8-Bit CPU Final Report

Malik Hubbard
Maxfield Parson-Scherban

VLSI

15 December 2022

RISCY - Final Report

The goal of this project was to design, layout, and verify a functioning microcontroller core. **Figure 1** shows the symbol for this core in its testing module. We named our microcontroller RISCY. **Figure 2** shows the microcontroller core circuit. This datapath consists of a PLA for issuing control signals, a 8-byte SRAM memory unit, a 8-bit bus driver, multiple 8-bit latches, a 3-input 8-bit multiplexer, a 2-input 8-bit adder, an 8-bit shifter, and a bypass circuit for bypassing the shifter when appropriate it. The key schematics of the core design are documented below in more detail with particularly slow components receiving a timing analysis.

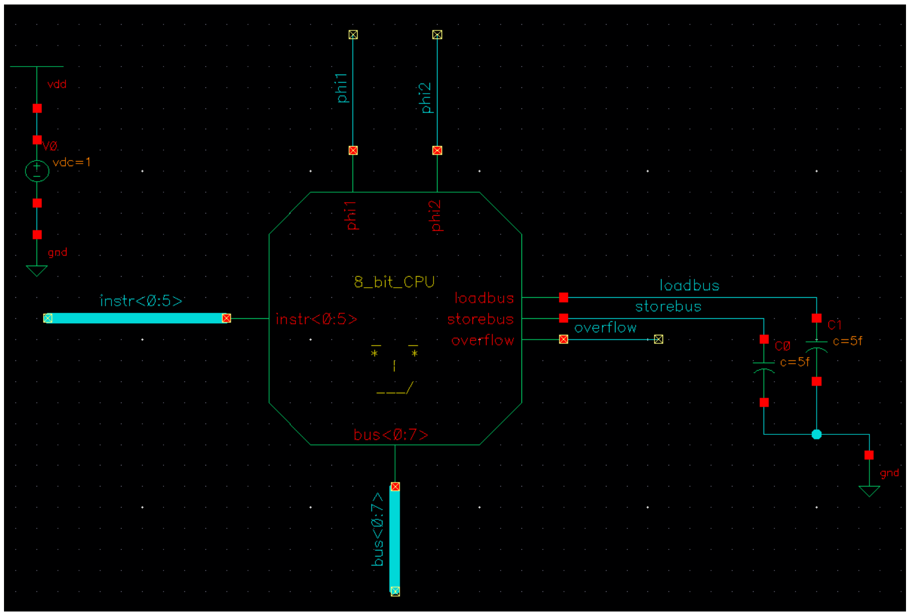


Figure 1: Datapath of core (RISCY)

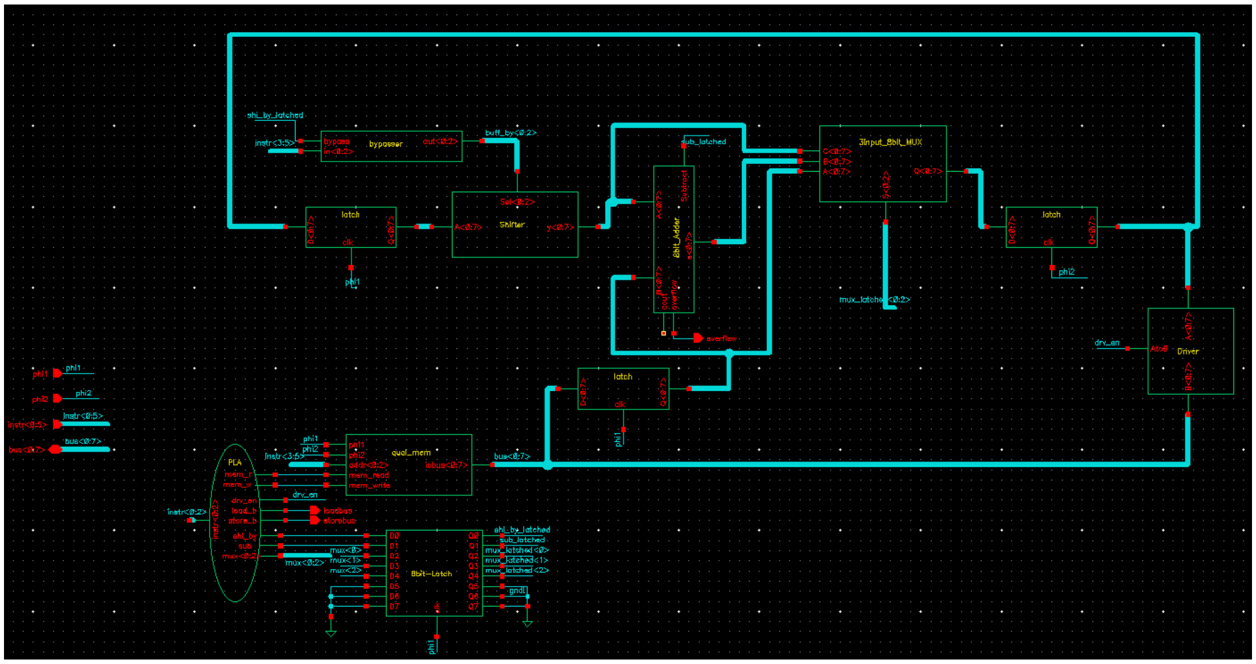
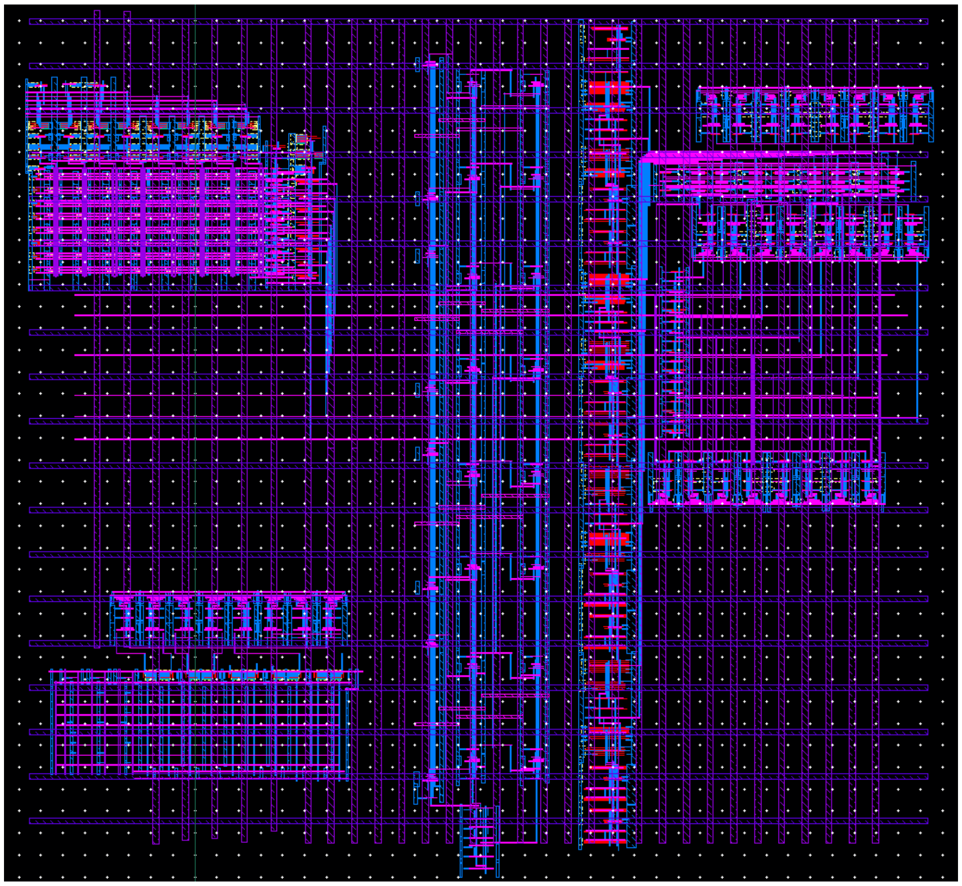


