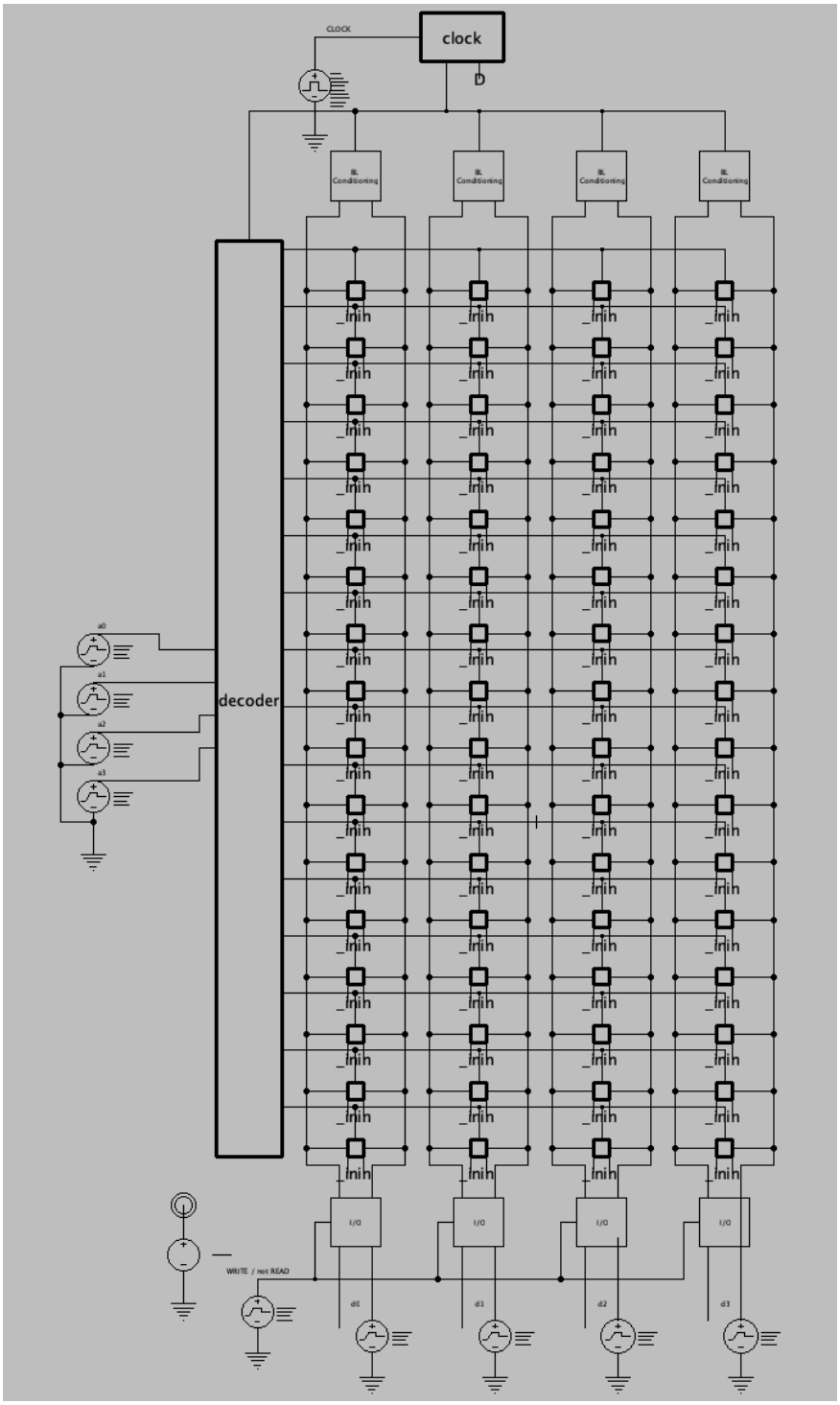
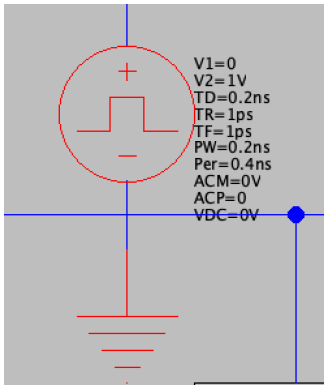
Project 2 Final Report

Schematics



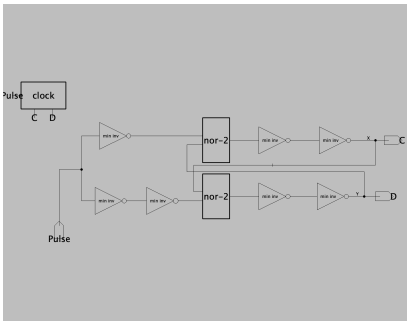
VPWLs on the left are the address inputs. The VPWLs on the bottom (except for the far left one) are the data inputs; the far left one is Write/not Read. The floating wires on the bottom are the data out pins. The Vpulse on the top is the driving clock.

Driving Clock



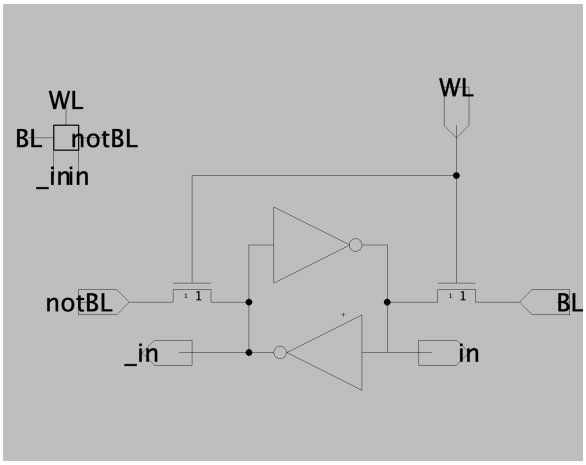
The min clock period is 0.4ns, so the max frequency is 2.5 GHz, which is well above the required max frequency of 500 MHz.

Clock Cell



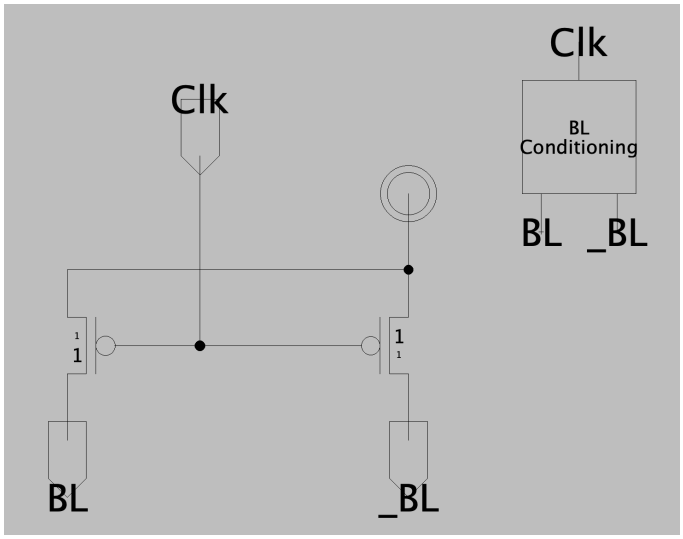
Note: inverters and nor-2 gates are min-size and implemented in CMOS.

