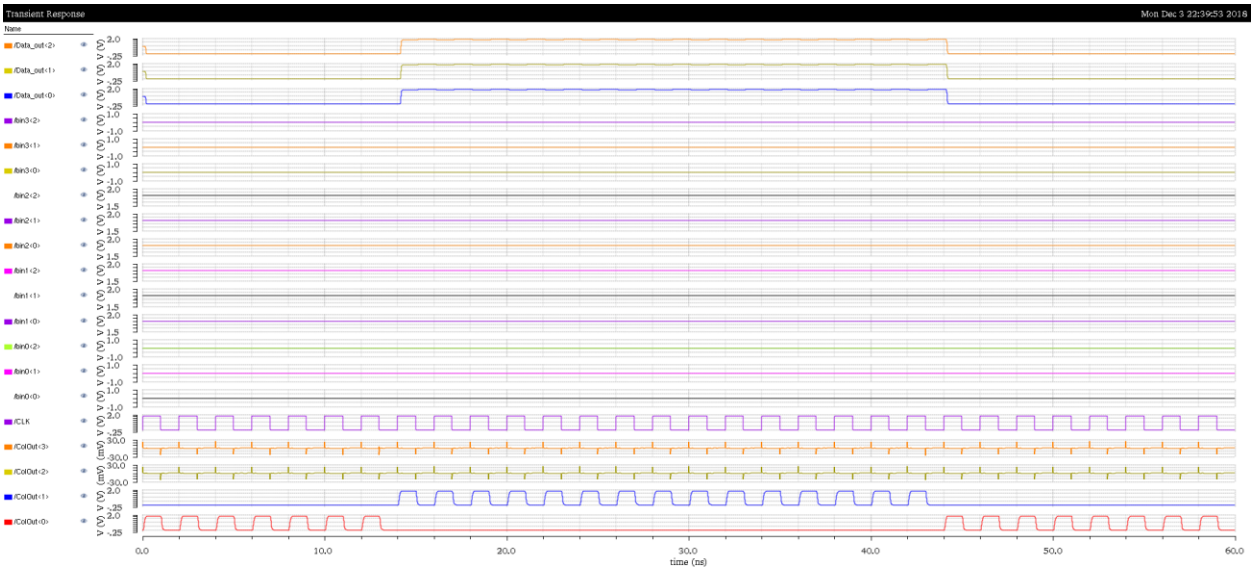


Figure 27: Pre-Layout 4 to 1 word mux (The 2nd word was selected)

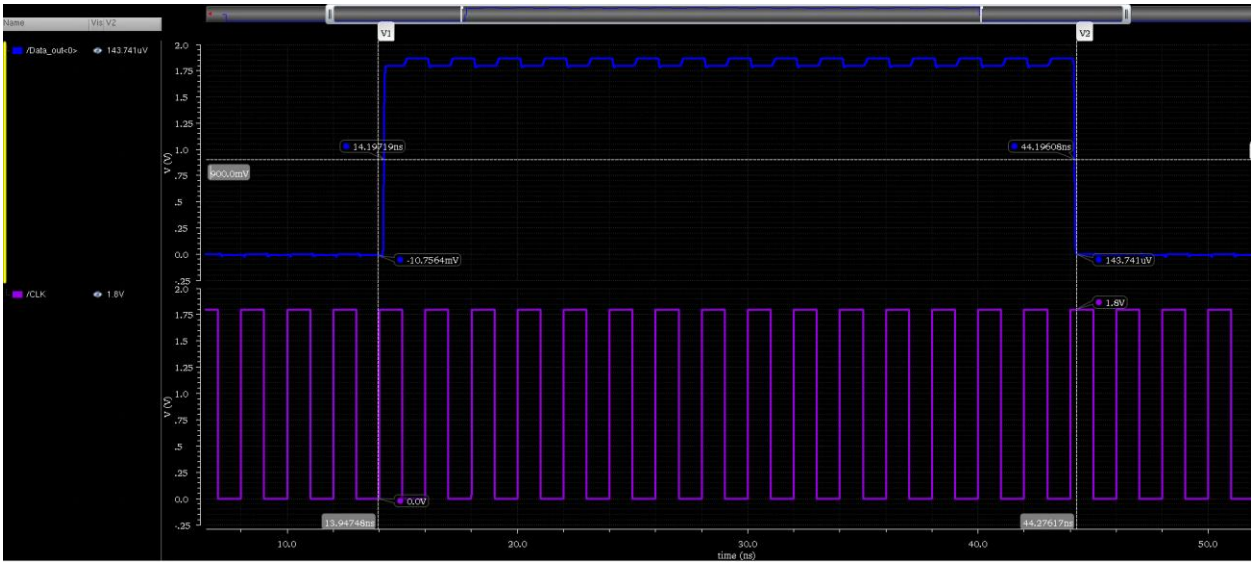


Figure 28: Pre-Layout 4 to 1 word mux (The 2nd word was selected) Propagation Delay

Figure 29: Pre-Layout SRAM Power Reading

Layout Design

Layout is the process of creating a physical circuit from circuit diagrams. The software tool we are using for this task is called Cadence. Cadence is an Electronic Design Automation (EDA) suit. The schematic and layout views are linked, once a circuit diagram has been constructed Cadence assists in the routing of connections called traces. The figures below display the final layout design for each circuit.

