First G. Eddie, Second S. Kris, *Member, IEEE,*

ECE 471: Final-SRAM Project (March 2014

Abstract—**Overview of SRAM creation. Paper of design and analysis of making a 256 bit memory cell with all blocks shown and explain in detail.**

I. Introduction

Most if not all modern CPU's have a large portion of their die dedicated to memory. Intel CPU's have on the order of 3-8mb of on die cache that is needed to speed up operations since almost all operations are dependent on manipulating memory leading to a bottleneck if the memory is not fast enough. The moral is the more memory on the CPU the better to a point. This overview of how to make an SRAM array takes in the considerations of wire delay, maximum performance while conserving power, and using minimum sized SRAM cells to allow for the minimum area needed for the ram cells. Towards the end of the project using these small Sram cells made it so the bit lines had to be precharge to Vcc to allow for proper pull up and down of the line by the minimally sized Sram cells. The Sram array consists of the following blocks. The Sram Cell, 16 Bit Decoder, Sense amplifiers, inverters, Tristate buffers, and Vdd precharge block.

II. Synopsis of skimming papers

A. Ultra Low voltage Cmos/ Razor / Sensor network

With increasingly mobile devices voltage scaling is effective in decreasing the power consumption quadratically. For optimal power saving benefits the voltage must be lowered to a subthreshold level, which can be adversely affected by V threshold variability. This region which these transistors operate at is in the subthreshold regime. Energy consumption can be reduced by factors of 20 times by making simple adjustments in the FET channel length to reduce variability of Vth. This Vmin is found by reducing designs Vdd voltage until operation stops or becomes to slow and not worth the performance drop. The true value for the voltage that should be strived for is one that allows for the most operations per unit energy. This can be derived with equations containing Boltzmann's constant, thermal voltage, and temperature. Basically if you use the absolute Vmin the processor will run at a much lower frequency and not actually be computing much. The key is to find where computation speed intersects with the power being used in the circuit. Other examples are having a multicore architecture with greater space then run each at half the speed and voltage and you have a processor that uses ¼ of the energy while being able to compute the same amount of data. This leads to the fact that you can have an extremely power efficient multicore architecture, but most software is not parallelized, with many problems not able to become paralyzed. Noise margin can be high in subthreshold processors since the voltage signal in the cpu is already low.

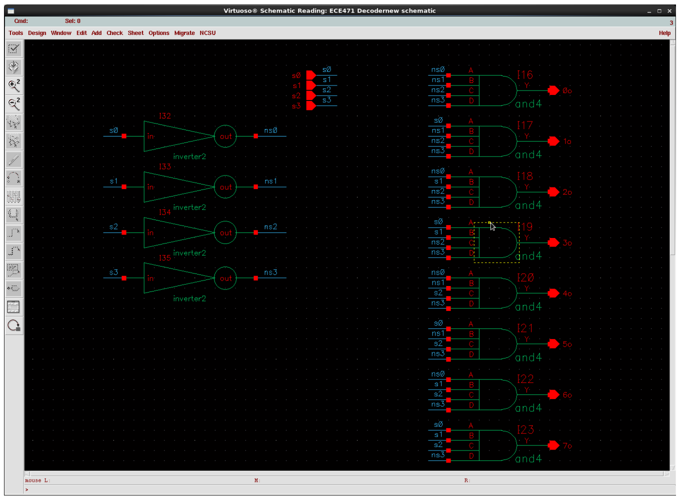
SRAM running in sub threshold from 1.2v to 200mv with power as low at 2.6pJ/per instruction. Reduction in Vdd from 1.2v down to the 200mv causes a decrease in speed by 18% while yielding much higher performance per watt. This is increasing the leakage current in retrospect to total dynamic and static current, but the percentage is still less than 33%.

In the past 7 years mobile phones have had their battery life increase by 50 times. Running the mobile phones processor at much lower frequency allows for the power saving characteristics of sub-threshold operation. Playing a video though needs much higher performance than playing an mp3 file, so allowing for dynamic frequency based on load and reduction in transistor size is how phones get the battery life that they do today. Problems when running these cellphone processors at low frequency are from reduction in power supply network voltage, temperature fluctuations, gate length, doping inconsistencies and the list goes on. Corner analysis can test if processors can operate correctly when these variations occur. Things like Monte Carlo runs. Calculating the worst case and ensuring margins are built into the circuit to allow for these inconsistencies. Things like temperature induced delay in inverters and local voltage variations cannot be completely accounted for in these cases so Razor has a new approach. The overview of Razor is to clock the processor as low as possibly possibly with 1-2% error, but then catch any errors that occur with dynamic detection. Say that a cpu can run at 1v with no errors, then at 0.7v with 1% errors, this will have a quadratic reduction in power consumption, while still processing almost as much data. This is idea of Razor that if these errors can be caught with some extra gate logic you will have a much lower power CPU.