6T Cell

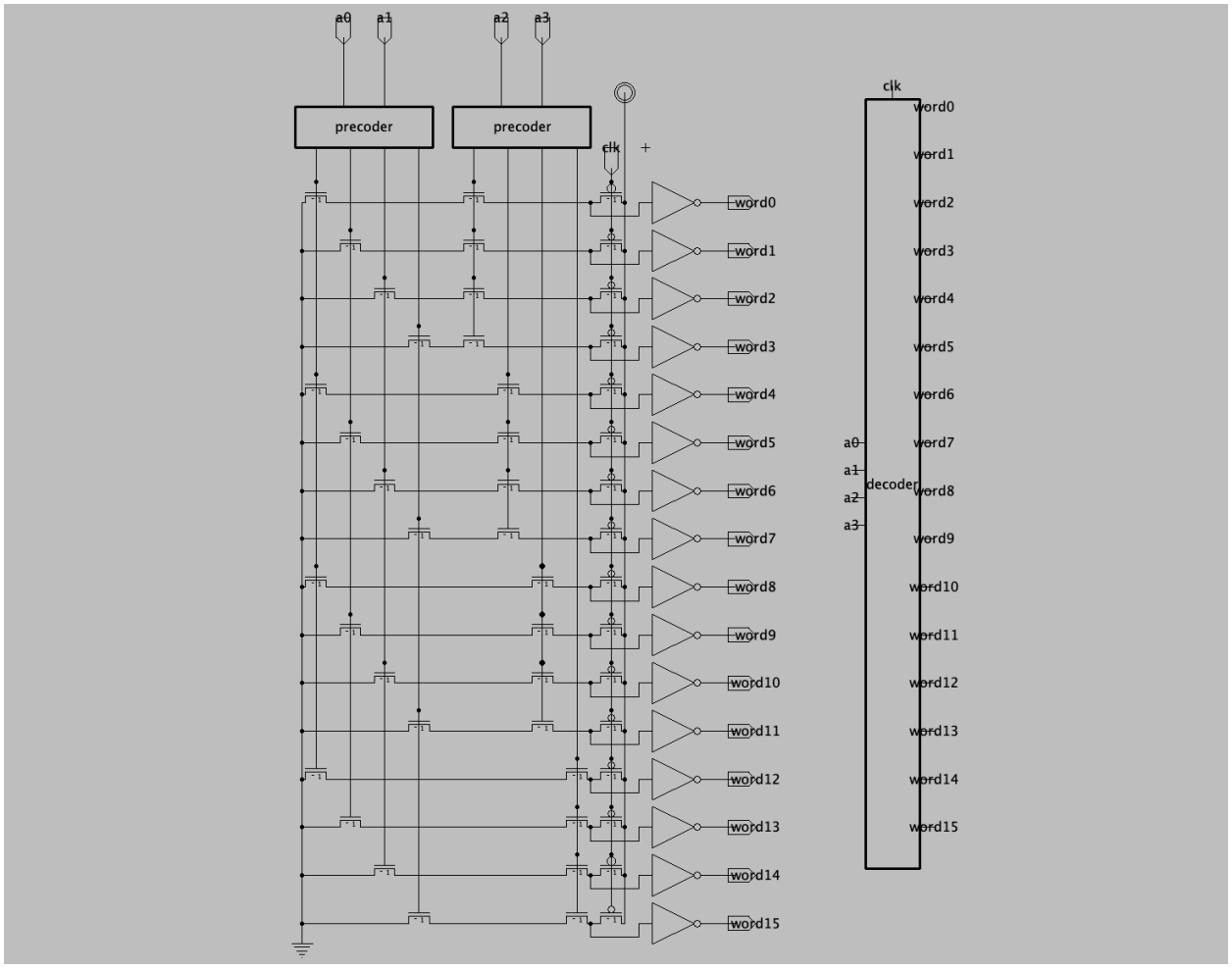


Note: inverters are min-size CMOS inverters

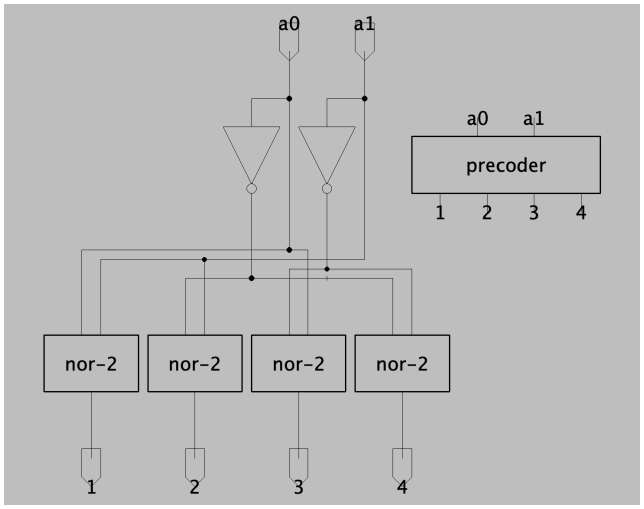
Bitline Conditioning Cell



Decoder Cell

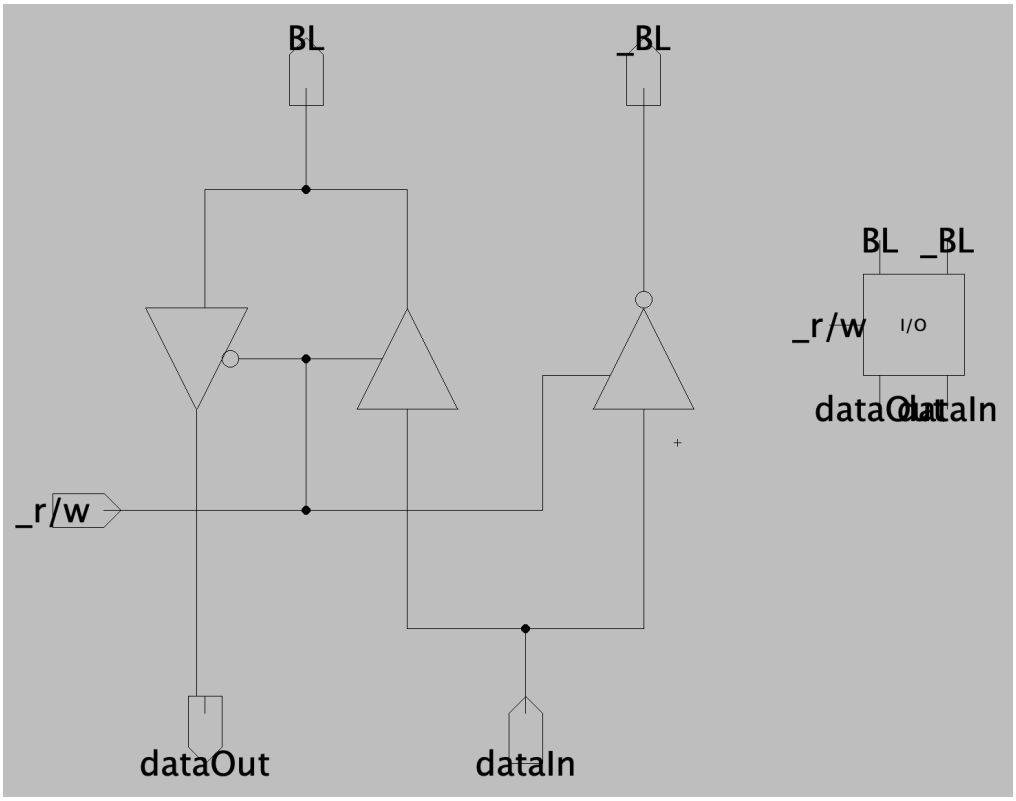


Note: precoder cell implementation is displayed below. All transistors are min-size.