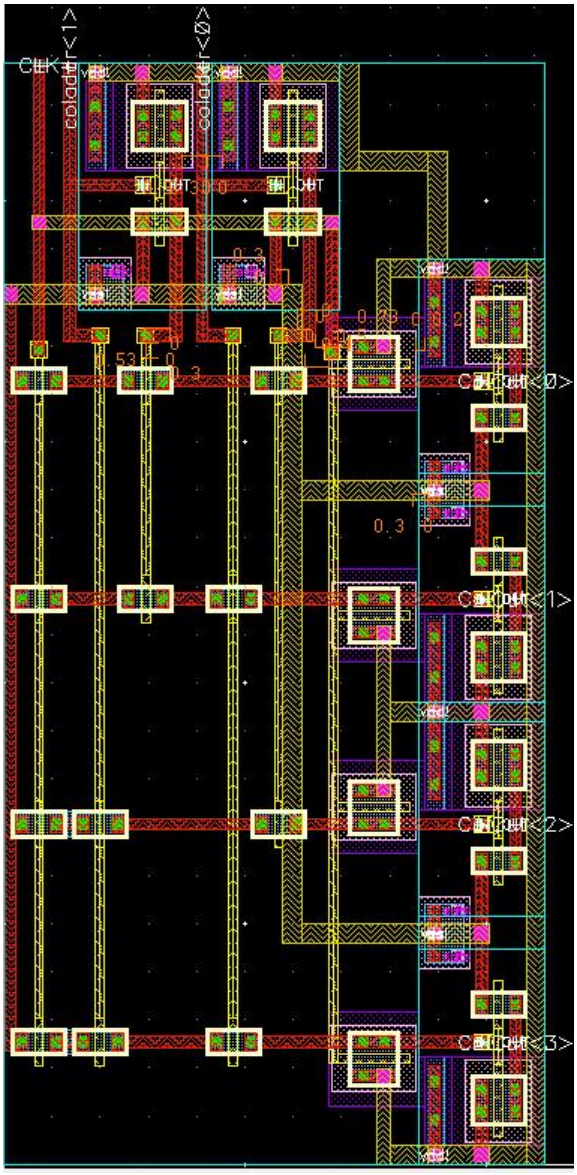


Figure 30: Column Decoder Layout

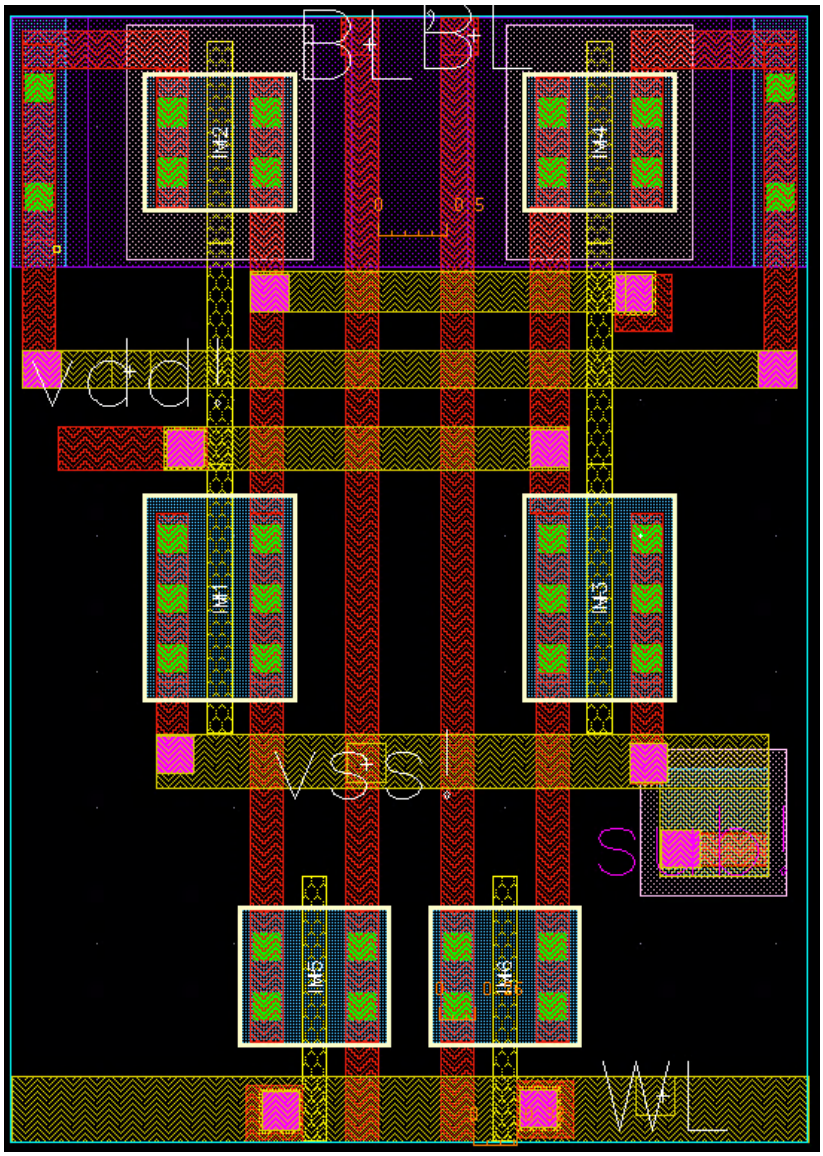


Figure 31: Memory Cell Layout

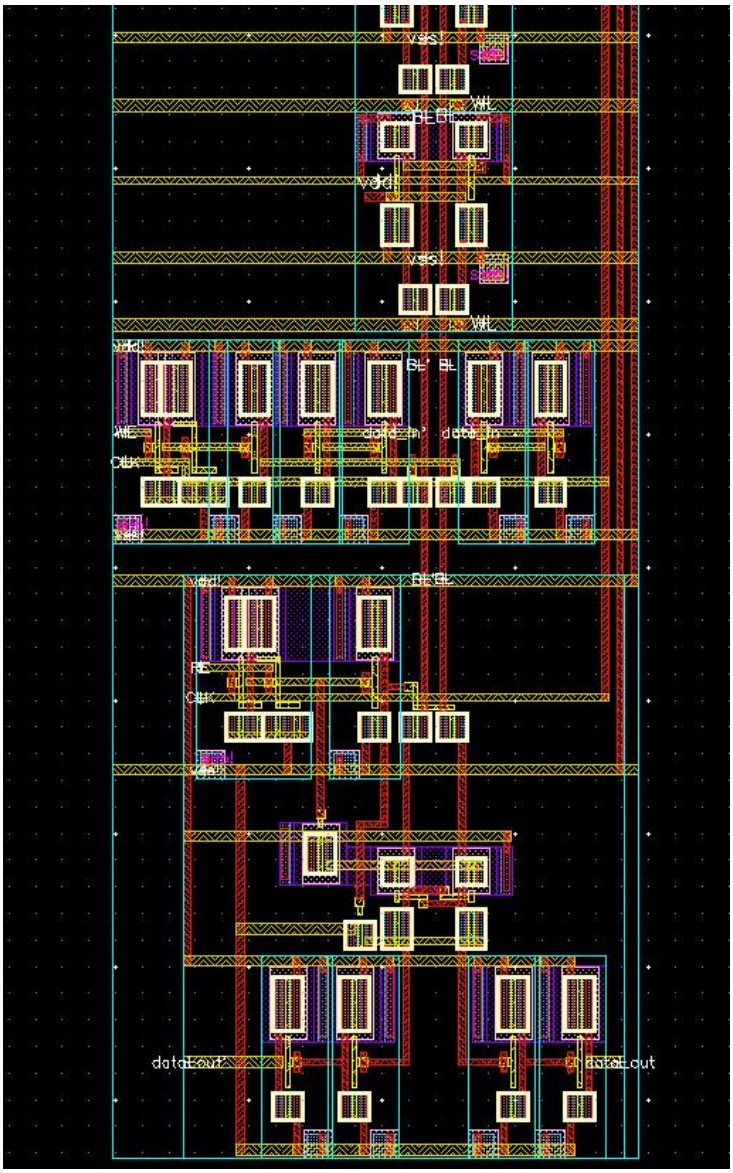


Figure 32 : Bottom part of Memory Column

Figure 35 shows the Precharge and Memory Cell 0 connection. This two parts makes up the top of the memory cell. The middle of the memory column consists of back to back memory cells. This is the case until the memory cell 31. Memory Cell 31 is connected to the read and write circuit as seen below in Figure 34.