Figure 2: RISCKY's circuit

Layout

The diagram below shows our core's layout. The streamout has also been attached to this submission. Unfortunately, this layout does not have all the connections required for the datapath, so it has not been verified. However, multiple components are connected and the ground and vdd distribution is presented and noticeable.



Functional Schematic

To verify the functionality of our cpu schematic we used many vector sim tests which can be found in the submission files. The most ambitious of which was the following test:

This vector sim loads 2 words to memory, multiplies one of them by 2, and subtracts it from the other word. As can be observed in the waveform below, this simulation was able to be executed with no errors.

LOAD val 254 to address 0
STORE address 0 // check if 254 made it to memory
GET address 0 // not necessary to find result, testing for a potential failure of LOAD after GET
LOAD 64 to address 1 // this is the hex value 40 we want to shift

STORE address 1 // check if 64 made it to memory
GET address 1 // we need 64 in accumulator to prepare for shift
NOP // not necessary, testing if accumulator value remains constant during NOP
SHIFT by 1 // 64 is in accumulator, we expect it to be multiplied by 2 resulting in 128 (80 in hex)
PUT to address 0 // we need our shift result in memory to prepare for subtracting it from 254
GET address 0 // 254 goes to accumulator
SUB address 1 // 254 - 128, expecting 126 or 7E in hex
PUT to address 2 // need to put in memory to prepare for STORE
STORE address 2 // check if result was 126
NOP // check if state of multiple memory values is maintained during NOP
STORE address 1 // check if 80 is still in address 1

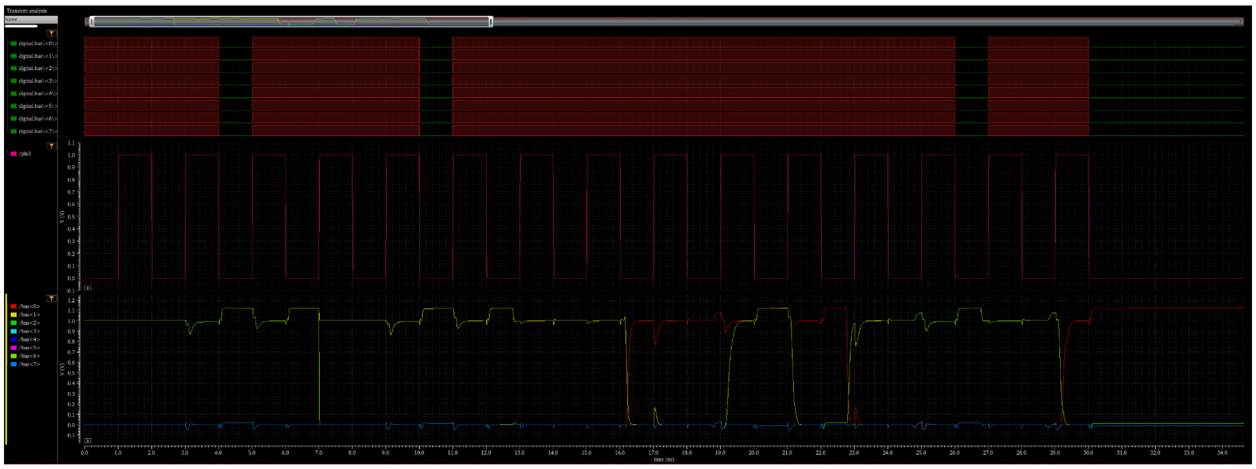


Diagram 1: CPU Schematic Verification Waveform

As visible in **Diagram 1**, no errors were generated and the instructions performed as expected, thus verifying our CPU schematic at 500 MHz.

2-Input 8-Bit Adder

Figure 3 depicts the adder circuit. 8 FA adders are connected in series with the *Subtract* pin being the first carry in. Each *A* input to each full adder is inverted every other full adder, while each *B* input is NORed with the *Subtract* signal. The *overflow* signal is computed by observing if the final two carries are the same using a NOR gate. **Figure 4** shows the layout for 1-bit of computation for the adder. This layout is LVS and DRC clean. The adder is 187.4 um long.