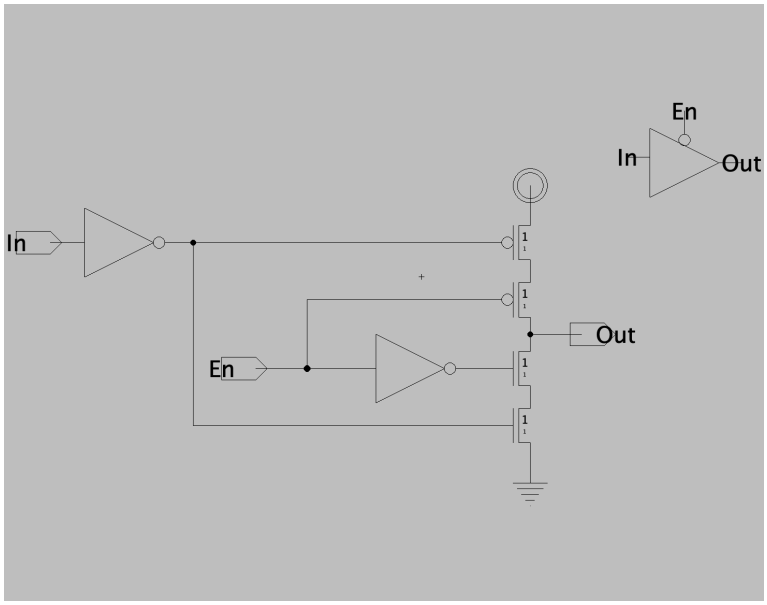


Nor-2 and inverter gates consist of min-size transistors.

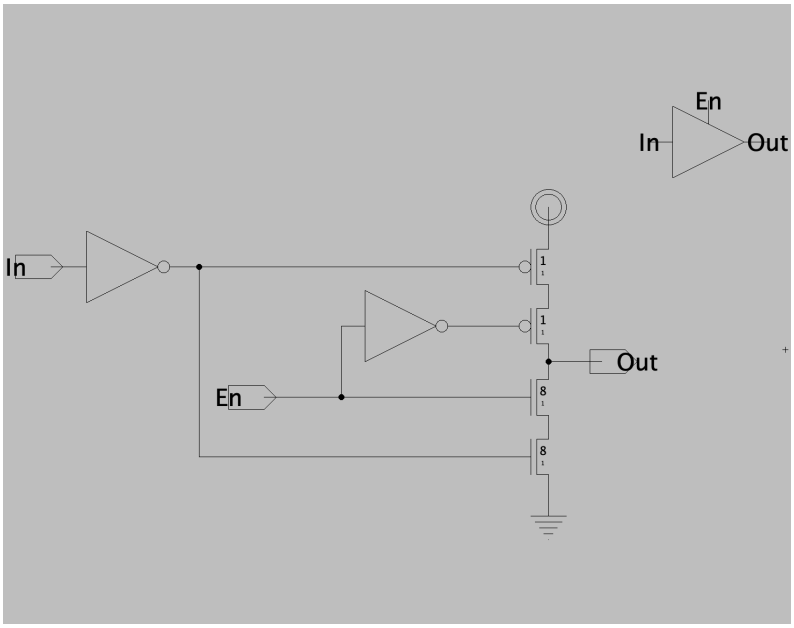
I/O Cell



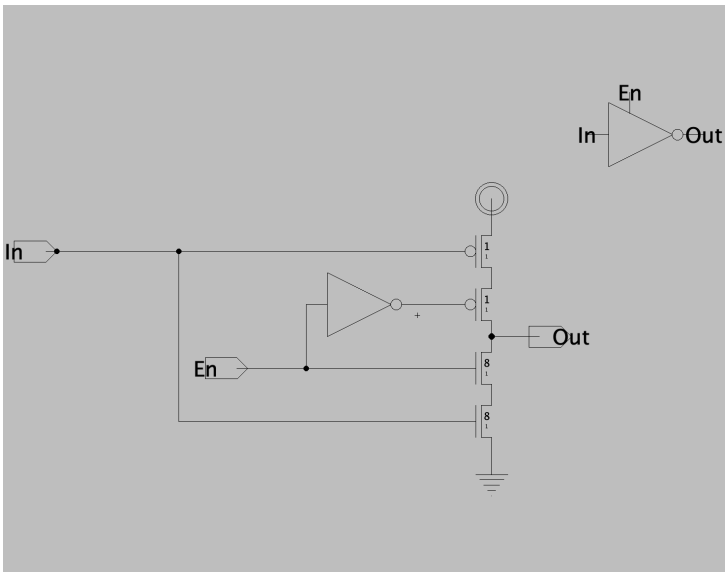
Note: for the icon, the dataOut pin is to the left of the dataIn pin. Also, the implementations for the tristate buffers/inverters are provided below



Tristate Buffer with inverted Enable pin. Inverters are min-size CMOS inverters.



Tristate Buffer. Inverters are min-size CMOS inverters. Note that the NMOS transistors are sized to width 8.



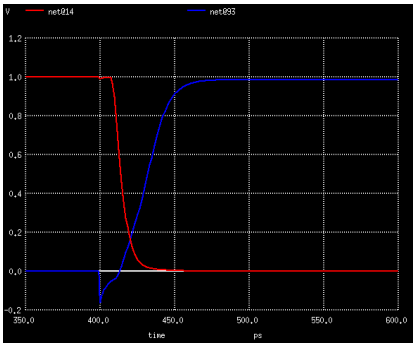
Tristate Inverter. Inverters are min-size CMOS inverters. Note that the NMOS transistors are sized to width 8.

Read and Write Operations

Reading

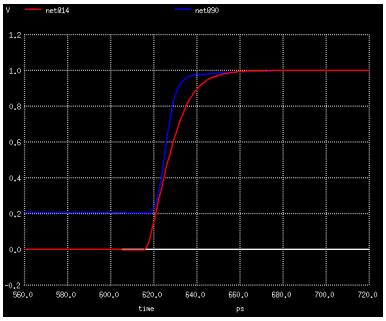
1. First, the clock will be pulled down, causing the bitline to be precharged.

   The cell being read from in this example is storing a 0.



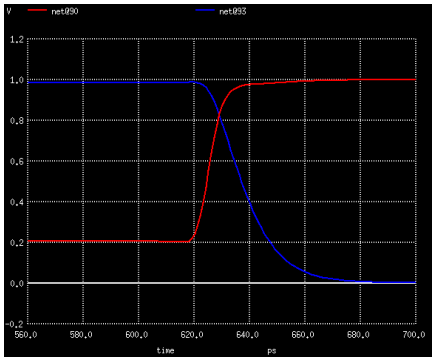
Note that net@14 is the clock being pulled down the net@93 is the bitline being precharged.
Delay: 18.01992 ps

2. Then, the clock will be pulled up, causing the bitline to float and the wordline for the desired cells to be pulled up.