III. Operation and Analysis of SRAM Array

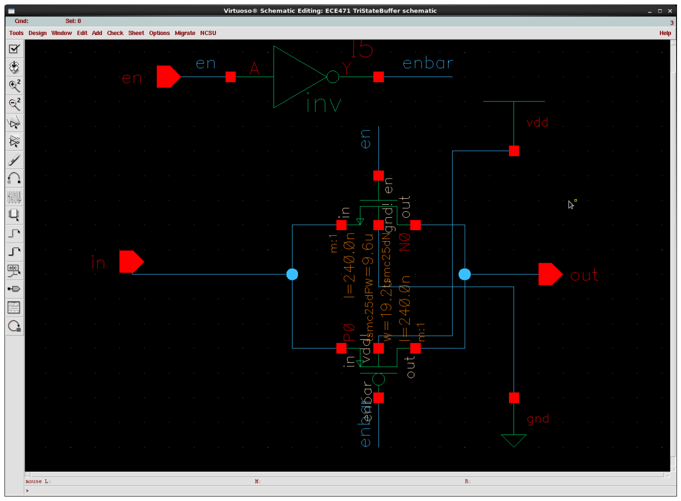
Operation of SRAM Array

The desired operation of the SRAM array is to have an SRAM cell which when the word line is high and a value is forced to the Bitlines by a Tristate buffer it will create a value being looped through the inverters on every clock cycle. Then when the word line goes low we have a value in the SRAM cell. The get the value out of the SRAM cell be have to first close the bitline off to all inputs or driving elements so it's floating. We then precharge the Bitline to Vdd with the Pullup transistors to allow for the small SRAM cells to drive the floating line. Then just after we stop the Vdd pull up we pulse the word line from the decoder which causes the floating bitline to be driven by the looping inverter circuit inside the Sram cell, which is opened to the bitlines by transistors connected to the Wordline. One the bitline is driven by the SRAM cell we can use the Sense Amplifier connected to the bitlines to take any charge on the line amplify it to one of the rails giving us a consistent 2.5v or 0v value. This is the basic overview of the SRAM operation, with the decoder being needed to both isolate the wordlines from the inputs, and allow for a 4 bit bus input instead of the need of a 16 bit bus. This can simplify layout as well as the architecture and programming of the cpu, since the logical portion of the processor will probably have a full adder with the 4 bit output to the Sram for selection.



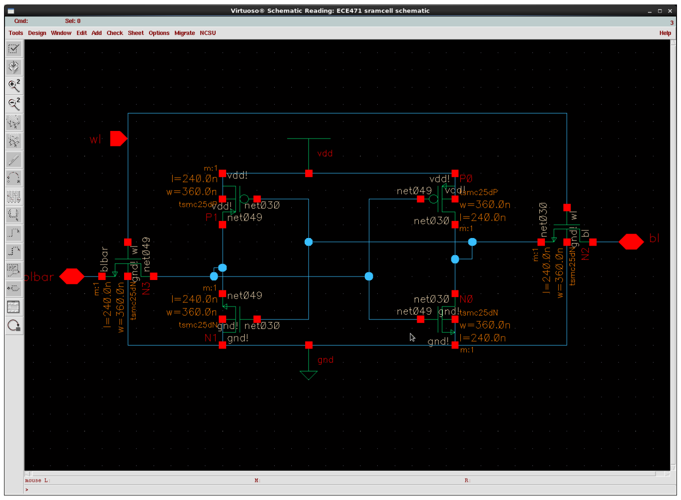
16 bit Decoder

The 16 Bit decoder functions as both a sort of buffer to the word line as well as allowing only a 4 bit input for driving 16 word lines. The decoder consists of 4 inverters and 16 4 input and gates which are sized Nmos 960nm and Pmos 1920 nm in order to drive the word lines effectively. This will make the area of the decoder larger than if it was minimally sized but this allows for a super fast clock rate since the word lines can be charge fast with larger and gates.


