Low Power SRAM

August Bresnaider
ECE 4332
University of Virginia
etc3xg@virginia.edu

ABSTRACT

Having designed 1Mb of low-power SRAM, this document highlights some of the crucial design decisions that came into play, as well as an analysis of critical metrics, including delay, active/idle power, and total area. Lastly, the robustness of the architecture is supported by verification of its viability within large ranges of process, voltage, and temperature variation.

1. INTRODUCTION

While Moore's Law continues to chug along unimpeded, the modern VLSI industry has not been able to support such advancements to their full potential. It is true that smaller technologies can support lower capacitances and higher frequencies, however, it is also true that power issues like leakage current and efficient heat dissipation have acted as large barriers, hindering all of society's processing power. As such, it becomes increasingly important to address such issues. As researchers, we have developed a proof-of-concept SRAM model using Cadence Virtuoso of 1Mb worth of SRAM. To target low power, we have worked to minimize the following metric:

*(Active Energy per Access)^2 * Delay * Area * Idle Power*

We decided to use NC State's FreePDK 45nm technology for this model. While we were unable to make changes to the MOSFET layout, we were able to change the gate dimensions and threshold voltages to fit our needs.

1.1 Specifications

To meet specification, our model met the following criteria:

Table 1. Specifications for SRAM Array

Description	Value
Word Size	32b
Inputs	Addr <14:0>, Read, Write, Clock, Din <31:0>
Outputs	Dout <31:0>
Total Model Size	1Mb

The following section details our design decisions of individual parts of the model.

ARCHITECTURE

1.2 Bitcell

Being that our bitcell is the heart and soul of our SRAM's operation, we spent a great deal of time thinking about the sizing of the different transistor types. The bitcell is a prime example of what is called a "ratioed" circuit, in that the gate widths of the technology included within require a certain ratio to operate correctly. For example: while a large access transistor width may seem desirable in that it would greatly reduce resistance and let the bitline affect the Q value of the cell easier, too large an access transistor can lead to unwieldy amounts of capacitance on the bitlines, slowing down the block's operation dramatically. In our final iteration, we decided on the values as presented in Table 2. In previous iterations, we had settled on much larger access and pull-down transistors, mainly as a fault of our simulations.

Table 2. Bitcell Transistor Gate Dimensions

	Access	Pull-Up	Pull-Down
Width(nm)	150	90	150
Length(nm)	50	50	50

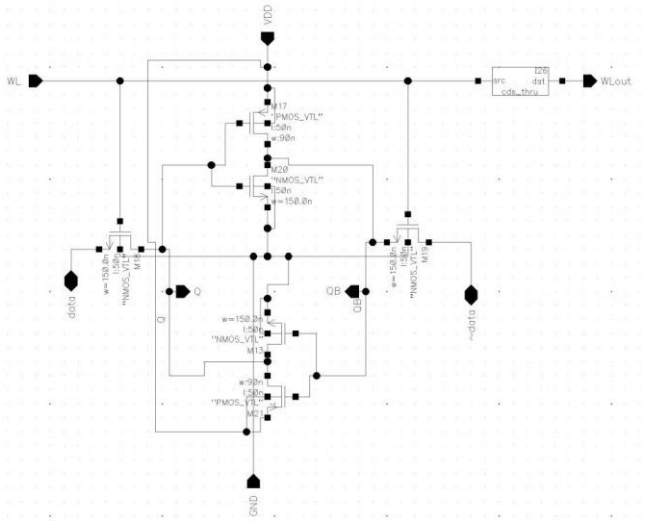


Figure 3. Bitcell Virtuoso Schematic

1.3 Sense Amplifier

To properly output the values stored in our bitcells, we needed a mechanism that would be able to sense differences in bitline voltages and output a value accordingly. While in theory this could be as simple as an inverter, in practice, something of the sort is much too weak and unpredictable, due to near-constant fluctuations in the bitlines, to produce a meaningful output. Instead, we decided on implementing a strong-arm sense amplifier, on account of its functionality as a differential amplifier, as well as its strength in producing a rail-to-rail output[1]. Our implementation can be seen in below:

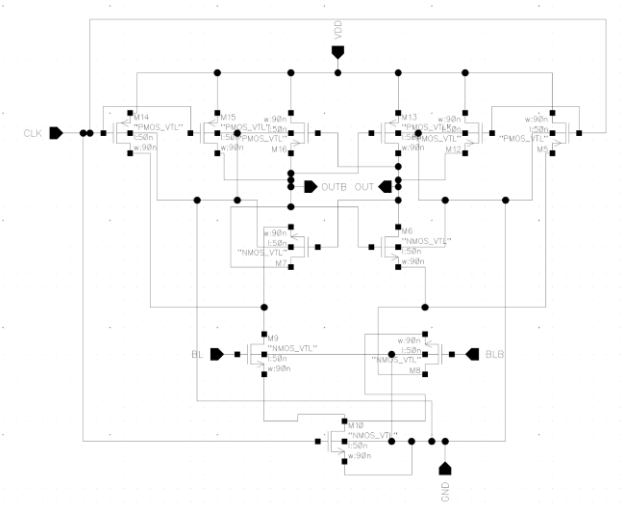


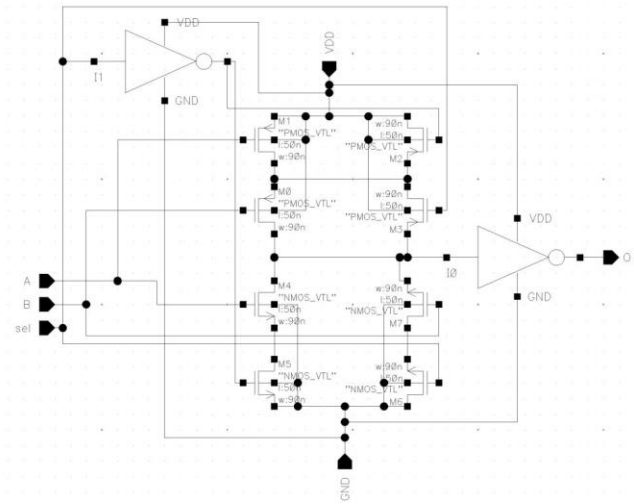
Figure 4. Strong-Arm Sense Amplifier Schematic

It should be noted that, while the enabling input is titled “CLK”, the sense amplifier in our block is driven by (CLK & Read), making this circuit only active during a read operation.

1.4 Multiplexing

Being that there are over 1 million cells that can be read from or written to, it is of crucial importance to have dedicated hardware assigned for the sole purpose of signal routing. This section and the next are dedicated to multiplexing and demultiplexing, respectively, which are circuits designed to select between multiple inputs or outputs. To begin, we will talk about the multiplexer, or mux. The job of a mux (specifically a 2-to-1 mux) is to decide whether to output value “A” or value “B”, as specified by a selector bit, which we will call “sel”. While the below figure may seem complicated, this simply represents static CMOS logic for O = (A& ~sel) | (B & sel). This is used in our circuit to decide which block’s output to use, or to pick which word to read in a block, when they share a wordline.

Figure 1. 2-1 Mux Schematic

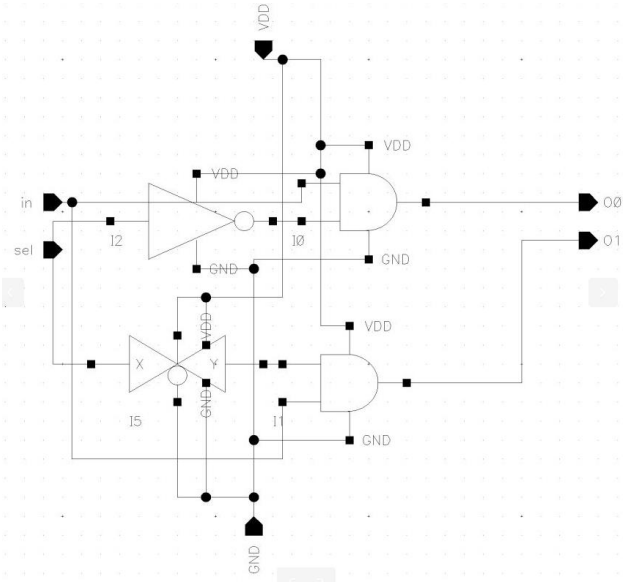


While this is a small example, this simple schematic can be chained and cascaded to create larger muxes. The largest we chose to use in our design ended up being a 32-1 (5 bit) mux.

1.5 Demultiplexing

Where a mux takes multiple inputs and sends one to the output, a demux does the opposite, taking only one input, and sending it to one of many outputs. Like with the mux, this can be easily chained to make larger circuits.

Figure 4. 1-Bit Demux Schematic

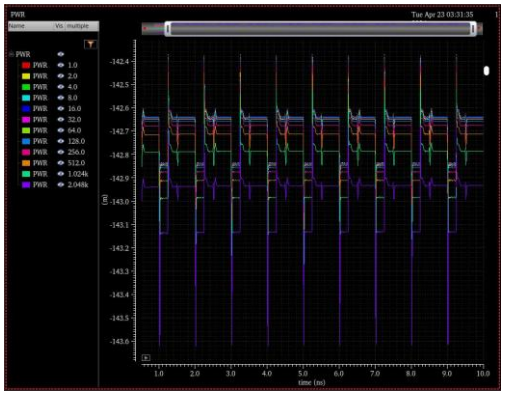


The largest we chose to use in our design ended up being a 6-bit demux, which works to drive the wordlines in our blocks.