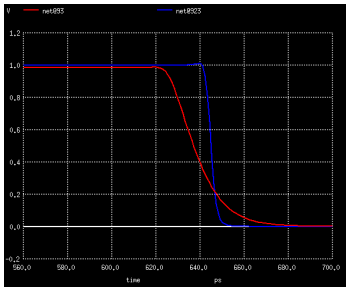
Note that net@14 is the clock being pulled up and net@90 is the wordline being pulled up. Net@14 rises before net@90, but net@90 rises faster, so I measured the time between when both started rising.
Delay: 0.7735762 ps

3. Then, the cells will output their internal values to their bitlines



Note that net@90 is the wordline being pulled up and net@93 is the bitline, where the cell is inputting its value to.
Delay: 41.23565 ps

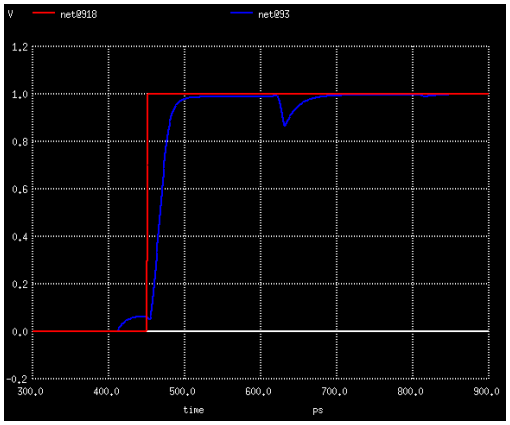
4. Finally, the I/O cells will read the values at the bitlines and output these values to their data pins.



Note that net@93 is the bitline and net@923 is the data out pin.
Delay: 7.870416 ps

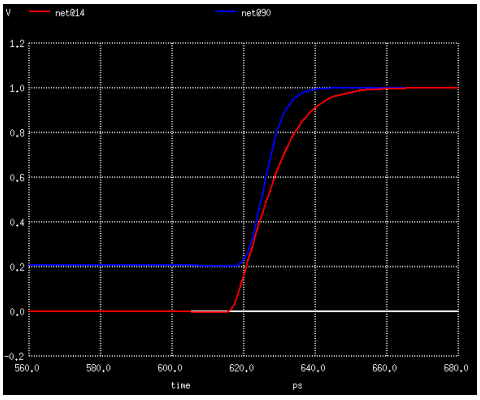
Writing

1. First, the write/not read input will be pulled up, causing the bitline to be driven to whatever is in the data input of the I/O cells. This happens while the clock is pulled down.
   The cell being written to in this example is initially storing a 0.



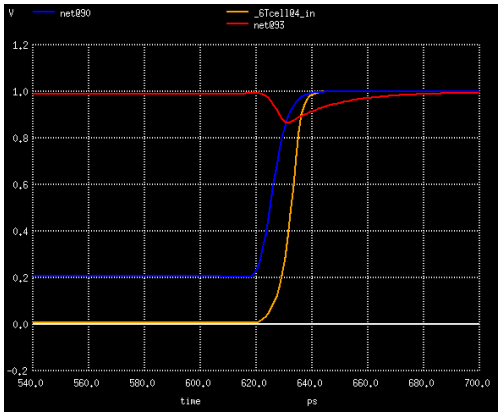
Note that net@918 is the Data Input Pin and net@93 is the bitline being driven.
Delay: 16.92133 ps

2. Then, the clock will be pulled up, causing the worldline for the desired cells to be pulled up.



Note that net@14 is the clock being pulled up and net@90 is the wordline being pulled up. Net@14 rises before net@90, but net@90 rises faster, so I measured the time between when both started rising.
Delay: 0.7735762 ps

3. Then, the values the bitlines are driven to will be written to the internals of the desired cells.



Note that net@90 is the wordline, net@93 is the bitline, and the bright orange line is the cell internal. The cell internal is being written to the value on the bitline when the wordline is pulled up

Delay: 24.26804 ps

Explanation of Design

The SRAM's clock cycles have two phases: the precharge phase and then the read/write phase. The clock differentiates these two phases by being pulled down for the first half of each cycle and then being pulled up for the second half of each cycle.

Sub-Components Overview

The Clock cell generates a realistic clock signal from the inputted Vpulse. I used this instead of directly using the Vpulse as a clock because it was more realistic. Additionally, I thought that the non-overlapping complement of the clock signal would prove useful, but I never ended up using it. As a result, one of the pins of the clock cell is unused.

The Bitline Conditioning cell precharges the bitlines when the clock is low and then leaves the bitlines floating when the clock is high. It uses PMOS transistors that have the clock as inputs to drive the clock high. I kept the PMOS transistors min-size to keep its drive strength low.

The I/O cell will output whatever is on the bitline to the data out pin if the Write/Not Read pin is low. It accomplishes this with a tristate buffer with an inverted enable input, so when the W/nR is low, the buffer turns on. If the Write/Not Read pin is high, then the I/O cell will drive the bitline and bitline bar to whatever is on the data in the pin (and its complement for the bitline bar). It accomplishes this using a tristate inverter and tristate buffer. These tristate components have strong pull down networks, so they will overpower the Bitline Condition cell's precharging phase if the bitline or bitline bar need to pulled down.

The decoder will output all 0s when the clock is low and then pull up the desired wordline when the clock is high, thus allowing values on the bitline to be read/written after precharging/driving the bitline. It uses dynamic logic, where the precoder converts the 4-bit signal into 8 bits, which can then be fed into a NMOS NOR array. The NOR gate that corresponds to the desired wordline will pull down the input of its inverter, causing the desired wordline to be pulled up. I kept everything min-sized here to minimize the area. Additionally, it worked fine during testing.

The individual cell is a 6T topology consisting of min-size transistors. The coupled inverters allow a 1 or a 0 to be "stored" within the cell and the pass transistors enable the value stored within the cell to be outputted or changed. I used this design because it had a low risk for read/write failures. Additionally, I kept everything min-sized here to minimize the area. While I could have sized the transistors, the cell already worked perfectly fine with min-size transistors, so I just kept it that way.

Reading

For read operations, the clock, which is first pulled down, will cause the Bitline Conditioning cell to pull up the bitline. This is because within the Bitline Conditioning cell the clock signal is inputted into the gate of a PMOS transistor, so when the clock is 0, the PMOS is turned on, pulling up the bitline and bitline bar.

For the second phase of the read operation, the clock is pulled up. This causes the PMOS transistor within the Bitline condition cell to be turned off, thus leaving bitline and bitline bar floating for the cell to output its internals to. At the same time, the clock signal being pulled up will cause the decoder to pull up the desired wordline. This is because when the clock signal is low, the decoder will always output all 0s, as all the PMOSs within the decoder are on, which will input all 1s to the inverters, outputting 0s. However, when the clock is high, the PMOSs will be turned off, enabling the decoder to pull up the desired wordline. Now, with all the bitlines floating and the wordline for the desired word being pulled high, the desired cells will output their internals to the bitline. Finally, because the Write/Not Read input will be low during read operations, the I/O cells will output whatever is on their bitline to their data out pins. Thus, the desired word will be outputted to the data out pins, completing the read operation.

Writing and Explanation of Sizing the Tristate Buffers and Inverters

For write operations, the Write/Not Read pin will be high, so the I/O cell will have its tristate buffers and inverters outputting whatever is on the Data In pins to the bitline. Note that these tristate buffers and inverters, as seen in their schematics, use NMOS transistors that are sized to width 8. This is because in order to write a 0 to a bitline that is being precharged to 1 by the clock, the tristate buffer/inverter's pull down network must be stronger than the Bitline Conditioning cell's single min-size PMOS transistor. In order to accomplish this, the NMOSs are sized the width 8, giving them a higher drive strength. Of course, the PMOSs in the tristate buffers/inverters don't need to be sized because they don't need to overpower the Bitline Condition cell, as both will be driving the bitline to 1.