Virtuoso® Schematic Editing: ECE471 TriStateBuffer schematic
en
enbar
inv
in
out
vdd
gnd

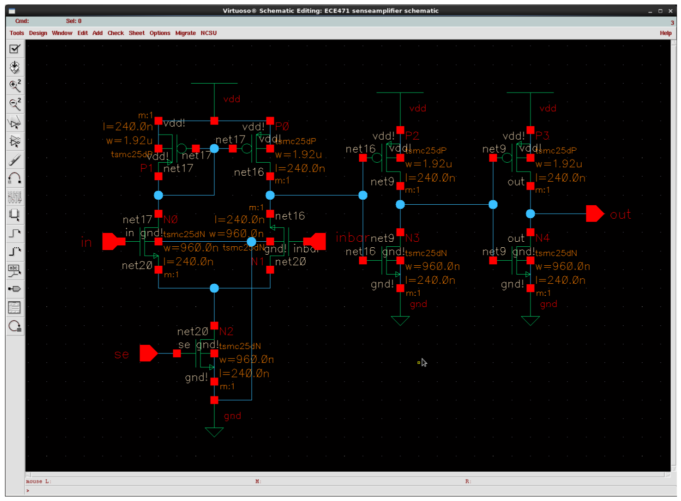
Tri State Buffer

The sizing are 960nm for the nmos and 1920nm for the pmos, since it has to drive the bitline and sram cell. The Tri State is using a pass gate design with an enable just tied to the gate of a couple mos transistors. The Tri State is used to buffer the bitlines from the input and the inverter for the bitline bar. These turned off makes the bitline floating allowing for the reading of cells on the line. Then the word line can be turned on, then the sense amplifier turned on to read the charge on the line.



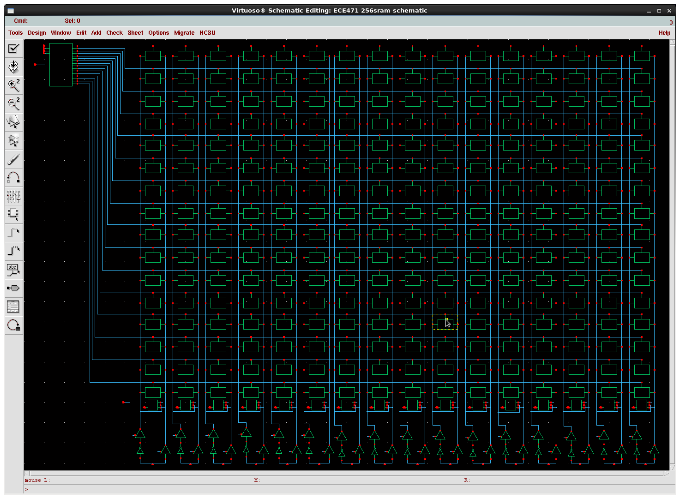
Sram Cell

Firstly the sizing is minimum sizing at 360nm for the width on both the pmos and nmos. The SRAM Cell is made from 6 transistors. The two inverters in the middle create a feedback loop that stores the data and the two on the side allow for the data to be access when reading and writing occurs. If the two transistors on the sides connected to the word line were not there then the data would interfere with data in other cells on the same bit line. Basically to read set the word line high then set the bitline to the value of the data 2.5v or 0v then the bitline bar to the be inverse. Then you tern off the word line and the data is saved in the memory. To read from the cell you must precharge the Bitline to Vdd, then make sure that the bitline is floating not connected to anything driving it and set the word line to high. The transistors will either pull the Bitline high or low depending on the value that was saved in memory.

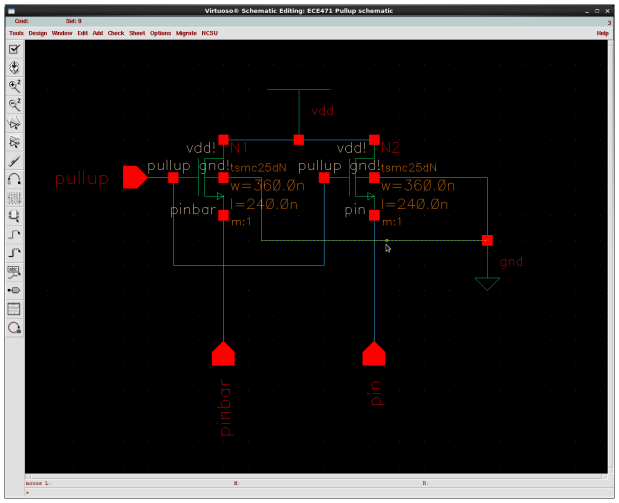

Virtuoso® Schematic Editing: ECE471 senseamplifier schematic
in
inbar
se
out
vdd
gnd

Sense Amplifier

Same sizing as the Tri State Buffer, 960nm, and 1920nm. The sense amplifier basically is just a half opamp design it seems. The Sense Amplifier's job is the take the voltage on the line that was driven from the memory cell and push it to Vdd or pull it down to ground. The Sram cell can only drive the Bitline to 2v in my design from the Vth of the transistors connected to the word line. The sense pulls and pushes the voltage to 2.5v and 0v respectivly.