1.6 512x64 Block

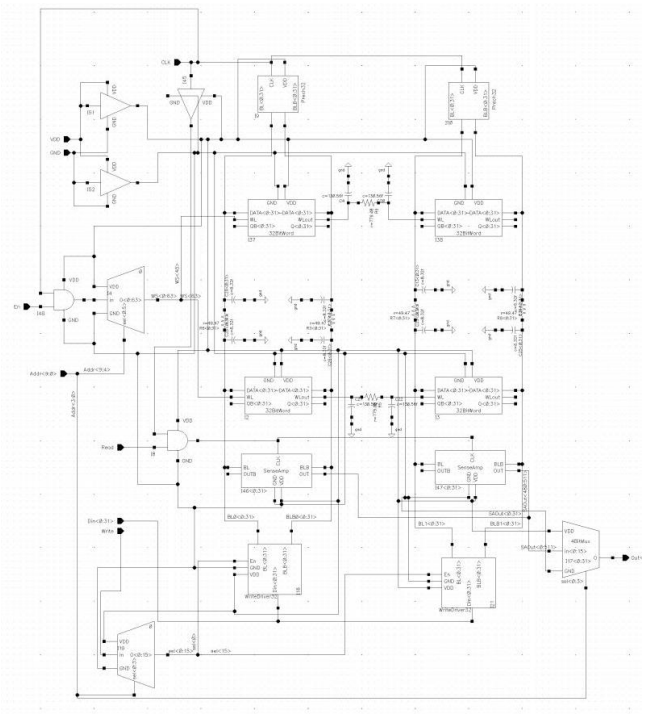
To keep our cells organized, it’s vital to place them in blocks that we can individually address. The sizes of these blocks is highly specific, as too small of a block leads to a large part of the block being dominated by write drivers, sense amplifiers and routing circuitry, where too large a block can lead to extreme delay due to interconnect capacitance and Elmore delay. As such, simulations were run much like the one pictured in figure 5 to weigh the power draw of n cells on the bitlines and wordlines.

Figure 5. Wordline Power Simulation



From these simulations, it was decided that a block would be built of 512 bits per row, and 64 bits per column. The below figure shows a simplified schematic of said block, though due to limitations in simulation, we were unable to include all cells in the schematic, requiring us to make equivalent RC pi models to behave as cells.

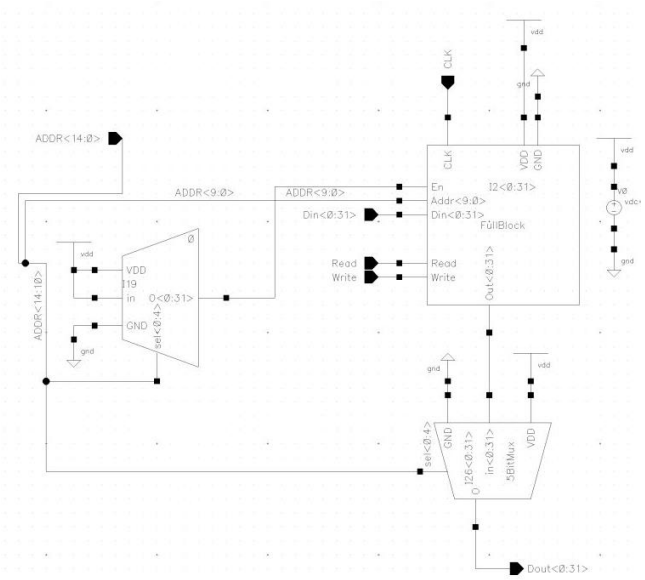
Figure 6. Full Block Schematic



1.7 Full Model

Finally, we integrated all the previous parts together to generate a full model. This includes 32 blocks, adding up to 1Mb in total. The following sections assess the metrics of said model.

Figure 7. Full SRAM Hierarchy Schematic



2. METRICS

2.1 Area

We found our bitcell area by adding up all the gate widths of all transistors in the model. This gave us the following figure:

Total Area: 27.49 mm

2.2 Delay

We were unable to calculate the delay, so we instead simulated it in a working block. Below show the propagation delays of all our read / write circuitry, respectively:

Figure 8. Read Propagation Delay Simulation

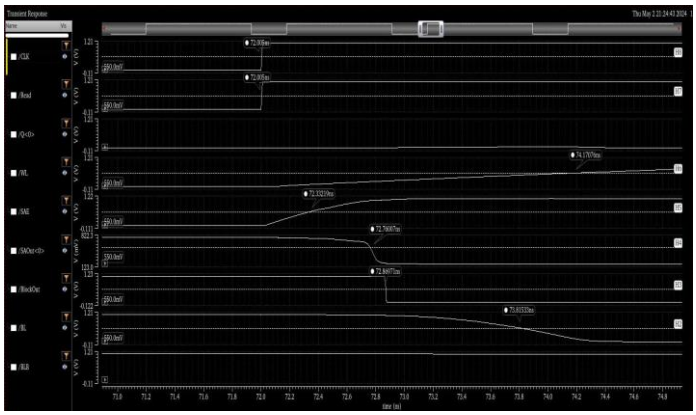
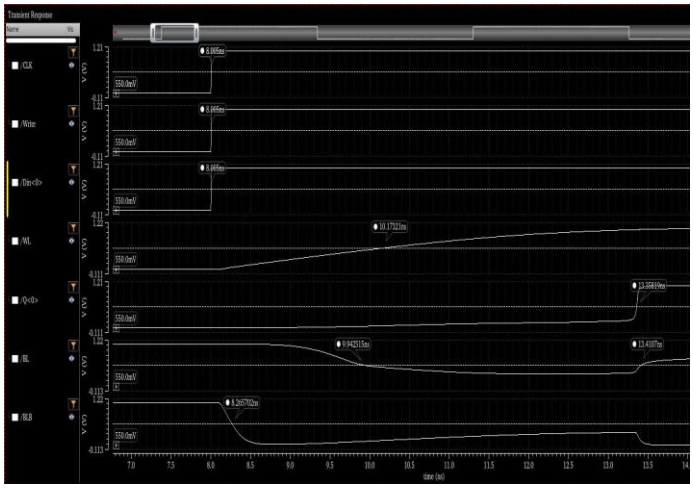


Figure 9. Write Propagation Delay Simulation



From these, we collected the following:

Read Delay: 0.865ns

Write Delay: 5.353ns

Total Delay: 5.353ns

Maximum Frequency: 186.811MHz

2.3 Energy

By reading the current and voltage draw from VDD for each block, we were able to establish values for idle, read and write power, and extrapolate to the entire model, as notated below:

Idle Power: 32*(1.1*0.352) = 12.39mW

Write Power: 27.0514mW

Write Energy: 27.0514mW * 5.353ns = 144.5318pJ / access

Read Power: 22.8734mW

Read Energy: 22.8734mW * 5.353ns = 122.4413pJ / access

Total Energy: [5(122.4413pJ)+1(144.5318pJ)]/6 = 124.2897pJ/access

2.4 Summary

Metric	2.816*10^-29 J^2*sec*mm*W
Total Area	27.49 mm
Read Energy	122.4413pJ
Write Energy	144.5318pJ
Total Energy	124.2897pJ
Read Delay	0.865ns
Write Delay	5.353ns
Total Delay	5.353ns
Idle Power	12.39mW

3. CONCLUSION

Looking at our final metric, we feel accomplished in that we were able to do what we set out to: we were able to develop a fast, low-power SRAM model that adheres to the previously laid out specifications. If one were to continue to pursue this architecture, it seems like decreasing write delay would go a long way to bettering the metric. Namely, it may be worth looking into having 256 cells per row, as the main time-sink came from charging the wordlines to enable a write operation. There is also a large deal of optimization that may come from using a differing layout: we used FreePDK45 throughout, but if one were to design their own MOSFETs, there would be potential for large power savings, as well as better accuracy in simulation figures. As previously stated, we were able to simulate our model across different process variations, modeling the SNM, RNM and WNM plots in the TT, FF, SS, FS, and SF process corners, as well as with voltage variation (VDD = 1.1V, VDD = 1.21V, VDD=0.99V), and at temperatures of 0C, 27C and 50C.

4. ACKNOWLEDGMENTS

Our thanks to Professor Calhoun as well as Anjali Agrawal and Aki Takana for their guidance and feedback throughout the semester. Their support and dedication to their field inspired us time and time again to keep moving forward.

5. REFERENCES

[1] B. Razavi, "The StrongARM Latch [A Circuit for All Seasons]," in IEEE Solid-State Circuits Magazine, vol. 7, no. 2, pp. 12-17, Spring 2015, doi: 10.1109/MSSC.2015.2418155.

6. APPENDIX

Model Area Calculations

Bitcell Area: 2 *150nm + 2* 90nm + 2* 150nm

Total Bitcells Per block: 512 * 64

Total Precharge per block: 512 * 2 * 90nm

Sense Amp Area: 11 * 90nm

Sense Amps Per Block: 512

Row Demux: (16 * 64 + 8) * 90nm

Column Demux: (12 * 16 + 8) * 90nm

Block Demux: (14 * 32 + 8) * 90nm

4 bit Mux: (15 * 16 * 16 * 90nm)

5 bit Mux: (31 * 32 * 16 * 90nm)

Total Area: 27.49 mm