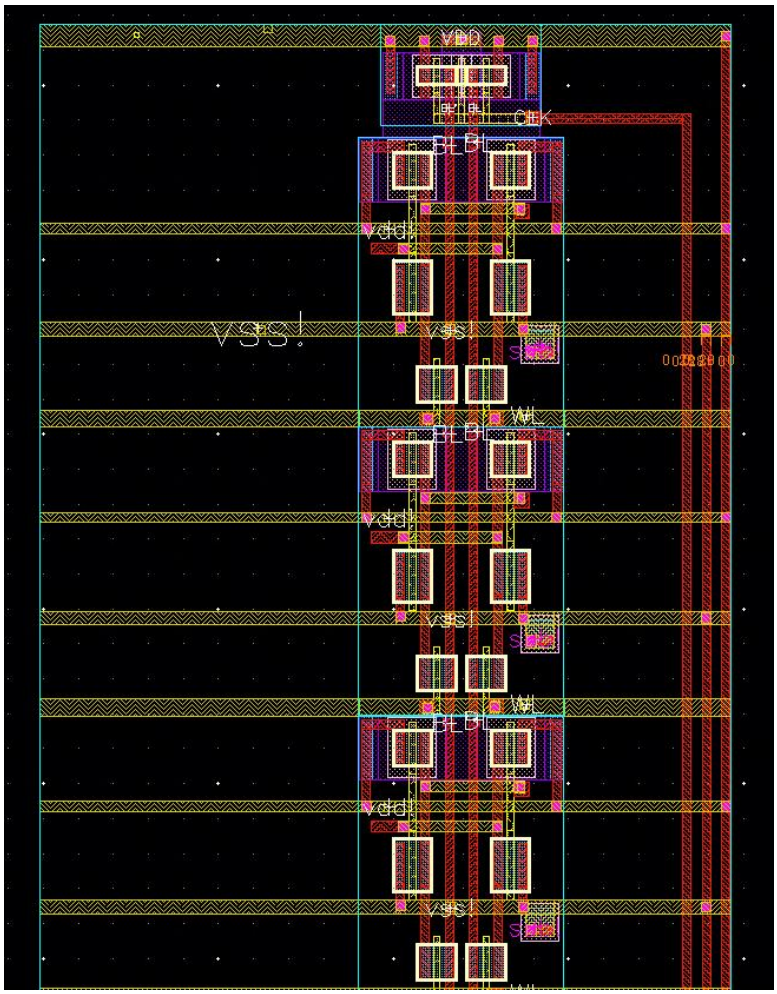


Figure 33: Top Part of Memory Column

Figure 34: Memory Column

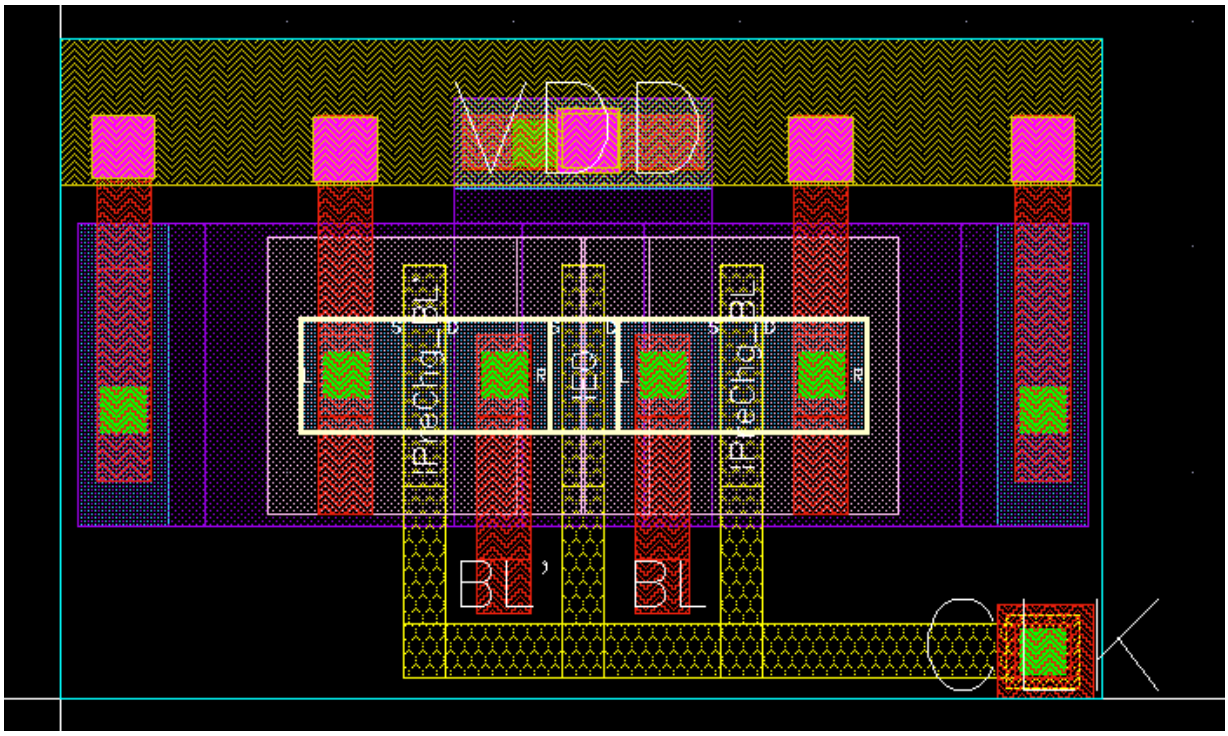


Figure 35: Precharge Layout

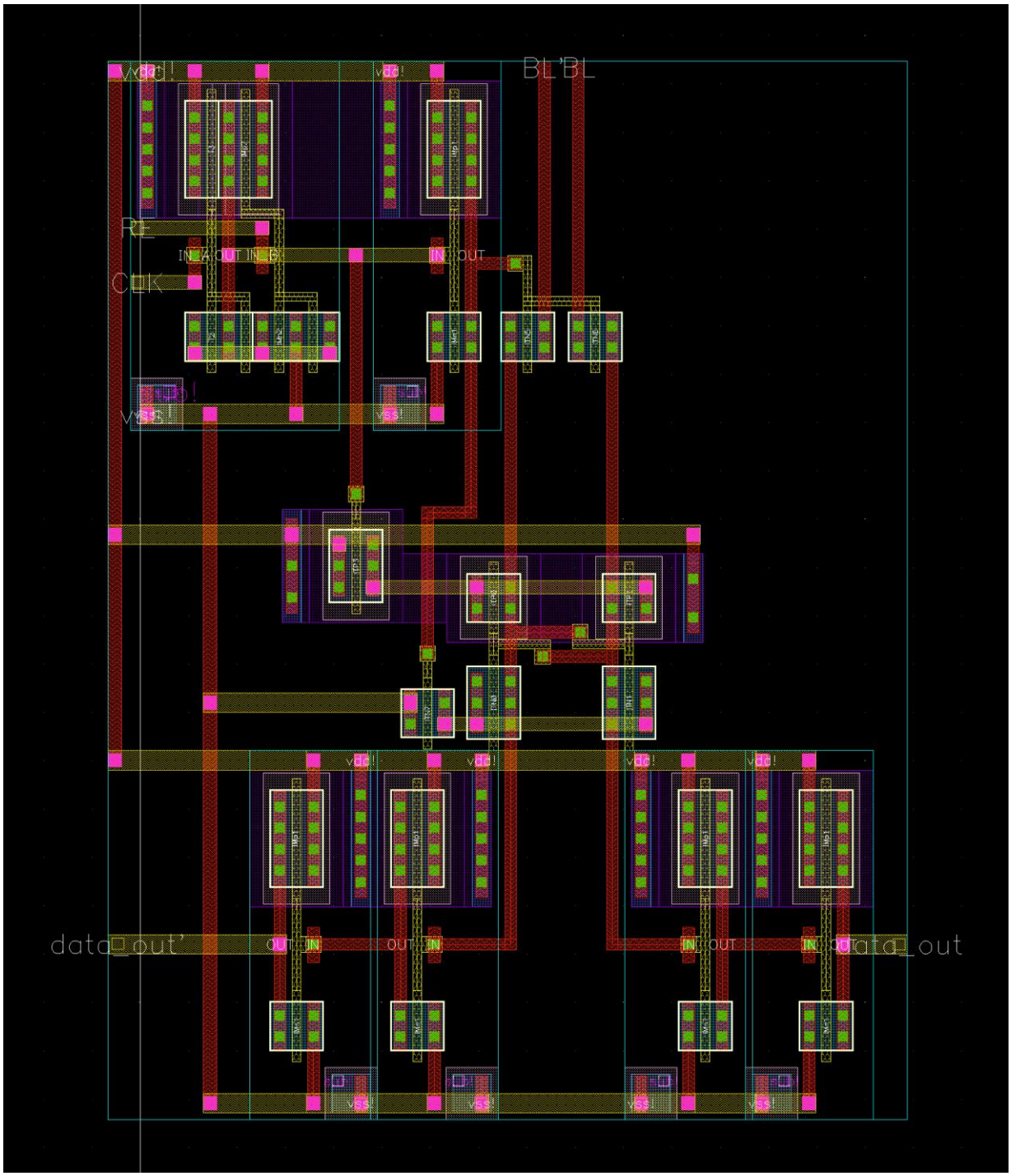


Figure 36: Read Circuit Layout

