

Project Report EE213, Spring 2015/16

Group Members:

Kevin Kinningham (kkinningh)

Nipun Agarwala (nipuna1)

Design Objectives:

Our design objective is to minimize the expected total energy usage for read and write while keeping the 99.7 percentile delay less than 1ns. We can formalize our design objective as follows:

Minimize

$$E[P_{\text{write}} + P_{\text{read}}]$$

Where

$$P(T_{\text{read}} < 1\text{ns}) > 0.997$$

$$P(T_{\text{write}} < 1\text{ns}) > 0.997$$

Note that we only used delay as a constraint, not an optimization target.

1. Summary of what ideas you explored. Please list the parts of the memory that you worked on for the project, and what you tried to do.

Decoder: *Reduce the energy dissipation by optimal downsizing.*

Decoder: *Found the marginal delay/performance trade-off from changing V_{dd} and resized the decoder to match this marginal cost.*

Decoder: *Investigated the effects of variation on delay*

SenseAmp: *Investigated the relationship between delay and power for the sense amp*

R/W Assist: *Investigated the effects of variation on V_{min} and delay.*

R/W Assist: *Used redundant columns to improve both V_{min} and three sigma delay*

TimingGen: *Use a simple replica to generate SAE. Also generated SAPC and PC to allow fast cycle times.*

2. Design specs:

Operating Voltage	664mV
Min Voltage Read	613mV
Min Voltage Write	573mV

FO4 in ps	36.697
------------------	--------

3. Performance Results (give delay in both ps, and FO4 delays).

Note: Worst case column is with three sigma variation decoder, three sigma variation in overall memory core (4.2 sigma for an individual cell), and 100mV offset in sense amp.

	Delay (FO4)	Delay (ps)	WC Delay (FO4)	WC Delay (ps)
Clk (or address) -> WL rise	9.21	337.9	9.21	338.047
WL rise -> Bit Line split	10.9	398.59	12.1	444.10
BitLine -> SenseAmp Out (inc mux)	5.80	212.85	5.82	213.579
Total access time	25.9	949.34	27.1	995.726
Min Cycle Time		1.053 GHz		1.004 GHz

	#Memory access/cycle
Final Throughput	1

What constrained the minimum cycle time? Which phase was critical (e.g. bitline precharge, SAE evaluation, etc)?

We didn't optimize for delay, so we didn't really have a "critical path" that constrained the cycle time (delay was simply a constraint; we could have gotten a much faster cycle time at 1V).

The phase that prevented us from reducing our supply voltage further was the memory core bitline evaluation. Lowering Vdd increased the delay of bitline swing, which meant the decoder had to use more power to fit the 1ns total cycle time constraint. This ended up increasing our overall power usage.

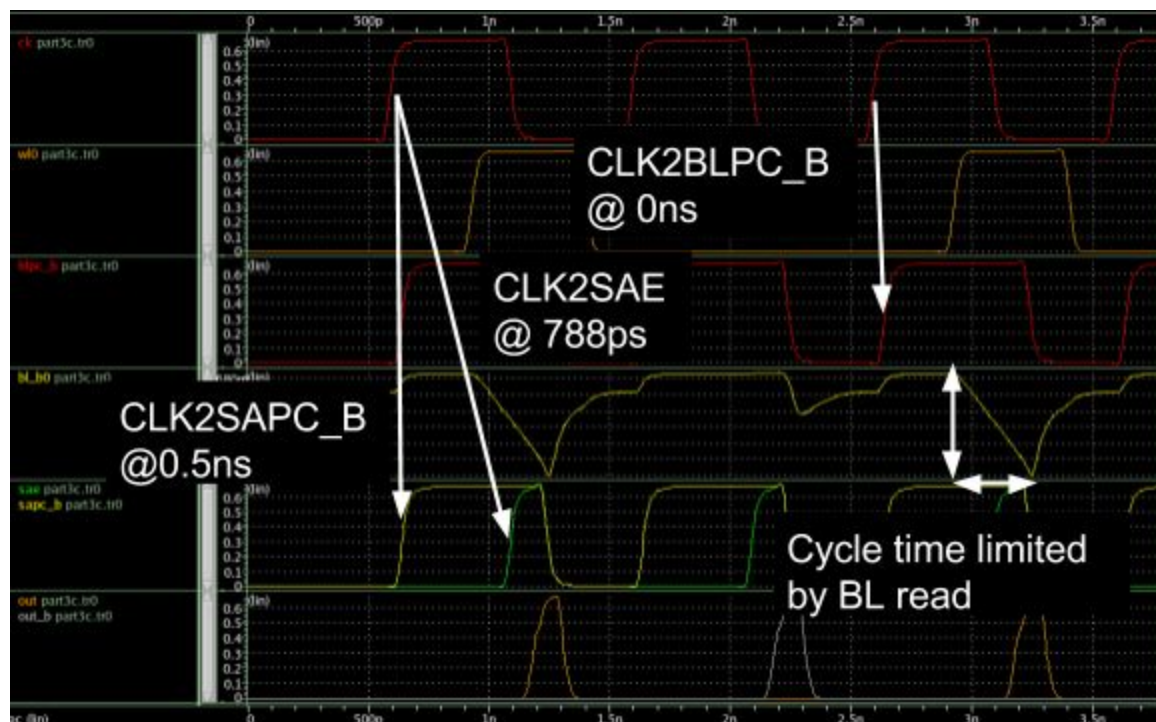
4. Energy per access: For leakage energy per access, integrate the leakage current over the minimum cycle time.