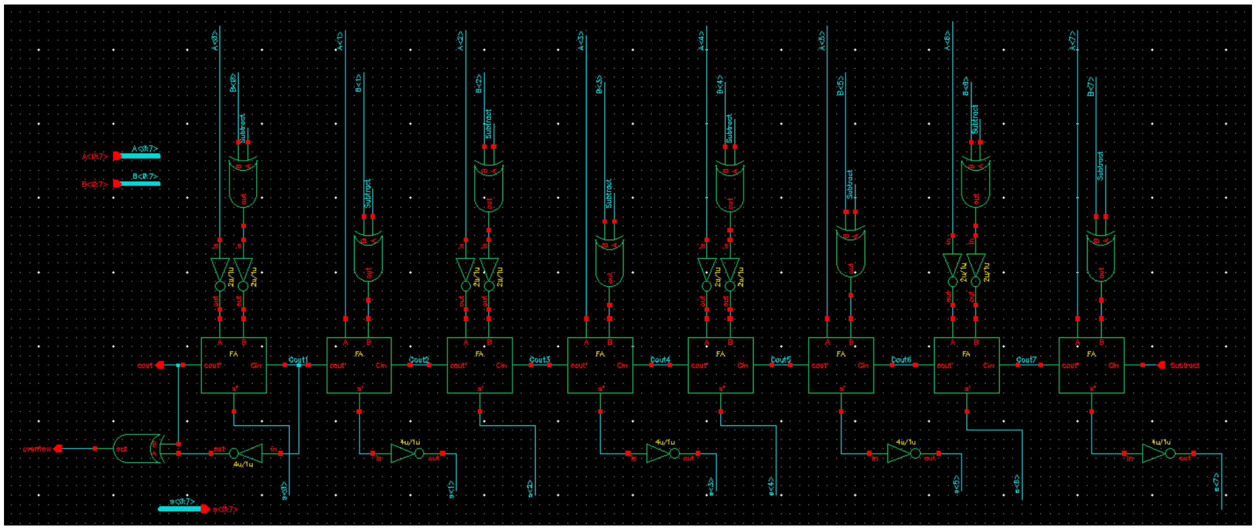


Diagram 3: Adder Circuit

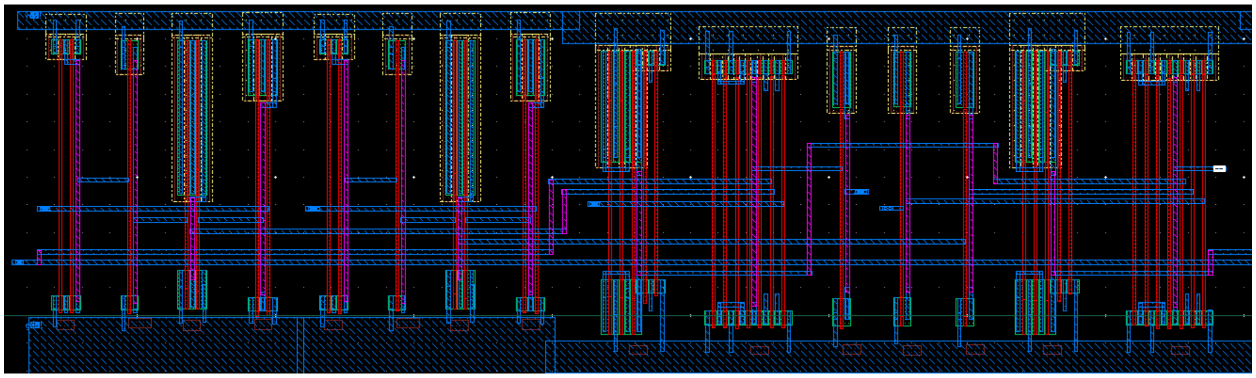


Figure 4: 1-bit of Adder Computation

Figure 5 shows the schematic for the testing circuit. The adder layout was extracted and then tested yielding the waveform of **Diagram 1**. Here we test the critical path of the circuit, by adding 255 to 1, leading to a carry through the entire circuit. We observe a delay of 2.7 ns for our final carry out signal on our extracted circuit.

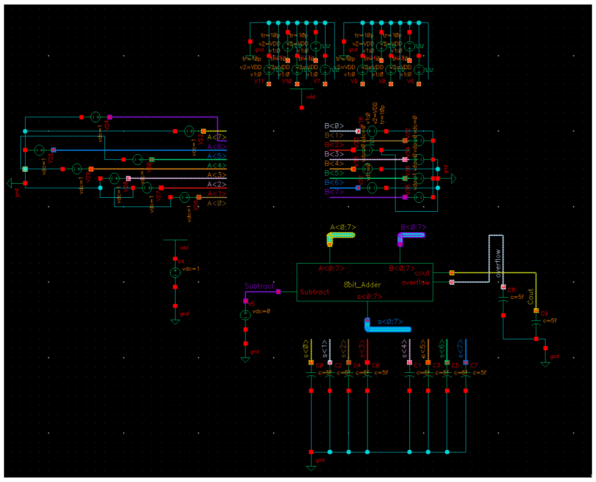


Figure 5: Testing Schematic for Adder

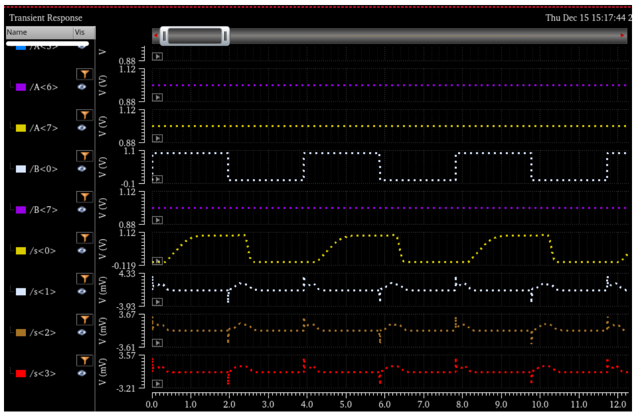


Diagram 1: Critical Path Test of 8-bit Adder

8-bit Shifter

Figure 6 shows the schematic for our logarithmic shifter circuit. Our shifter takes in 8 bits of input, *A<0:7>* and shifts based upon the *Sel<0:2>* bits. These selection bits tell the shifter to shift from 0 to 7. The shifter is composed of 24 2-to-1 1-bit multiplexers. These multiplexers are implemented using complementary pass transistors. The LVS and DRC clean layout for our shifter is shown in **Figure 7**. Importantly, the bit pitch between the shifter and adder is maintained.