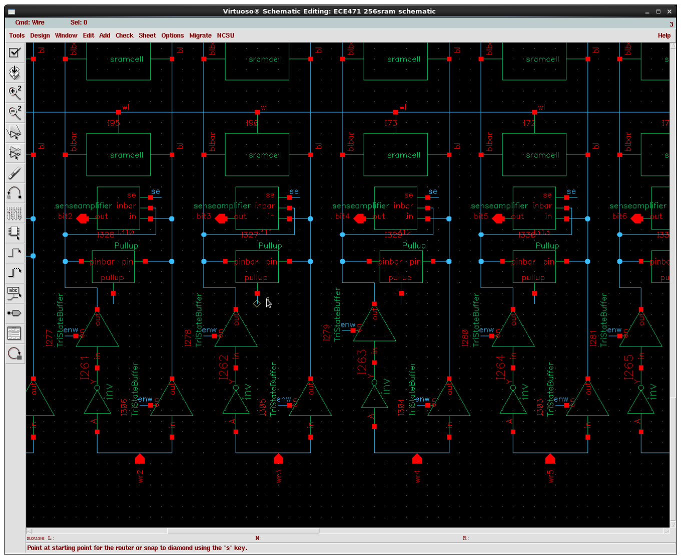


Top Figure



Vdd precharging Circuit

Pulls up the Bitline just before reading.



Bottom Figure

256 Bit SRAM Array

The picture on top is the full Sram circuit. Bottom picture shows the two Tristate buffers on the bottom with an enable. There is also an inverter sized at 960nm nmos and 1920nm pmos for the bitline bar. Above this is the precharge block with the three inputs that is precharging the line before each read, since the minimally sized mos in the SRAM cells have trouble pushing the bitline up and down. Lastly the Sense amplifier is the 3 input one output block above this that is connected to both lines and can be enabled when we want to see the value that is on the bitline when reading the SRAM cells.

IV. Calculations and Answers

Calculations

The estimated leakage power is calculated from the Vdd voltage and current when the circuit is not switching just on. When taking the integral over a small time frame I got the dynamic power to be 5mw, this is during the switching period of the memory cells. When measuring the static power by looking at the Vdd voltage and current and integrating I got 4uw about 3 orders of magnitude less than the dynamic power. This sounds reasonably. Probably just from the almost negligible wire delays and resistors from the attached pie model.

The maximum clock frequency is shown in the screenshot of the scope diagram. To calculate the max frequency I wrote an hspice sim_deck, the one attached in the Appendix, that increased the reading and writing speeds until the Sense Amplifier was not able to detect a high or low value from the bit lines. I think what is limiting the clock speed is having minimally sized Sram cells that can't move the bit line fast enough. At least that is what I think is limiting the speed of the Sram. The value I calculated for the maximum frequency of writing then reading to the SRAM is 1/(2ns) approximately 500mhz.

Area of the total design is calculated by using the size of each transistor based on the widths and process technology, and then adding 25% for wiring and layout spacing.

SramCell = 360nm*6 *(16 * 16) = 552.960um

Decoder = AndGate(16 * (6(ntransistors) * 960nm + 3ptransistors*1920nm)) = 184.320um