	Gate (W)	Clock (W)	Leak (W)
Decoder	7.5956e-05	5.5568e-06	1.4322e-06
Memory Core	1.7443e-04	6.2032e-04	1.7399e-04
Sense Amp	1.1287e-04	8.3697e-05	6.6400e-07
Total	3.6325e-04	7.0957e-04	1.7608e-04

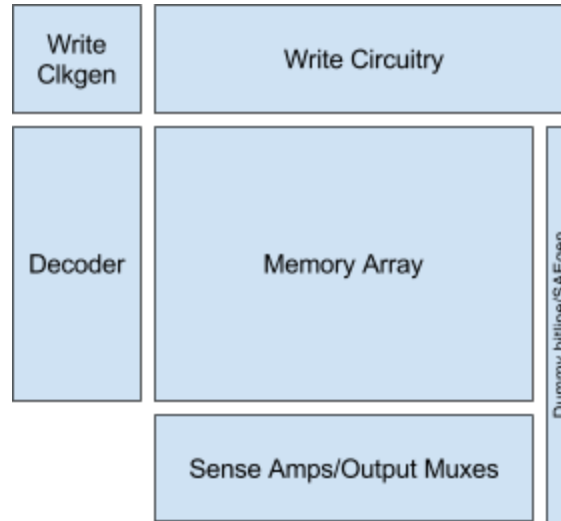
Explain how power scales with f_{clk} .

Assuming we don't change the design, if we increase the cycle time the power consumption will actually go up due to additional leakage. However, if we can redesign the circuit (or power down) power will go down as the cycle time increases (mostly due to less aggressive sizing and reductions in operating voltage).

5. Overall Timing Diagram: Draw all the major clocks you generate (SAE, PC, SAPC, etc) and show with arrows which signals cause each transition. Also show on this diagram what limits the min cycle time.



6. Basic Floorplan: Please draw a floorplan of the chip indicating where all the parts are located. Rough size estimates for the memory (at least the memory array) and some regular structures.



	Size Estimate (um ²)
Memory Array	0.418176

7. Design Description

7.1 Decoder Optimization

7.1.1 Describe what you hoped the change would accomplish

We hoped to optimally size the decoder for both power, while meeting the max total system delay requirement of 1ns and while taking into account 3 sigma variation of the entire decoder. We additionally hoped to investigate how changing the voltage for the entire system changed delay and find the optimal Vdd to use for the entire system.

Mathematically, we say:

Minimize

$E[\text{decoder}]$

Subject to

$P(T_{\text{decoder}} < 1\text{ns} - T_{\text{cell_read}} - T_{\text{senseAmp}}) > 0.997$

$P(T_{\text{decoder}} < 1\text{ns} - T_{\text{cell_write}} - T_{\text{senseAmp}}) > 0.997$

7.1.2 Describe what the change actually accomplished

Under our objective, we were able to get a much lower power than initially calculated in previous sizings and models.