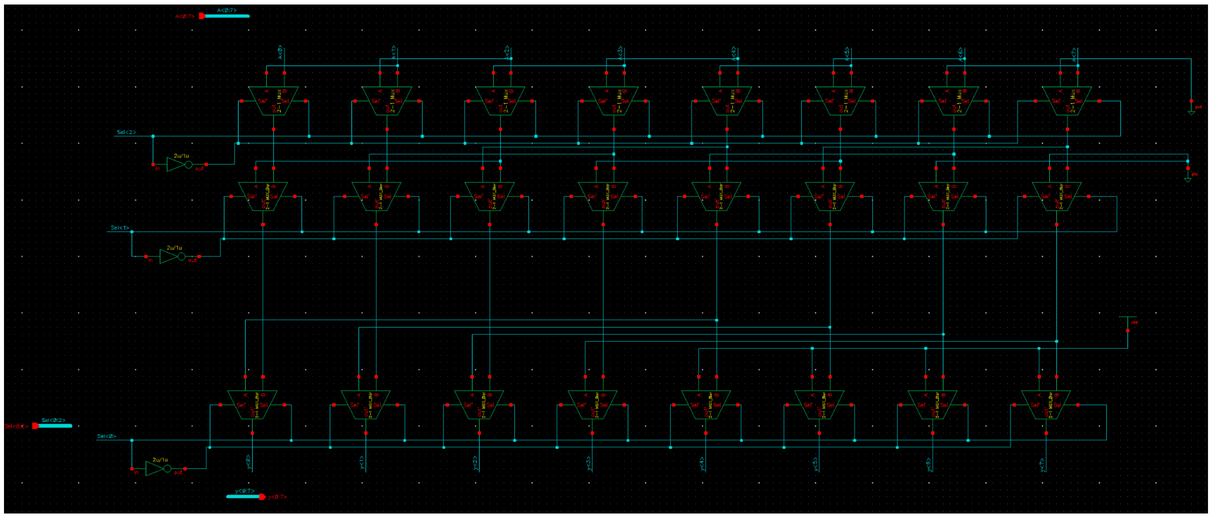


Figure 6: Logarithmic shifter circuit schematic

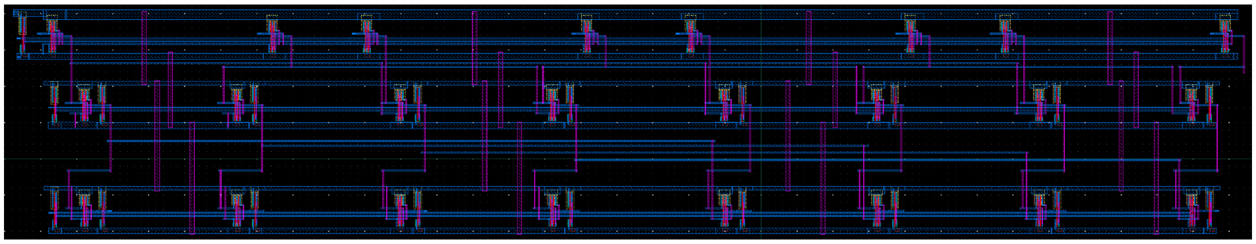


Figure 7: Layout of shifter circuit (same bit pitch as adder circuit)

Shifter Bypass Circuit

The purpose of the shifter bypass is to supply the appropriate selection bits to the shifter. When we are bypassing the shifter we want the selection bits to be all zero. When we are not bypassing the shifter we want the shifter selection bits to be the signals issued by the PLA. **Figure 8** shows the schematic for this bypassing circuit. The circuit uses a combination of a tristate and a nFET pulldown transistor. When the bypass is 1, which happens whenever we are not using the shifter, the tristates are in their high-impedance state and the outputs to the shifter are pulled to 0. When the *bypass* is low, the tristate drivers read the input signal through and into the shifters selection bits. **Figure 9** shows the schematic for a 1-bit tristate driver. This tristate driver is used extensively in our microcontroller design. The layout for the shifter bypass circuit is shown in **Figure 10**. This layout is LVS and DRC clean.