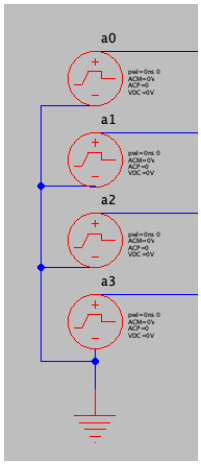
In this way, the I/O will drive the bitline to the value on the data-in pin as long as the Write/Not Read pin is high. Then, in the middle of the write cycle, the clock will be pulled up, causing the decoder pull the desired wordline high. Now, the bitlines, which are driven to their desired values, will write to the desired word in the SRAM, thus completing the write operations.

SRAM Validation

First, I tested read/write operations on a single cell (first cell in the word with address 0 0 0 0)

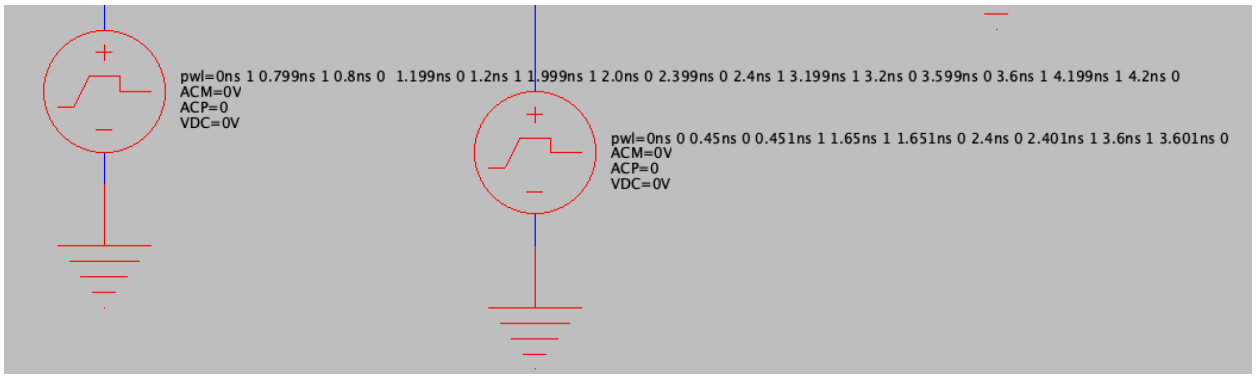
Setup

Word Address:



The cell under test was the first cell in the word with 0 0 0 0, so a0, a1, a2, and a3 were all 0V.

I combined a series of write and read cases into a single transient:



Note that the min clock frequency is 2.5 GHz, so the period is 0.4ns.

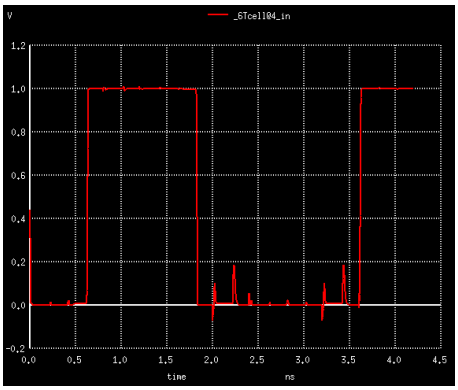
The left VPWL is the write/not read pin. The right PWL is the data input for the first cell in the word. These VPWLs will do the following:

1. Write 0 and then write 1
2. Read the 1
3. Write 1 and then write 0
4. Read the 0
5. Write 0 and then write 0
6. Read the 0
7. Write 1 and then write 1

This way, all both cases of reads are tested (1 and 0) and all possible 2-combinations of writes are tested.

Simulation and Explanation of Cell Behavior

Plot of Cell Internal Voltage



Explanation:

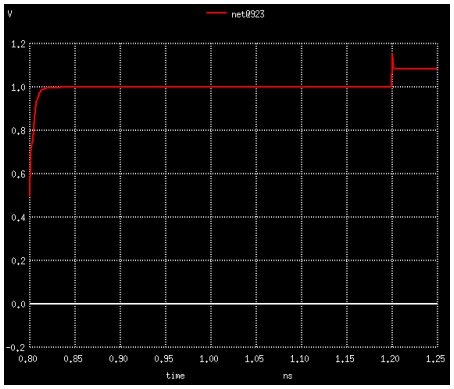
In the first write cycle (0 to 0.4ns), a 0 is written, which is why the plot starts a 0. In the second write cycle (0.4ns to 0.8ns), a 1 is written, so the cell is pulled up to 1V at ~0.6ns. Then, the 1 is read from the cell in the next cycle (0.8ns to 1.2ns). Note that this does not overwrite the cell internals.

Then, in the next two write cycles (1.2ns to 1.6ns and 1.6ns to 2.0ns), a 1 is written and then a 0. Therefore, the internals of the cell switches from 1 to 0 at ~1.8ns. Then, the 0V is read from the cycle in the next cycle (2.0 to 2.4 ns). Note that this does not overwrite the cell internals.

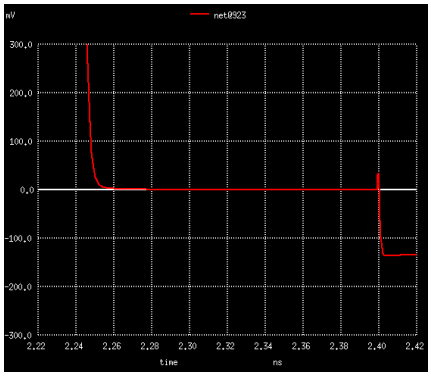
The next two write cycles (2.4ns to 2.8ns and 2.8ns to 3.2ns) sees two 0s being written, so the cell is held at 0V. This is then read between 3.2ns and 3.6ns, which does not overwrite the internals. Finally, the last two write cycles (3.6ns to 4.0ns and 4.0ns to 4.2ns) sees two 1s being written, so the cell is pulled up to 1V.

Plots for Data Output Pin for the Read Operations

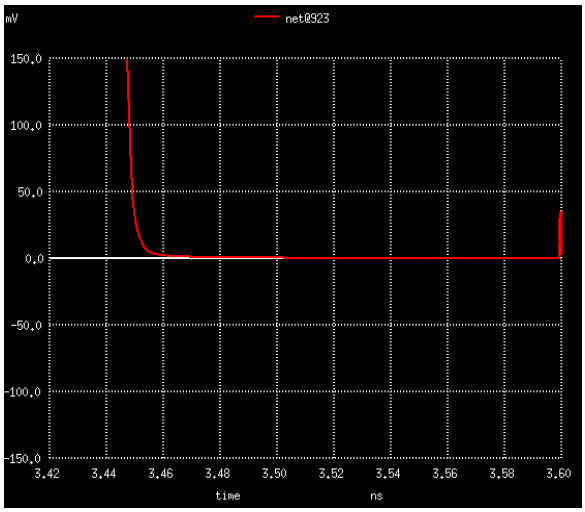
The data out pin (net@923) is where all the reads operations are outputted to.



Clearly, a 1 is outputted to the data out pin between 0.8ns and 1.2ns, correctly indicating that a 1 was read from the cell.



Clearly, a 0 is outputted to the data out pin within the 2.0ns to 2.4ns read cycle. Note that during the first half of the cycle (2.0ns to 2.2ns) the bitline is being precharged. It's only during the second half of the cycle (2.2ns to 2.4ns) that the cell internals are outputted, which is clearly seen.