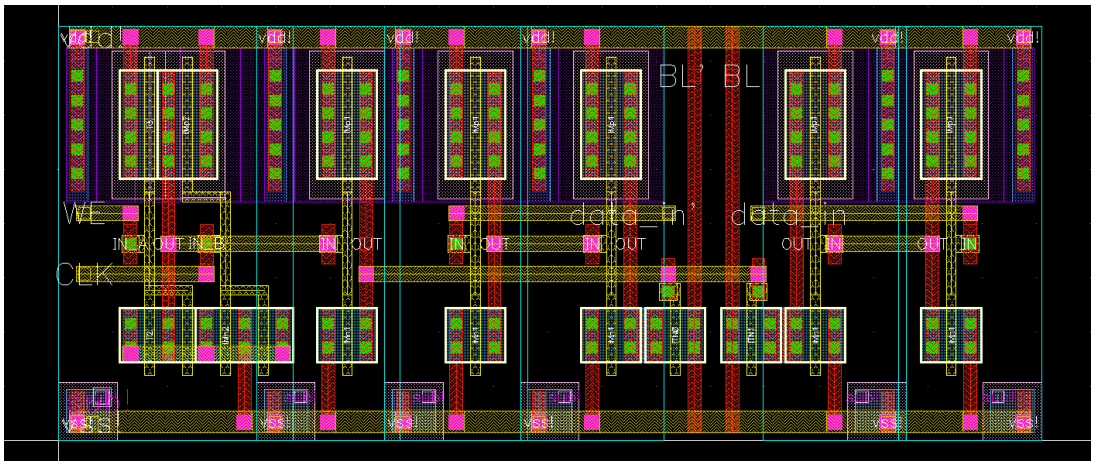


Figure 37: Write Circuit Layout

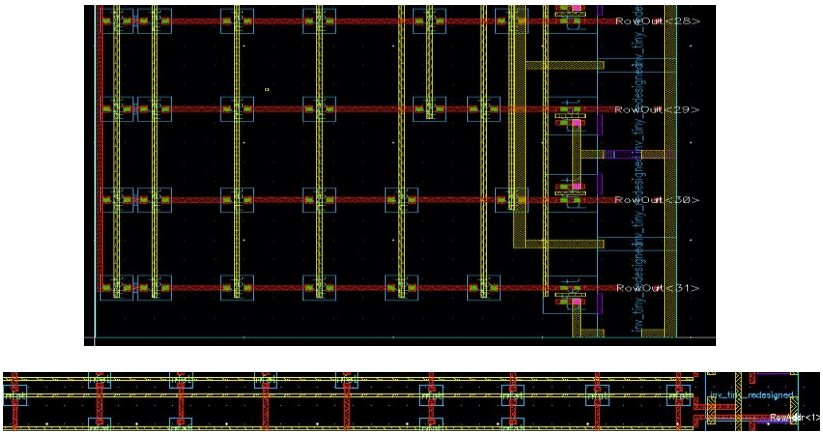


Figure 38: Row Decoder Layout

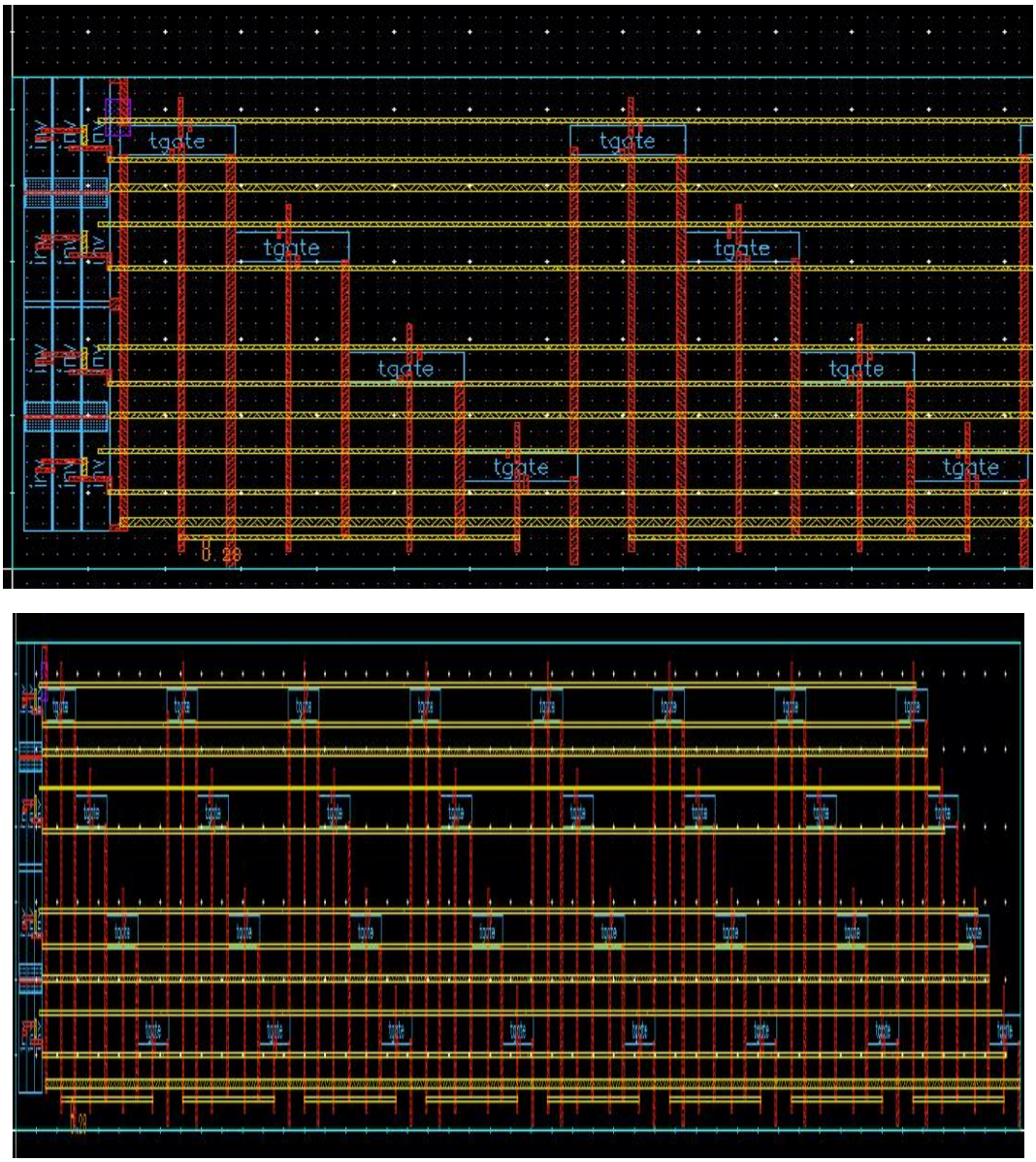


Figure 39: Word Mux Layout