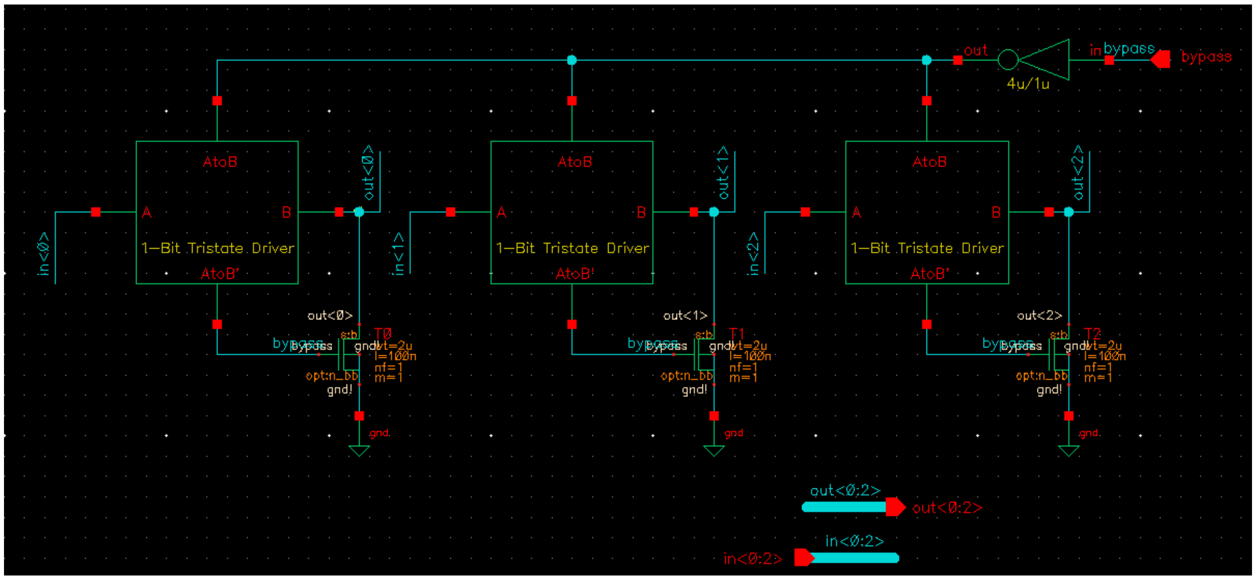


Figure 8: Schematic of shifter bypass circuit

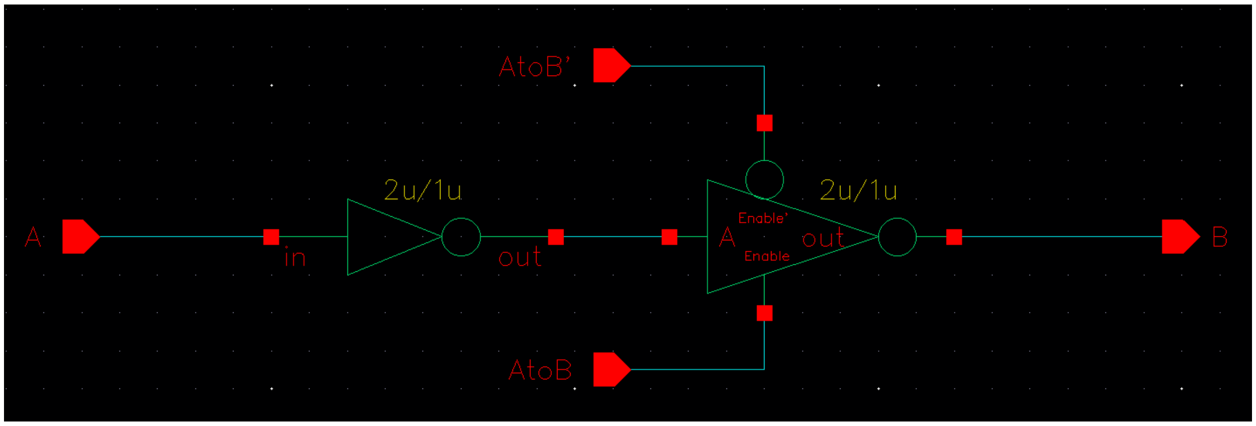


Figure 9: Schematic of 1-bit tristate driver

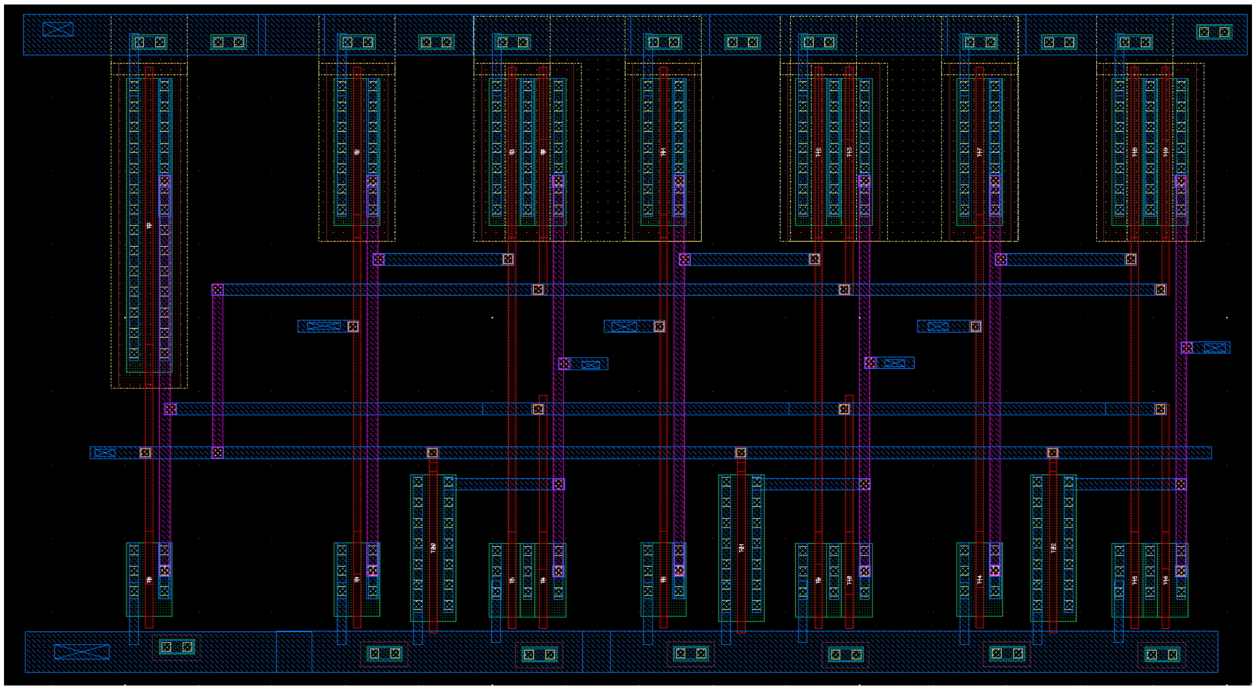


Figure 10: Bypass circuit layout

3-Input 8-bit MUX

Figure 11 shows the schematic for our 3-input 8-bit MUX. Our MUX uses 3 fully-decoded (one-hot) select lines. The circuit uses 8 1-bit 3-input MUX connected in a bus formation. The schematic for the 1-bit 3-input MUX is shown in **Figure 12.** This MUX uses complementary pass transistors to connect the appropriate input to the output. The layout for 1 bits computation, one part of the entire 3-input 8-bit MUX, is shown in **Figure 13.**