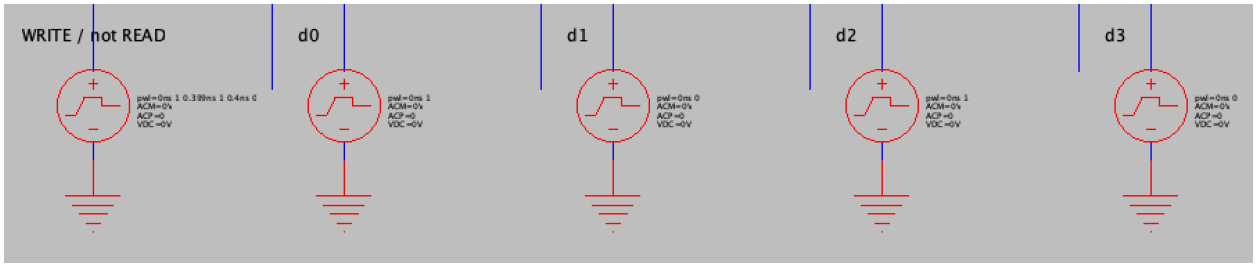


Clearly, a 0 is outputted to the data out pin within the 3.2ns to 3.6ns read cycle. Again, the first half of the cycle (3.2ns to 3.2ns) is dedicated to precharging the bitline, so the 0 is outputted during the second half of the cycle (3.4ns to 3.6ns).

Then, just to ensure the encoder and the other bitlines were working correctly, I tested writing a 1010 to the 15th word (second to last).

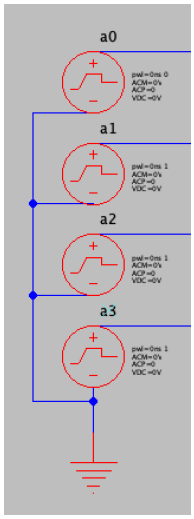
Setup

VPWL Parameters



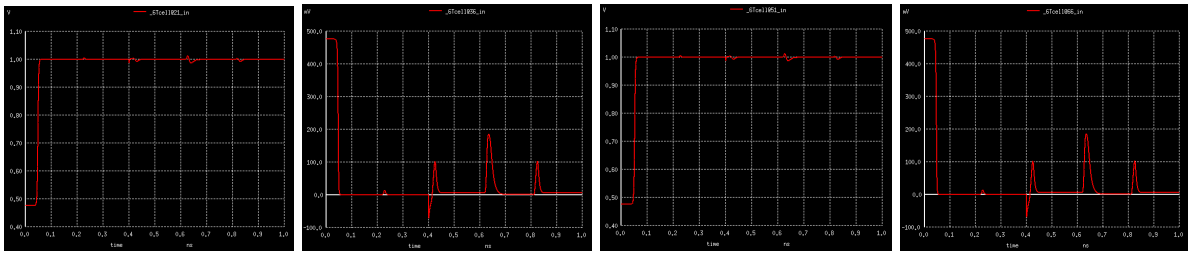
Note that the min clock frequency is 2.5 GHz, so the period is 0.4ns.

Word Address



Note: the 15th word is actually 14 in binary, which is 0111.

Simulation and Explanation of Cell Behavior



The cells are displayed from LSB to MSB. Clearly, their internals contain the bit 1010 (also LSB to MSB).

FOM Metric Measurements - Delay

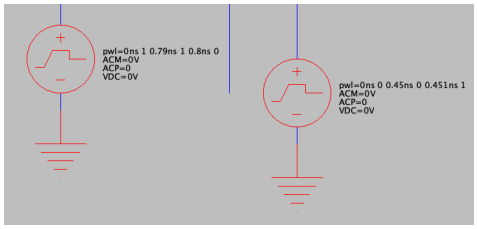
For measuring the delay of write operations, I measured the 50 to 50 delay between the clock being pulled down (signaling the start of a write operation) and the cell internal rising/falling.

For measuring the delay of read operations, I measured the 50 to 50 delay between the clock being pulled down (signaling the start of a read operation) and read being outputted to the data out pin. Note that it was not possible to measure the delay for reading a 1 because there is no falling edge on the data out pin when reading a 1 (it is initially pulled high because of the precharge and then held high for the rest of the cycle). So, for this case, I just measured the delay between the clock and the bitline bar falling.

Write 1

Delay: 216.6860 ps

VPWL Parameters



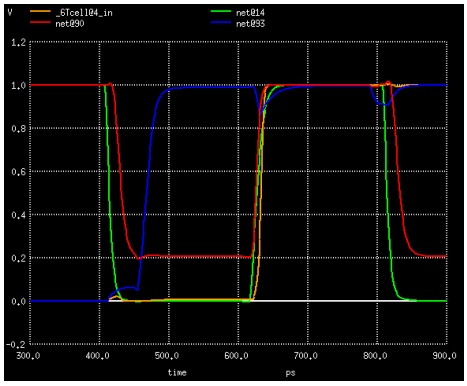
Note that the min clock frequency is 2.5 GHz, so the period is 0.4ns.

Commands

plot net@90 net@93 _6Tcell@4_in net@14

meas tran delay trig net@14 val=0.5 fall=1 targ _6Tcell@4_in val=0.5 rise=1

Plot

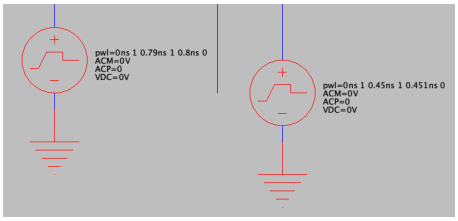


Note that the bright green (net@14) line is the clock and the bright orange line (_6Tcell@4_in) is the cell internal. The delay is measured from when the clock is pulled down to 0.5V at ~ 410ps (write operation starting) to when the cell internal is pulled up to 0.5V at ~620ps (writing a 1).

Write 0

Delay: 216.6860 ps

VPWL Parameters



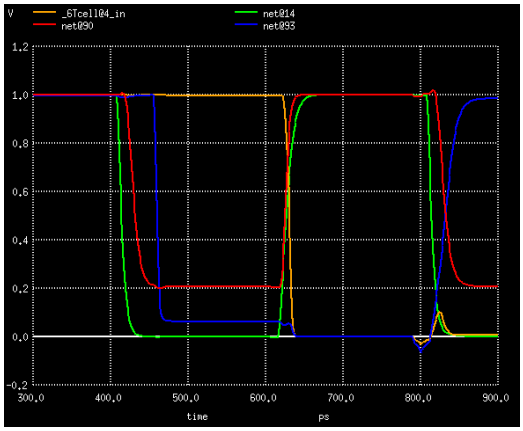
Note that the min clock frequency is 2.5 GHz, so the period is 0.4ns.

Commands

plot net@90 net@93 _6Tcell@4_in net@14

meas tran delay trig net@14 val=0.5 fall=1 targ _6Tcell@4_in val=0.5 fall=1

Plot

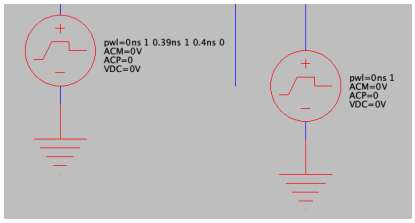


Note that the bright green (net@14) line is the clock and the bright orange line (_6Tcell@4_in) is the cell internal. The delay is measured from when the clock is pulled down to 0.5V at ~ 410ps (write operation starting) to when the cell internal is pulled down to 0.5V at ~620ps (writing a 0).