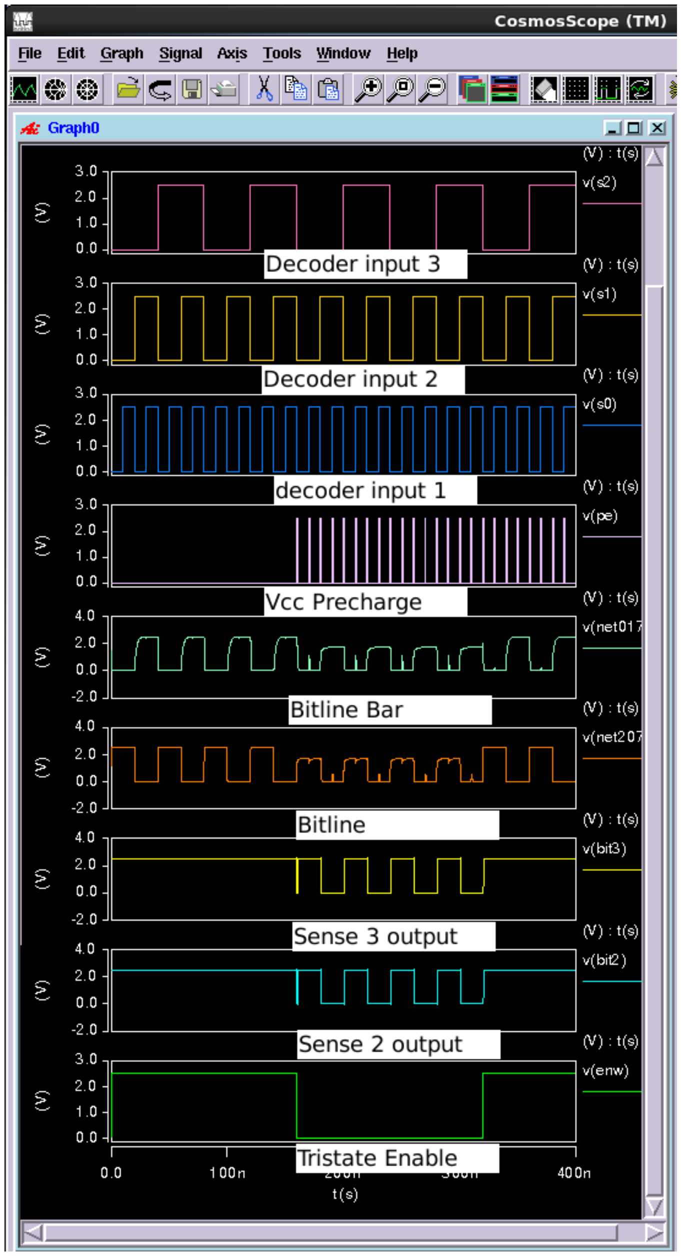
Tri State = Passgatedesign(960nm*2) * 32 = 61.440um

Sense Amp = (4 * 960nm + 3*1920nm) * 16 = 184.320um

PullUp = (2 * 360nm)*16 = 11.520um

Total Area = (994.56um) + (0.25% * 994.56um) = 1243.2 um

Noise margin for the SRAM Array at the max clock is 10% of Vdd. So I may actually be able to clock this circuit faster and still have an ok output if the Sense Amplifier can detect the change on the bitline.



256 Bit SRAM Array Waveforms.

The figure above shows reading and writing of the memory. The first 160ns is the writing phase where I enable the Tristate and write to each cell in memory. This is done by the sweeping the decoder out puts with the wave forms above writing to all 16 bits per Bitline. The bitline is getting amplified by the Sense Amplifier to get the square output of the memory shown above. The difficult part of this project for me was not so much the building of the circuit or cadence design, just making the netlist in Hspice to test the circuit. In order for the precharging of Vdd to work on every bit I would have to have a brief period in between reads where I have a time to precharge Vdd. With more practice at Hspice I should be able to read write sequentially. This waveform should be enough to prove that the memory works. I don't have the Hspice I used to calculate the clock frequency.

ACKNOWLEDGMENT

Patrick Chang for teaching the class. Neil Glover and the other TA for answering questions in lab, and dealing with problems with Cadence.

REFERENCES

[1] G. O. Young, “Synthetic structure of industrial plastics (Book style with paper title and editor),” in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.

[2] W.-K. Chen, *Linear Networks and Systems* (Book style).Belmont, CA: Wadsworth, 1993, pp. 123–135.

[3] H. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1985, ch. 4.

[4] B. Smith, “An approach to graphs of linear forms (Unpublished work style),” unpublished.

[5] E. H. Miller, “A note on reflector arrays (Periodical style—Accepted for publication),” *IEEE Trans. Antennas Propagat.*, to be published.

[6] J. Wang, “Fundamentals of erbium-doped fiber amplifiers arrays (Periodical style—Submitted for publication),” *IEEE J. Quantum Electron.*, submitted for publication.

[7] C. J. Kaufman, Rocky Mountain Research Lab., Boulder, CO, private communication, May 1995.

[8] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interfaces (Translation Journals style),” *IEEE Transl. J. Magn.Jpn.*, vol. 2, Aug. 1987, pp. 740–741 [*Dig. 9th Annu. Conf. Magnetics* Japan, 1982, p. 301].

[9] M. Young, *The Techincal Writers Handbook.* Mill Valley, CA: University Science, 1989.

[10] J. U. Duncombe, “Infrared navigation—Part I: An assessment of feasibility (Periodical style),” *IEEE Trans. Electron Devices*, vol. ED-11, pp. 34–39, Jan. 1959.