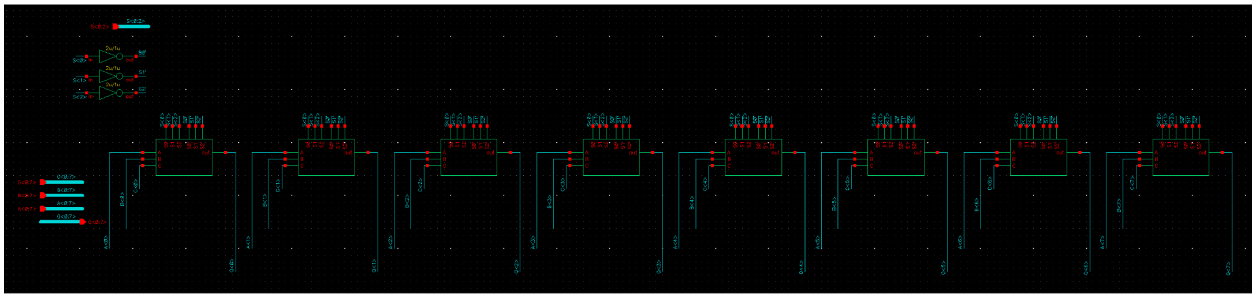


Figure 11: Schematic of 3-input 8-bit MUX

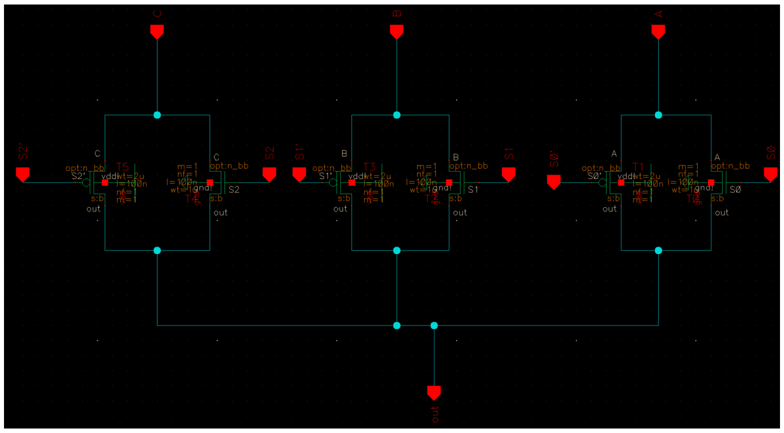


Figure 12: 1-bit 3-input MUX

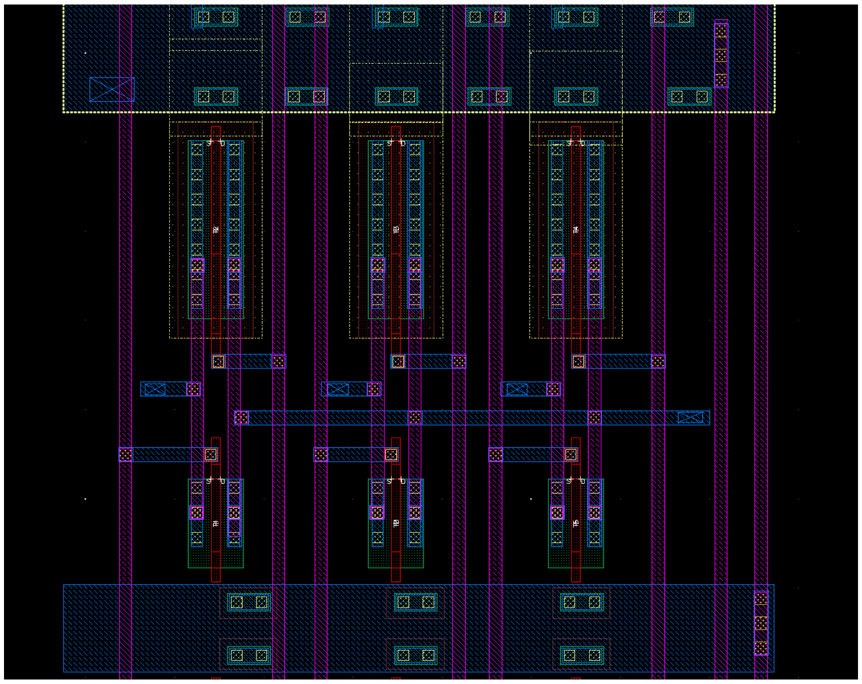


Figure 13: 1-Bit 3-input MUX

<u>8-Bit Latch</u>

8-bit latches are used extensively in our design to implement our accumulator and to latch our control signals when needed. The schematic for our latch is shown in **Figure 14**. This 8-bit latch is composed of 8 D-latches all talking the same input signal. The schematic and layout for our 1-bit D-latch are shown in **Figure 15** and **Figure 16**. The layout for our full 8-bit latch is shown in **Figure 17**. All of these layouts are DRC and LVS clean.

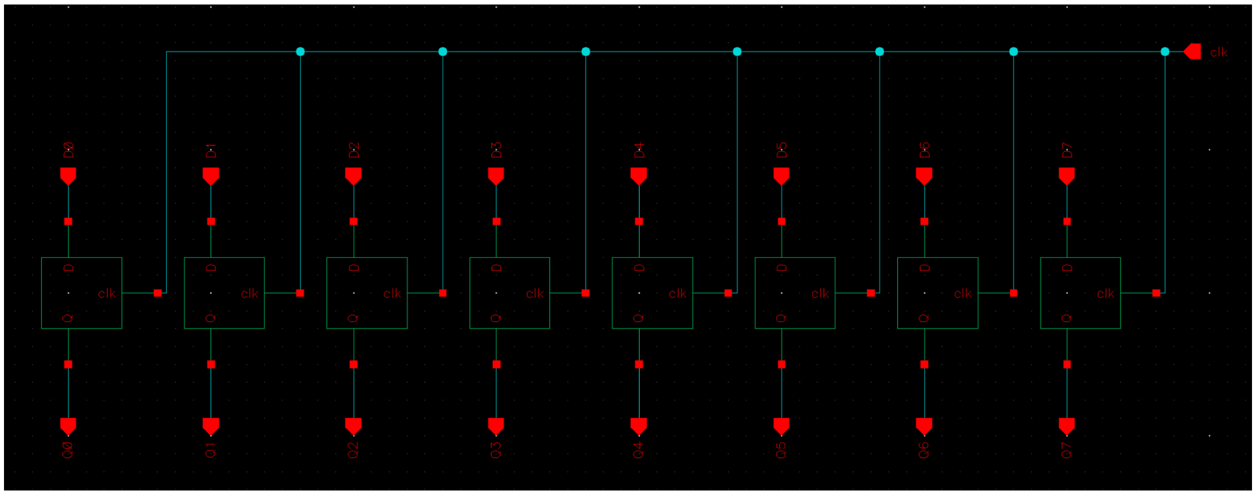
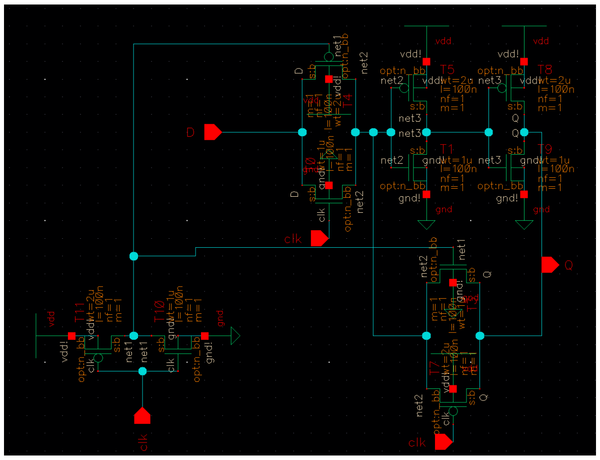


Figure 14: Schematic 8-Bit Latch



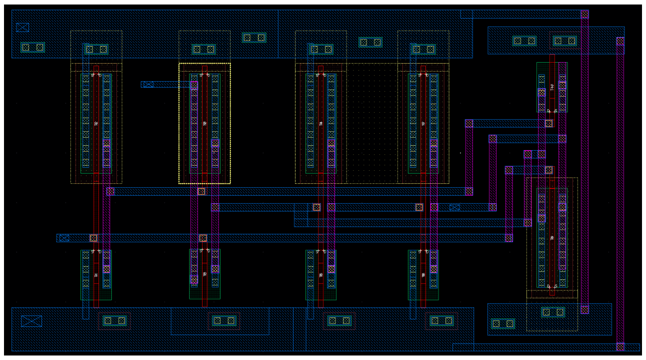


Figure 15: Schematic 1-bit D-Latch

Figure 16: 1-bit D-latch layout

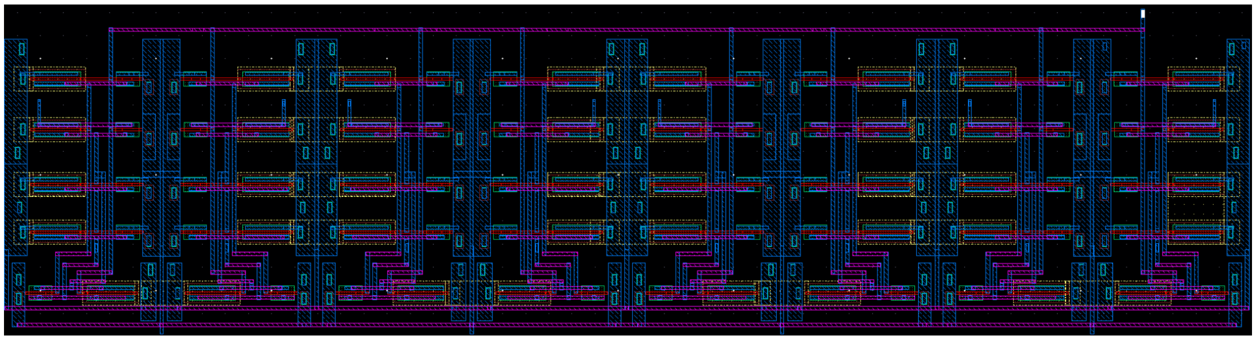


Figure 17: Layout of 8-bit Latch

8-Bit Driver

Figure 18 shows the circuit schematic for our 8-bit bus driver. Our driver is implemented using 1-bit tristate drivers, detailed in **Figure 9**. This circuit drives the 8-bit data bus whenever the control signal *AtoB* is high. This signal is an output of the control PLA. The DRC and LVS clean layout for this driver is shown in **Figure 19**.