Read 1

Delay: 225.3387 ps

VPWL Parameters



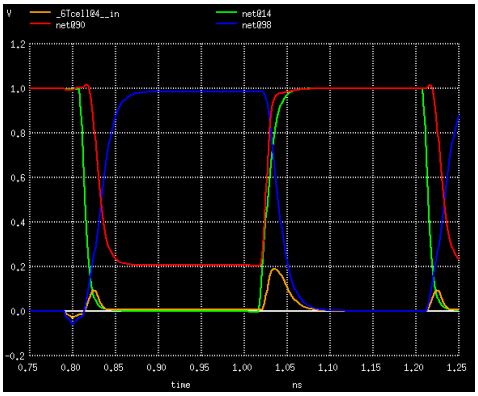
Note that the min clock frequency is 2.5 GHz, so the period is 0.4ns.

Commands

plot net@90 net@98 _6Tcell@4__in net@14

meas tran delay trig net@14 val=0.5 fall=1 targ net@98 val=0.5 fall=1

Plot

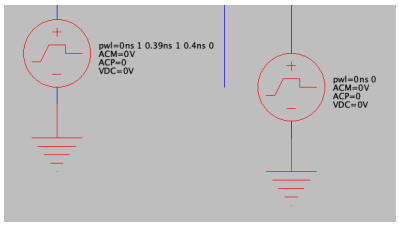


Note that the bright green (net@14) line is the clock and the blue line (net@98) is bitline bar. The delay is measured from when the clock is pulled down to 0.5V at ~ 0.82ns (read operation starting) to when bitline bar is pulled down to 0.5V at ~1.04ns (reading a 1).

Read 0

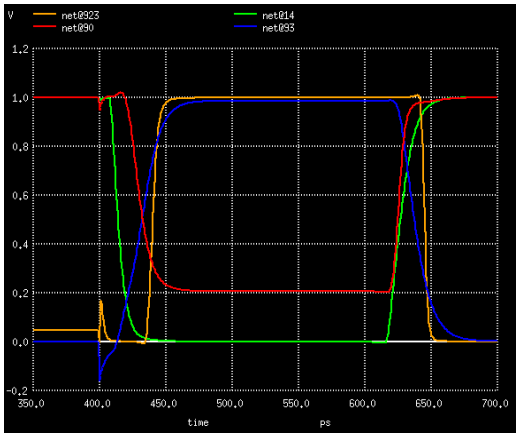
Delay: 231.2697 ps

VPWL Parameters



Commands
plot net@90 net@93 _6Tcell@4_in net@14
meas tran delay trig net@14 val=0.5 fall=1 targ net@98 val=0.5 fall=1

Plot



Note that the bright green (net@14) line is the clock and the bright orange line (net@923) is the data out pin. The delay is measured from when the clock is pulled down to 0.5V at ~ 0.41ns (read operation starting) to when the data out pin is pulled down to 0.5V at ~0.49ns (reading a 0).

FOM Metric Measurements - Power

Power Consumption: 4.01385 pJ

VPWL Decoder Parameters (couldn't screenshot VPWL as it was too long):

a0

0ns 0 0.399ns 0 0.4ns 1 0.799ns 1 0.8ns 0 1.199ns 0 1.2ns 1 1.599ns 1 1.6ns 0 1.999ns 0 2.0ns 1 2.399ns 1 2.4ns 0 2.799ns 0 2.8ns 1 3.199ns 1 3.2ns 0 3.599ns 0 3.6ns 1 3.999ns 1 4.0ns 0 4.399ns 0 4.4ns 1 4.799ns 1 4.8ns 0 5.199ns 0 5.2ns 1 5.599ns 1 5.6ns 0 5.999ns 0 6.0ns 1 6.399ns 1 6.4ns 0 6.799ns 0 6.8ns 1 7.199ns 1 7.2ns 0 7.599ns 0 7.6ns 1 7.999ns 1 8.0ns 0 8.399ns 0 8.4ns 1 8.799ns 1 8.8ns 0 9.199ns 0 9.2ns 1 9.599ns 1 9.6ns 0 9.999ns 0 10.0ns 1 10.399ns 1 10.4ns 0 10.799ns 0 10.8ns 1 11.199ns 1 11.2ns 0 11.599ns 0 11.6ns 1 11.999ns 1 12.0ns 0 12.399ns 0 12.4ns 1 12.799ns 1 12.8ns 0 13.199ns 0 13.2ns 1 13.599ns 1 13.6ns 0 13.999ns 0 14.0ns 1 14.399ns 1 14.4ns 0 14.799ns 0 14.8ns 1 15.199ns 1 15.2ns 0 15.599ns 0 15.6ns 1 15.999ns 1 16.0ns 0 16.399ns 0 16.4ns 1 16.799ns 1 16.8ns 0 17.199ns 0 17.2ns 1 17.599ns 1 17.6ns 0 17.999ns 0 18.0ns 1 18.399ns 1 18.4ns 0 18.799ns 0 18.8ns 1 19.199ns 1 19.2ns 0 19.599ns 0 19.6ns 1 19.999ns 1 20.0ns 0 20.399ns 0 20.4ns 1 20.799ns 1 20.8ns 0 21.199ns 0 21.2ns 1 21.599ns 1 21.6ns 0 21.999ns 0 22.0ns 1 22.399ns 1 22.4ns 0 22.799ns 0 22.8ns 1 23.199ns 1 23.2ns 0 23.599ns 0 23.6ns 1 23.999ns 1 24.0ns 0 24.399ns 0 24.4ns 1 24.799ns 1 24.8ns 0 25.199ns 0 25.2ns 1 25.599ns 1 25.6ns 0 25.999ns 0 26.0ns 1 26.399ns 1 26.4ns 0 26.799ns 0 26.8ns 1 27.199ns 1 27.2ns 0 27.599ns 0 27.6ns 1 27.999ns 1 28.0ns 0 28.399ns 0 28.4ns 1 28.799ns 1 28.8ns 0 29.199ns 0 29.2ns 1 29.599ns 1 29.6ns 0 29.999ns 0 30.0ns 1 30.399ns 1 30.4ns 0 30.799ns 0 30.8ns 1 31.199ns 1 31.2ns 0 31.599ns 0 31.6ns 1 31.999ns 1 32.0ns 0 32.399ns 0 32.4ns 1 32.799ns 1 32.8ns 0 33.199ns 0 33.2ns 1 33.599ns 1 33.6ns 0 33.999ns 0 34.0ns 1 34.399ns 1 34.4ns 0 34.799ns 0 34.8ns 1 35.199ns 1 35.2ns 0 35.599ns 0 35.6ns 1 35.999ns 1 36.0ns 0 36.399ns 0 36.4ns 1 36.799ns 1 36.8ns 0 37.199ns 0 37.2ns 1 37.599ns 1 37.6ns 0 37.999ns 0 38.0ns 1 38.399ns 1 38.4ns 0 38.799ns 0 38.8ns 1 39.199ns 1 39.2ns 0 39.599ns 0 39.6ns 1 39.999ns 1 40.0ns 0 40.399ns 0 40.4ns 1 40.799ns 1 40.8ns 0 41.199ns 0 41.2ns 1 41.599ns 1 41.6ns 0 41.999ns 0 42.0ns 1 42.399ns 1 42.4ns 0 42.799ns 0 42.8ns 1 43.199ns 1 43.2ns 0 43.599ns 0 43.6ns 1 43.999ns 1 44.0ns 0 44.399ns 0 44.4ns 1 44.799ns 1 44.8ns 0 45.199ns 0 45.2ns 1 45.599ns 1 45.6ns 0 45.999ns 0 46.0ns 1 46.399ns 1 46.4ns 0 46.799ns 0 46.8ns 1 47.199ns 1 47.2ns 0 47.599ns 0 47.6ns 1 47.999ns 1 48.0ns 0 48.399ns 0 48.4ns 1 48.799ns 1 48.8ns 0 49.199ns 0 49.2ns 1 49.599ns 1 49.6ns 0 49.999ns 0 50.0ns 1 50.399ns 1 50.4ns 0 50.799ns 0 50.8ns 1