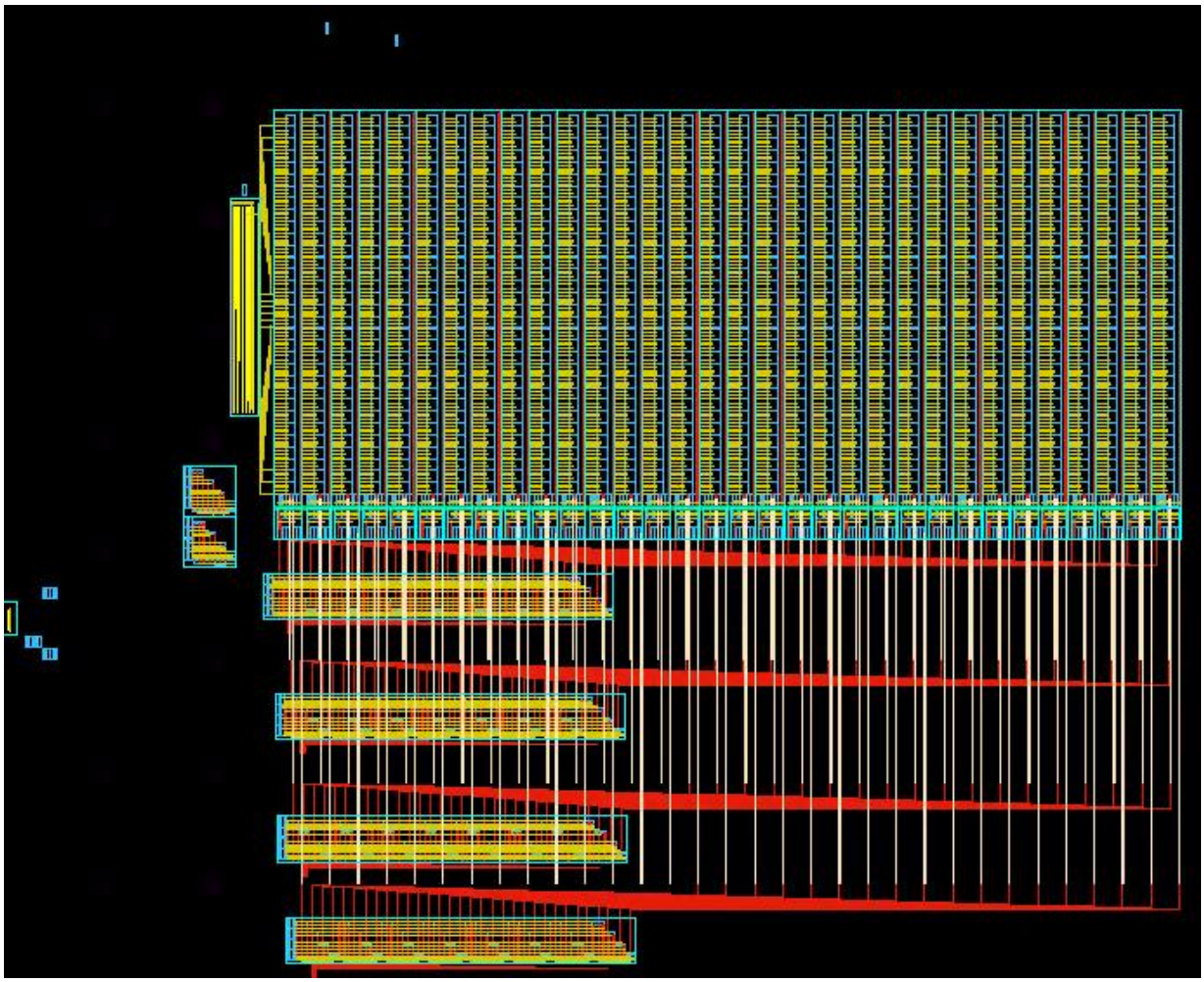


Figure 40: Complete Layout

Post-Layout Simulation Results

The row decoder simulation shows the propagation delay in layout. Figure 32 and 33 show the difference between the theoretical delay and the layout delay.