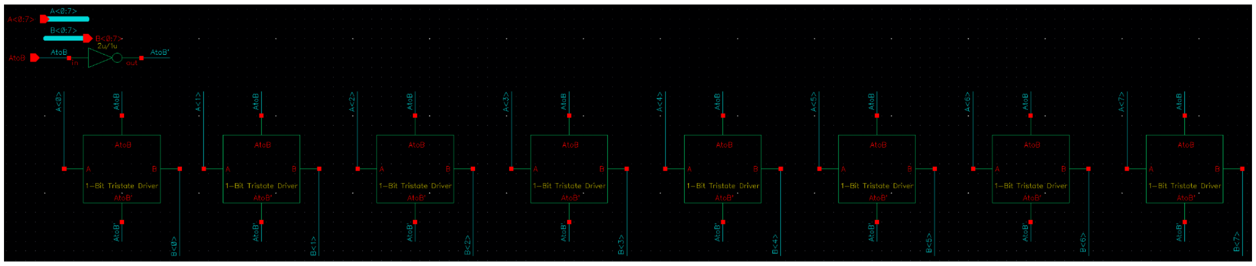


Figure 18: Schematic of 8-bit tristate bus driver

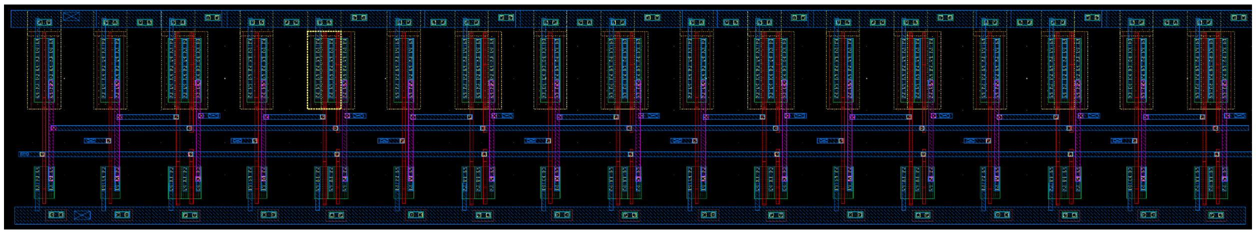


Figure 19: Layout of 8-bit tri state bus driver

Controls PLA

The controls for our different circuitry (shifter, adder, memory unit, bus driver, shift bypass, mux) are all generated by a programmable logic array. **Diagram 2** shows the input-output combination for our PLA inputs, *instr<0:2>*, and our many outputs. This table was then fed to espresso to minimize product terms. **Figure 20** shows the PLA's symbol in order to depict the different output signals. **Figure 21** shows the circuit schematic for this PLA. Each transistor has a width of 500 nm while the signal output inverters are sized 8u/4u. **Figure 22** shows the DRC and LVS clean layout for the PLA.