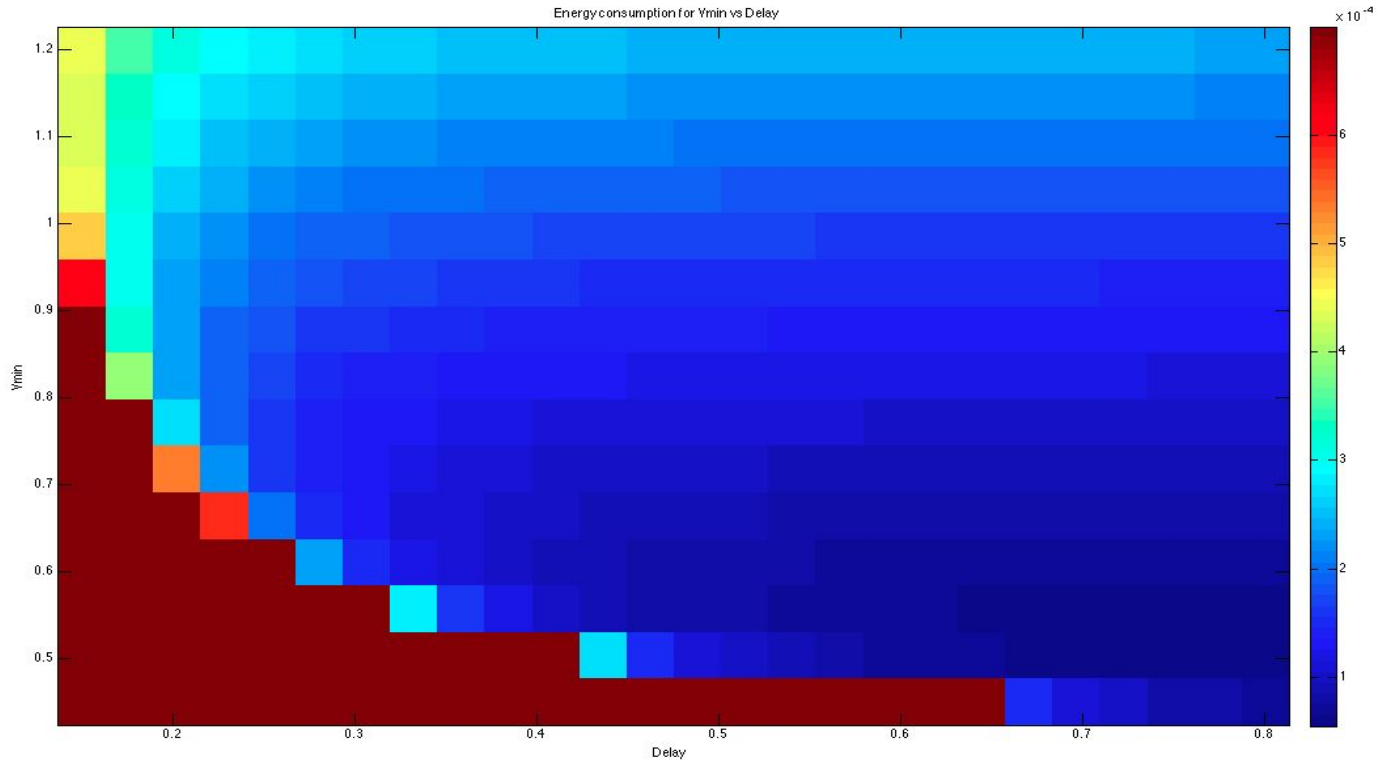
7.1.3 Describe how you implemented this change

We modeled the decoder sizing as a convex optimization problem and used the CVX matlab package to get power-minimal sizings for every reasonable combination of V_{min} and delay (plots in 7.1.4). From this, and our experiments with the sense amplifier and memory cell (7.2 and 7.4), we found the V_{min} that minimized the overall power of our design, while still keeping the total delay less than 1ns i.e. using the 1GHz clock. This turned out to be about **664mV**.