a1

0ns 0 0.799ns 0 0.8ns 1 1.599ns 1 1.6ns 0 2.399ns 0 2.4ns 1 3.199ns 1 3.2ns 0 3.999ns 0 4.0ns 1 4.799ns 1 4.8ns 0 5.599ns 0 5.6ns 1 6.399ns 1 6.4ns 0 7.199ns 0 7.2ns 1 7.999ns 1 8.0ns 0 8.799ns 0 8.8ns 1 9.599ns 1 9.6ns 0 10.399ns 0 10.4ns 1 11.199ns 1 11.2ns 0 11.999ns 0 12.0ns 1 12.799ns 1 12.8ns 0 13.599ns 0 13.6ns 1 14.399ns 1 14.4ns 0 15.199ns 0 15.2ns 1 15.999ns 1 16.0ns 0 16.799ns 0 16.8ns 1 17.599ns 1 17.6ns 0 18.399ns 0 18.4ns 1 19.199ns 1 19.2ns 0 19.999ns 0 20.0ns 1 20.799ns 1 20.8ns 0 21.599ns 0 21.6ns 1 22.399ns 1 22.4ns 0 23.199ns 0 23.2ns 1 23.999ns 1 24.0ns 0 24.799ns 0 24.8ns 1 25.599ns 1 25.6ns 0 26.399ns 0 26.4ns 1 27.199ns 1 27.2ns 0 27.999ns 0 28.0ns 1 28.799ns 1 28.8ns 0 29.599ns 0 29.6ns 1 30.399ns 1 30.4ns 0 31.199ns 0 31.2ns 1 31.999ns 1 32.0ns 0 32.799ns 0 32.8ns 1 33.599ns 1 33.6ns 0 34.399ns 0 34.4ns 1 35.199ns 1 35.2ns 0 35.999ns 0 36.0ns 1 36.799ns 1 36.8ns 0 37.599ns 0 37.6ns 1 38.399ns 1 38.4ns 0 39.199ns 0 39.2ns 1 39.999ns 1 40.0ns 0 40.799ns 0 40.8ns 1 41.599ns 1 41.6ns 0 42.399ns 0 42.4ns 1 43.199ns 1 43.2ns 0 43.999ns 0 44.0ns 1 44.799ns 1 44.8ns 0 45.599ns 0 45.6ns 1 46.399ns 1 46.4ns 0 47.199ns 0 47.2ns 1 47.999ns 1 48.0ns 0 48.799ns 0 48.8ns 1 49.599ns 1 49.6ns 0 50.399ns 0 50.4ns 1

a2

0ns 0 1.599ns 0 1.6ns 1 3.199ns 1 3.2ns 0 4.799ns 0 4.8ns 1 6.399ns 1 6.4ns 0 7.999ns 0 8.0ns 1 9.599ns 1 9.6ns 0 11.199ns 0 11.2ns 1 12.799ns 1 12.8ns 0 14.399ns 0 14.4ns 1 15.999ns 1 16.0ns 0 17.599ns 0 17.6ns 1 19.199ns 1 19.2ns 0 20.799ns 0 20.8ns 1 22.399ns 1 22.4ns 0 23.999ns 0 24.0ns 1 25.599ns 1 25.6ns 0 27.199ns 0 27.2ns 1 28.799ns 1 28.8ns 0 30.399ns 0 30.4ns 1 31.999ns 1 32.0ns 0 33.599ns 0 33.6ns 1 35.199ns 1 35.2ns 0 36.799ns 0 36.8ns 1 38.399ns 1 38.4ns 0 39.999ns 0 40.0ns 1 41.599ns 1 41.6ns 0 43.199ns 0 43.2ns 1 44.799ns 1 44.8ns 0 46.399ns 0 46.4ns 1 47.999ns 1 48.0ns 0 49.599ns 0 49.6ns 1

a3

0ns 0 3.199ns 0 3.2ns 1 6.399ns 1 6.4ns 0 9.599ns 0 9.6ns 1 12.799ns 1 12.8ns 0 15.999ns 0 16.0ns 1 19.199ns 1 19.2ns 0 22.399ns 0 22.4ns 1 25.599ns 1 25.6ns 0 28.799ns 0 28.8ns 1 31.999ns 1 32.0ns 0 35.199ns 0 35.2ns 1 38.399ns 1 38.4ns 0 41.599ns 0 41.6ns 1 44.799ns 1 44.8ns 0 47.999ns 0 48.0ns 1

Data In VPWL Parameters (couldn't screenshot VPWL as it was too long):

for all:

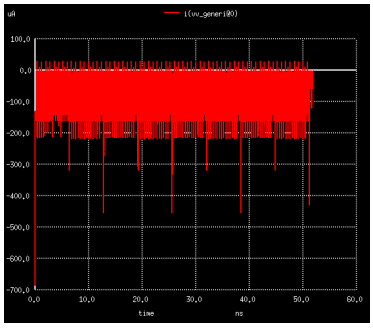
0ns 0 6.399ns 0 6.4ns 1 12.799ns 1 12.8ns 0 19.199ns 0 19.2ns 1 25.599ns 1 25.6ns 0 31.999ns 0 32.0ns 1 38.399ns 1 38.4ns 0 44.799ns 0 44.8ns 1

Commands

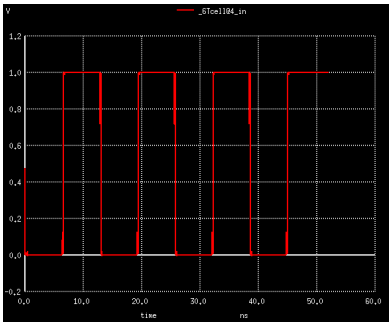
plot I(vv_generi@0)

meas tran yint integ I(vv_generi@0) from=0.0ns to=51.2ns

Plots



I(vv_generi@0) is the current entering the voltage source, which is why it is negative.



Behavior of a random cell in the SRAM. Clearly, it is written to a 0 and then written to a 1 four times.

FOM Metric Measurements - Area

Area of a single bitcell is (6 * 1) = 6

Metrics Table and Calculated FOM

The simulations for these measurements are provided in the previous section.

	Delay	**Power**	**Area**
Case 1 (write 1)	216.6860 ps	4.01385 pJ	6
Case 2 (write 0)	216.6860 ps		
Case 3 (read 1)	225.3387 ps		
Case 4 (read 0)	231.2697 ps		
Worst Case	231.2697 ps	4.01385 pJ	6

*FOM = 60 * BitcellArea * Power * Delay^2 = 7.72860503 * 10^{-29}*

Statement of Academic Integrity

I, Tim Liang, certify that I have complied with the University of Pennsylvania's Code of Academic Integrity in completing this project.