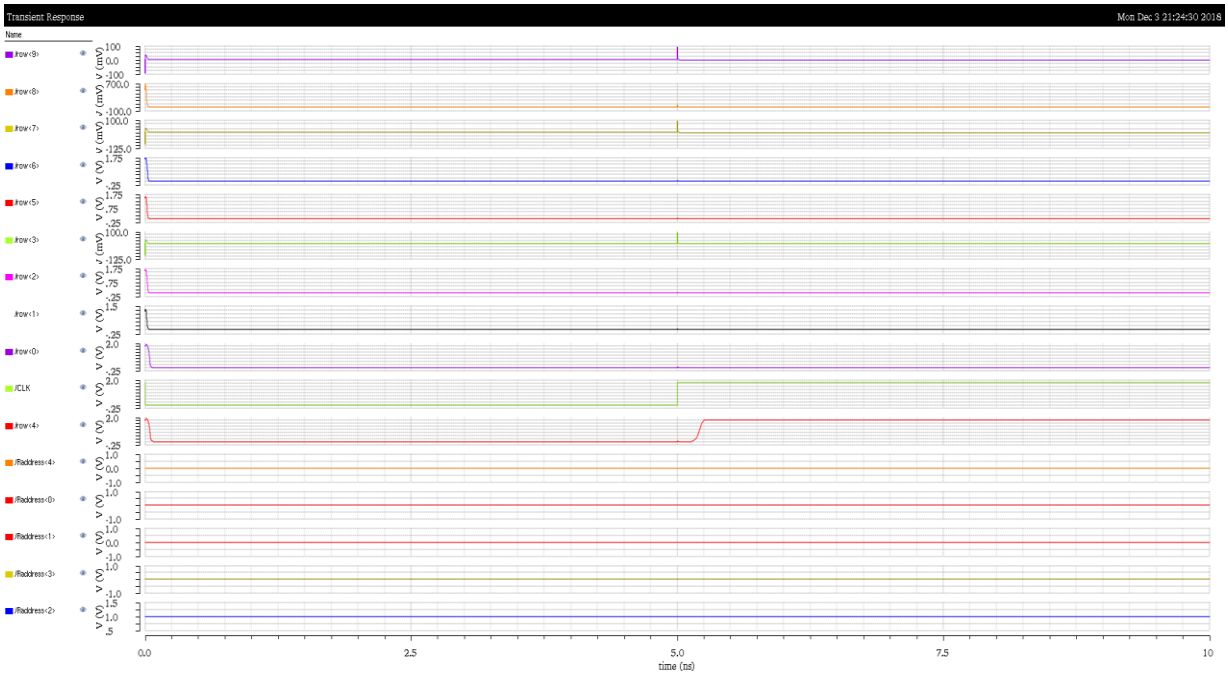


Figure 41: Row Decoder Full Simulation

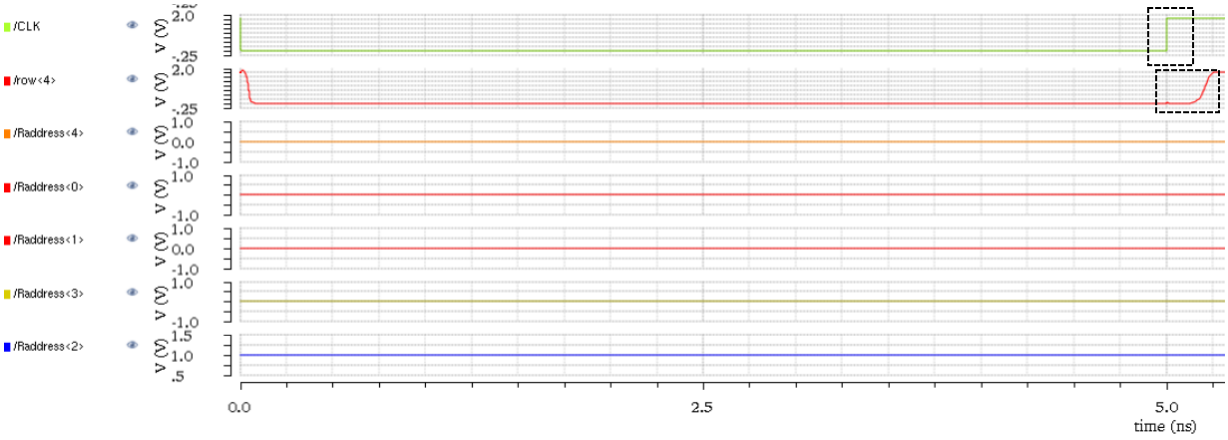


Figure 42: Row Decoder $(0100)_2$

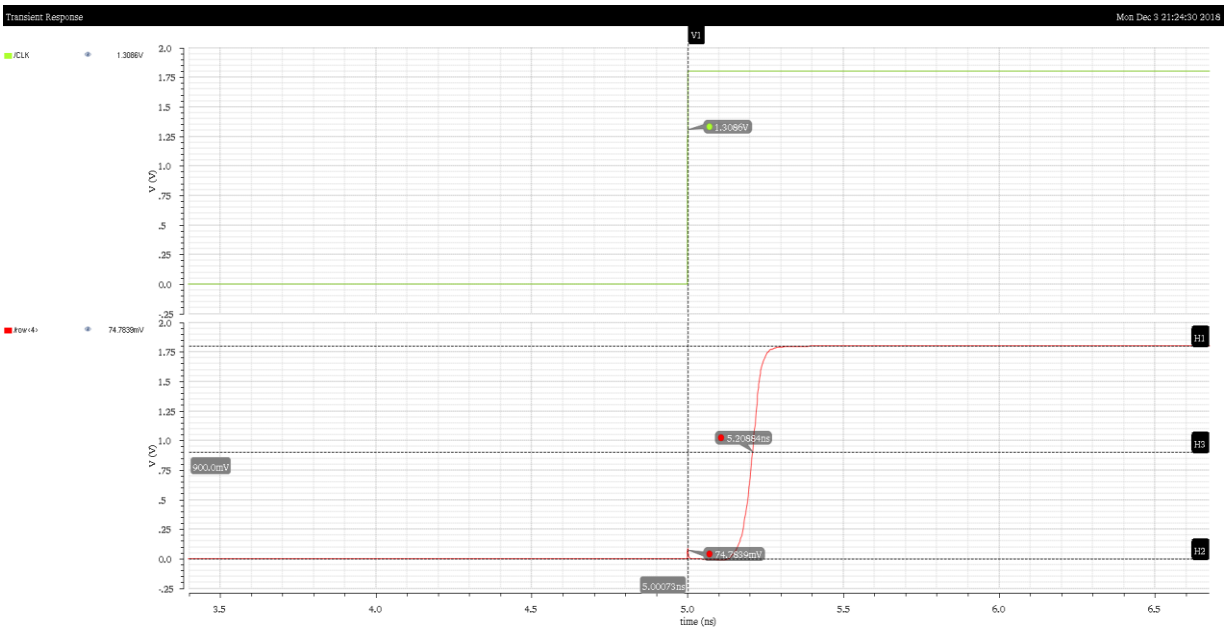


Figure 43: Row Decoder Theoretical Delay $(0100)_2$

Propagation Delay = 0.20811 ns

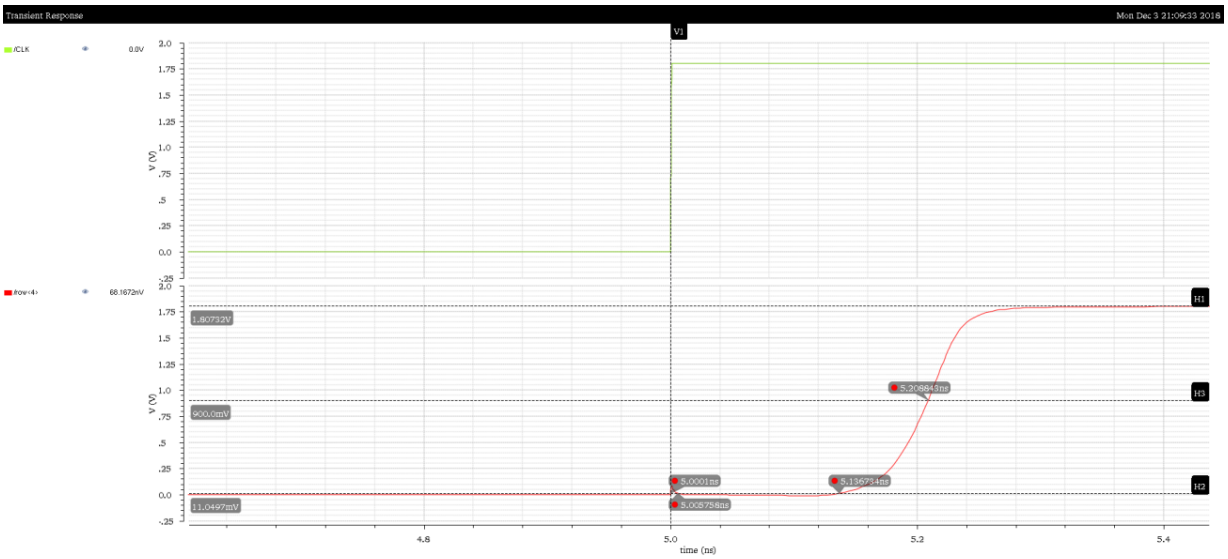


Figure 44: Row Decoder Layout Delay $(0100)_2$

Propagation Delay = 0.208743 ns

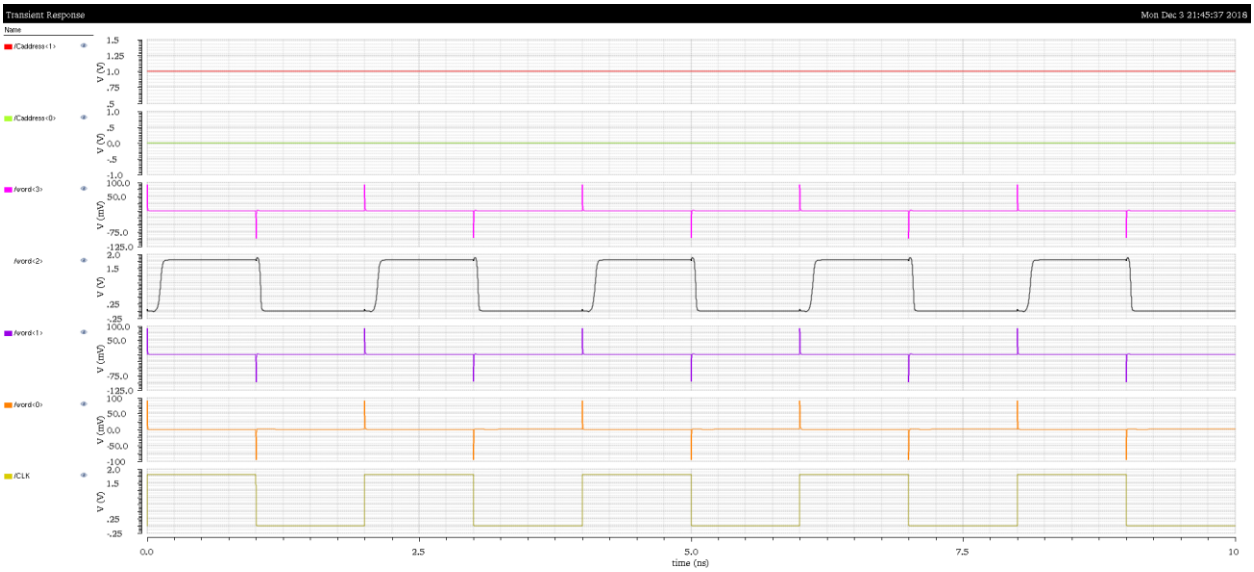


Figure 45: Post-Layout Column Decoder with Address $(10)_2$

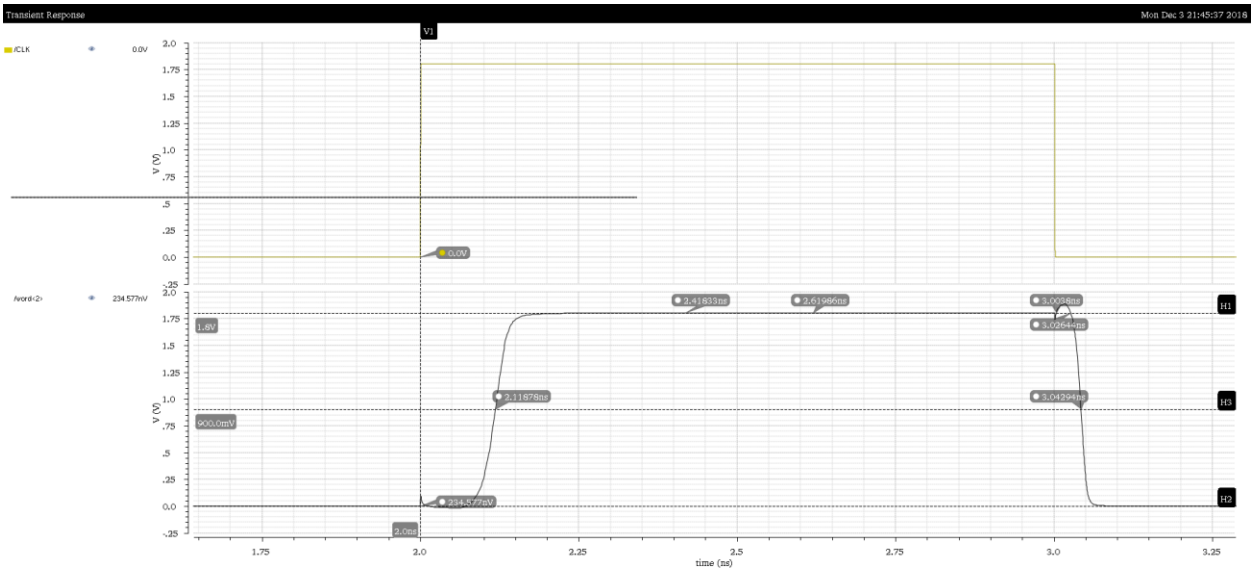


Figure 46: Post-Layout Column Decoder with Address $(10)_2$ Propagation Delay

Propagation Delay = 0.08086ns

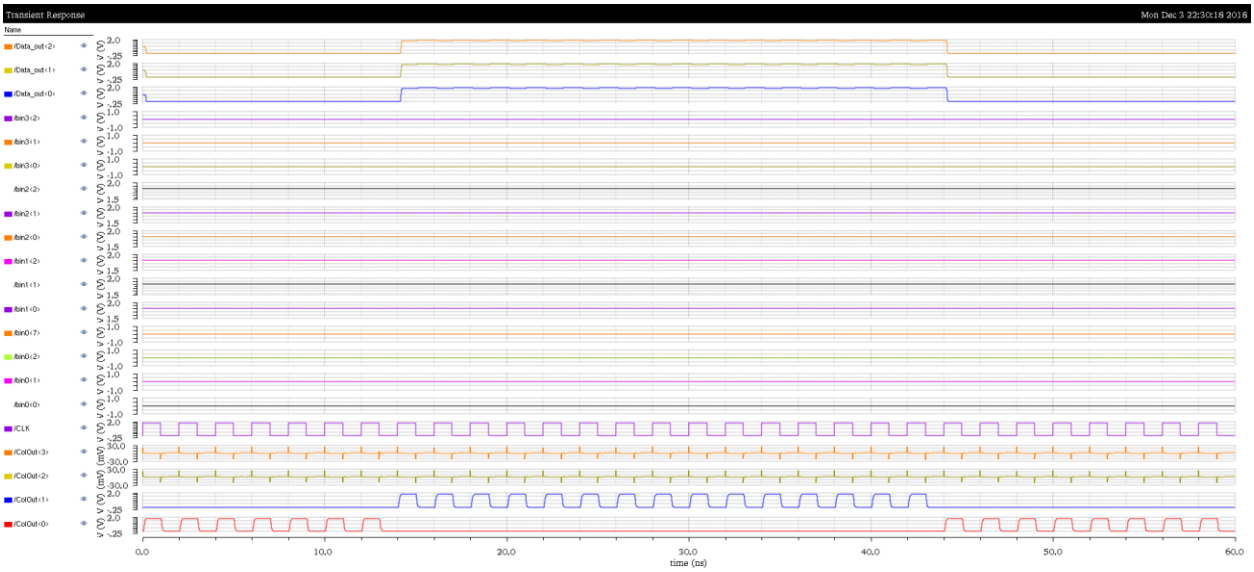


Figure 47: Post Layout: Word 4-1 mux Testbench

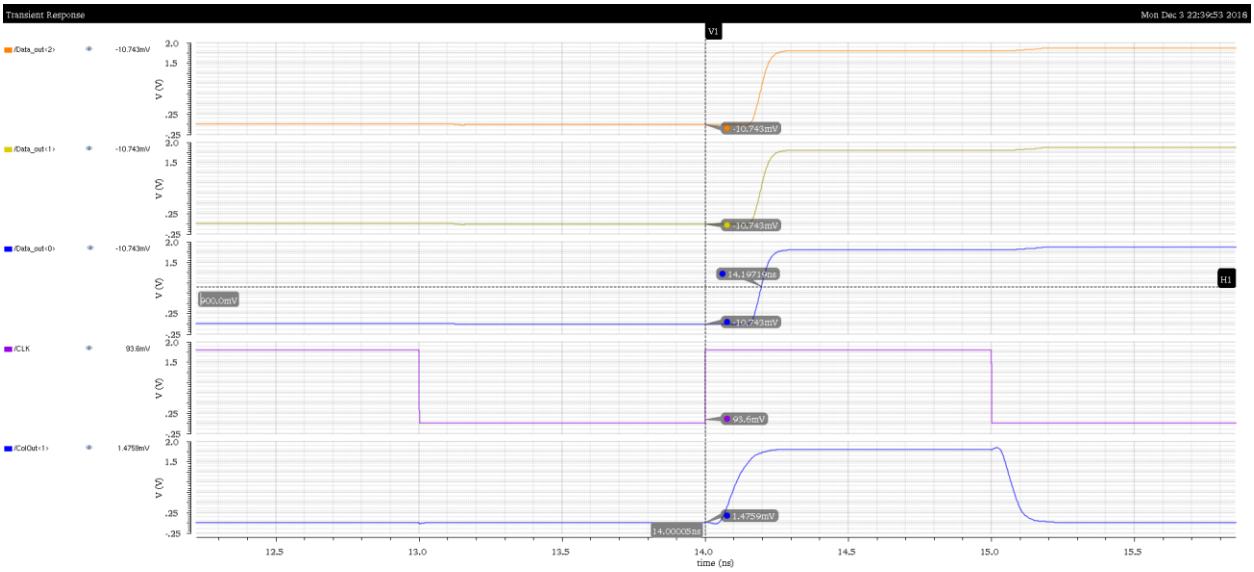


Figure 48: Post Layout 4 - 1 Mux delay

Delay = 0.1971ns

Conclusion