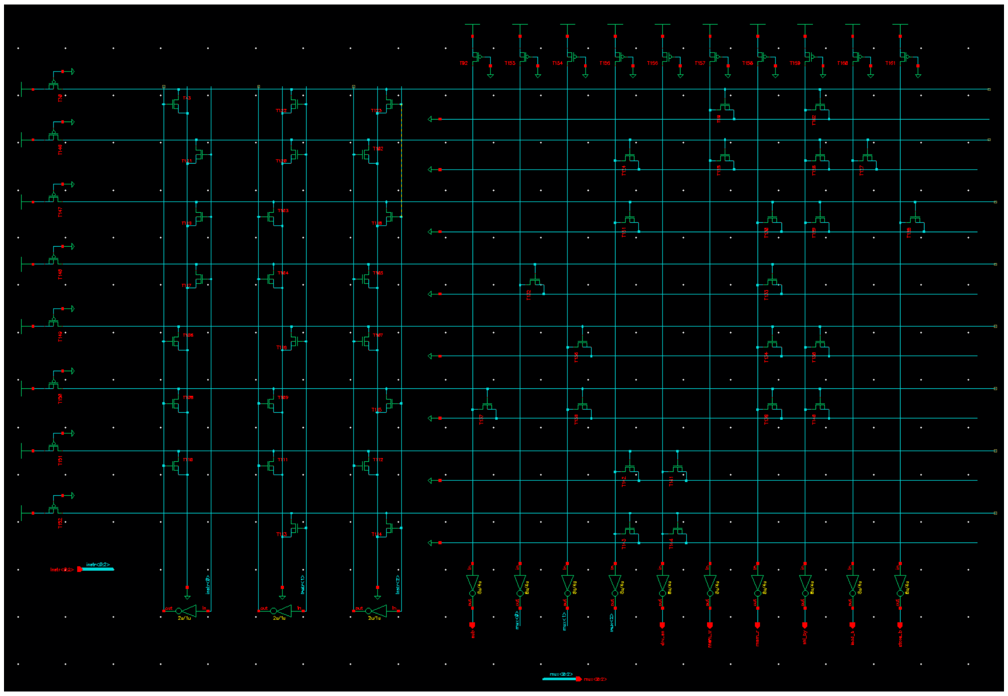


Figure 20: Controls PLA schematic

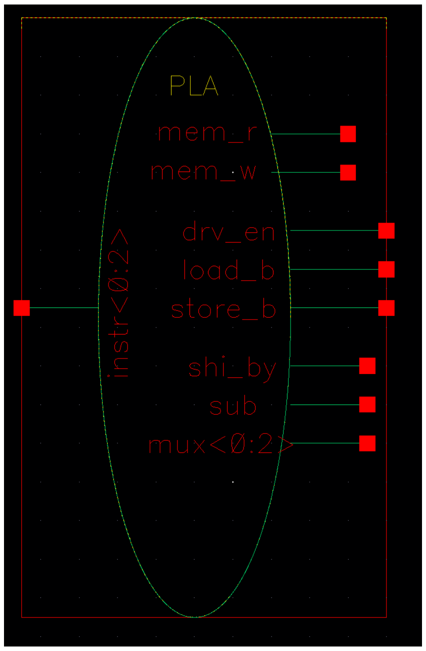


Figure 21: PLA symbol with control outputs

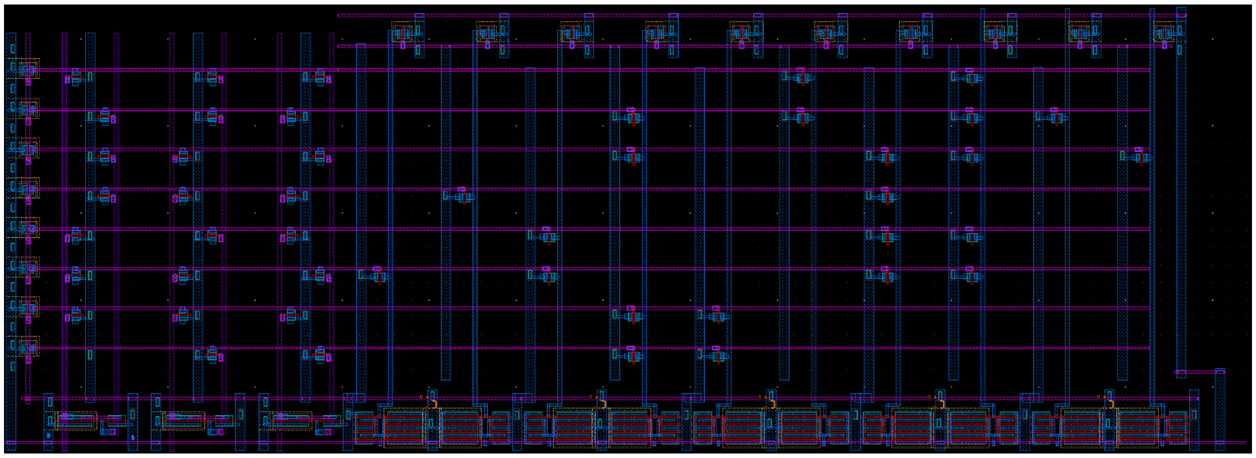


Figure 22: Layout of Controls PLA

<u>8 Bit-8 Word-Memory Cell</u>

The memory cell takes in a 3 bit address and the read and write signals along with the clock. It is the heart of the functionality of our CPU. Each Sram cell is on bit and bit bar rails that are pulled up in phase with phi2. When phi2 and the rails go low, the rails nodes are in a floating state and to write one of them is pulled down by nfets at the bottom of the rails. To read the rails during phi1 high are only driven by the sram cells, and whatever state they store will pull either bit or bit bar to ground. The memory required some fairly extensive debugging including resizing and adding circuitry to qualify memread with phi1. The layout is DRC LVS clean and measured 47u by 66u.

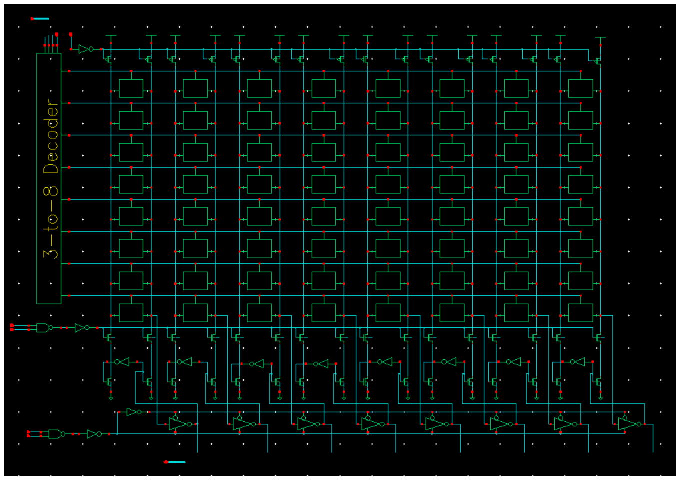


Figure X: Schematic of 8 Bit-8 Word Memory Block

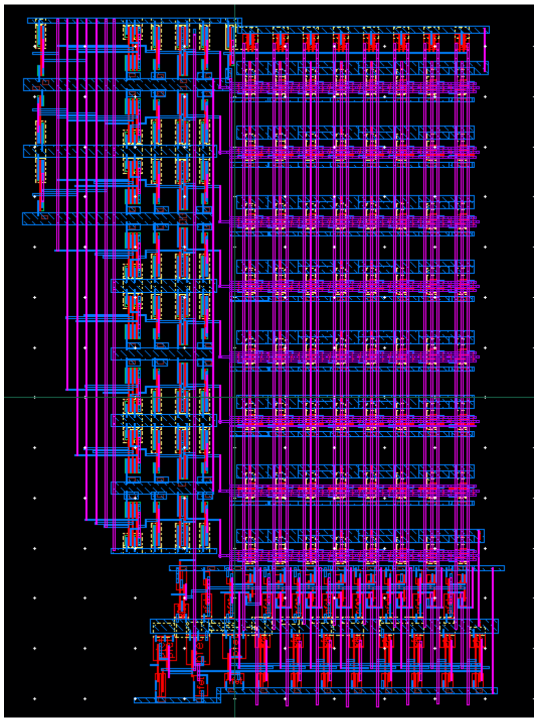


Figure X: Layout of 8 Bit-8 Word Memory Block

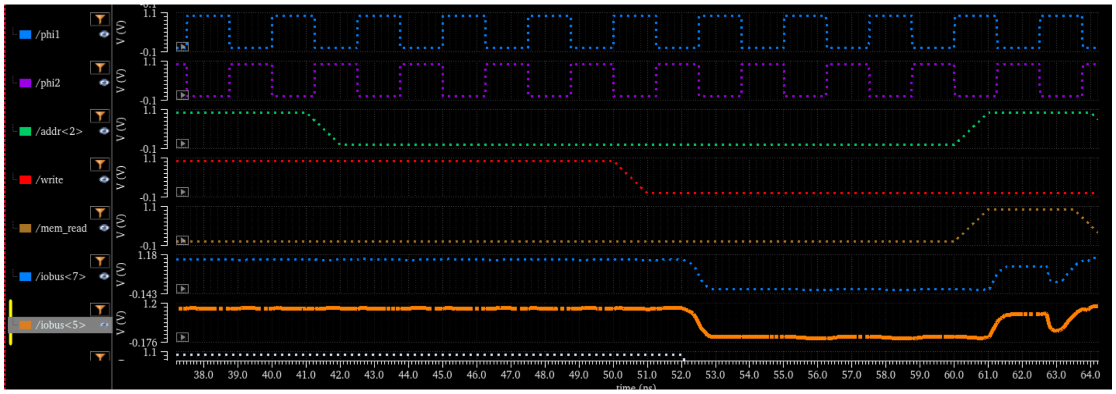


Figure X: Memory functionality Specter simulation with PEX

Figure X above shows the PEX extracted memory cell functioning well at a 400MHz period. addr<2> is functioning as the worldline, and we see that that iobus<7> and ios<5> are high while th write and worldline(addr<2>) signals are high until t=42ns when worldline(addr<2>) goes low. Next, the write signal goes low at t=50ns. This has no effect because the worldline is already low. At t=51ns we see that both iobus<7> and ios<5> are driven low by another signal on the bus. Finally, at t=60ns, both the worldline(addr<2>) and mem_read go high, and at the next rising edge of phi1 at t=61ns the values of 1 and 1 that had been saved for bit<7> and bit<5> are sent to the bus.