Despite the relative simplicity of such a device compared to commercial products learning its construction proved technically challenging. There are a few takeaways from this project. Planning is of utmost importance. The team underestimated the difficulty in successfully splitting up tasks and combining them into one. Another item that the team did not account for was the conflicting time in the team's schedule. In Conclusion, the team is satisfied with the experience gained from designing a 1024 bit Static RAM from scratch.

Works Cited

1. https://stackoverflow.com/questions/1371400/how-much-faster-is-the-memory-usually-than-the-disk
2. http://www.extremetech.com/computing/192711-samsungs-new-20nm-ddr4-clears-the-way-for-massive-128gb-dimms

Acknowledgments

Joseph Chiocca	Wrote the background section, ran pre-layout simulations for the row select driver and pre-layout read circuit, offered suggestions and help during layout, added various commentary throughout the document and contributed to various editing.
Matthew Wiechec	Designed some of the various schematics and layouts, including the row and column decoders, and part of the full layout.
Youngsoo	Designed, simulated, and constructed the majority of the memory module. He also guided team members into their roles and helped establish order through the project. He also resolved Linux issues involving file permissions.
Lam	Did simulation and layout of various pieces. When he was not working in Cadence he was exporting data from Cadence into the group write-up and helping to flesh out the body of the report.
Wen Hong (Derick) Lew	Designed on the layout of various pieces in the integrated circuit, ran some of the simulations, and did some the calculations.
Sami	Wrote much of the documentation and assembly for the final report, assembled most the graphs and figures from Cadence, and wrote commentary about them.