Using this as our V_{min} and the delays found for the sense amplifier and memory cell, we reoptimized the decoder sizing (again using CVX) and simulated the resulting sizings to confirm our model matched HSPICE. Once this was confirmed, we measured the resulting delay and power.

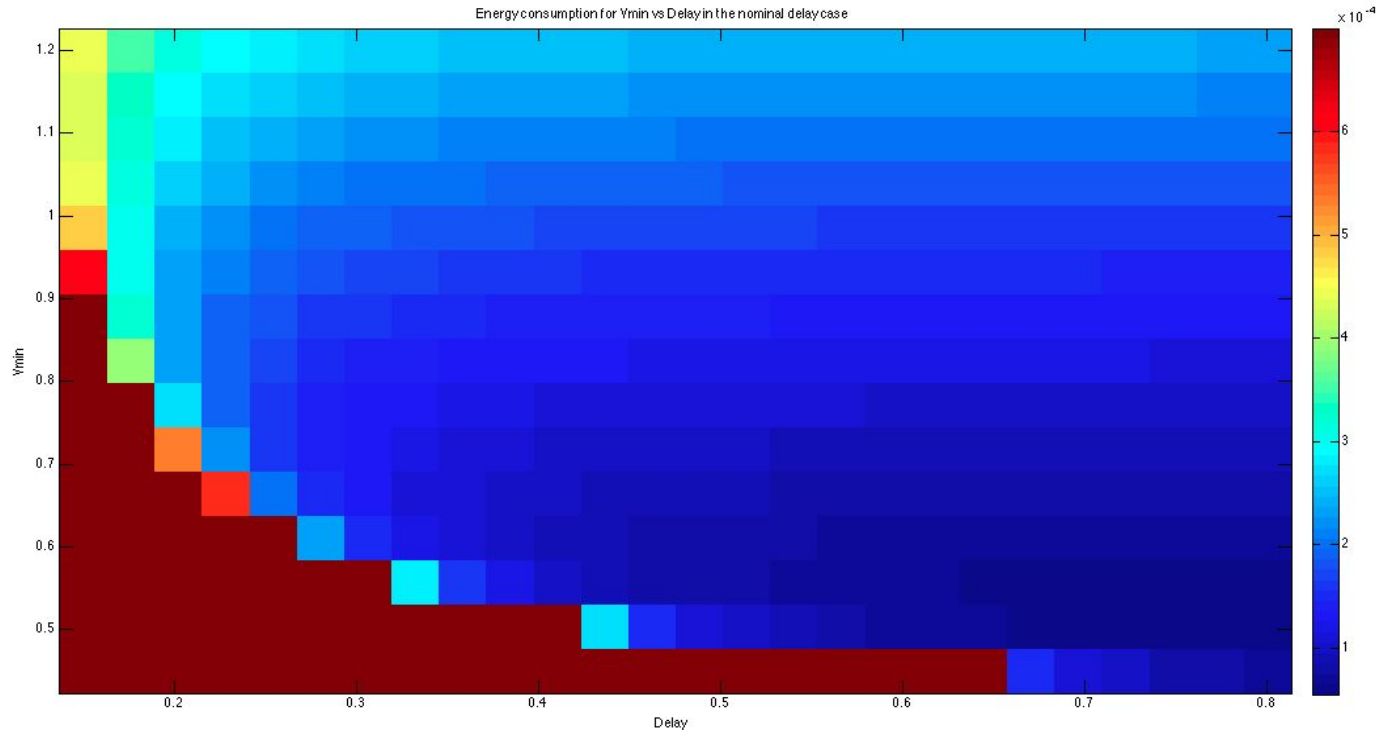
7.1.4 Performance and power of your solution

Our optimal value of V_{dd} is 0.664 V. For that, we saw that the HSpice simulations give us a **delay of 337.9ps**, which is close to **2x worse than the 186ps delay we calculated in Part 2** of the project. For energy, HSpice calculations gave us a **power of 0.07595 mW**, as compared to the **1.1mW** needed for the designing in Part 2 of the project. **This is more than a 15x decrease in power**



Energy HeatMap for Voltage (in V) vs Worst case Delay (in ns)

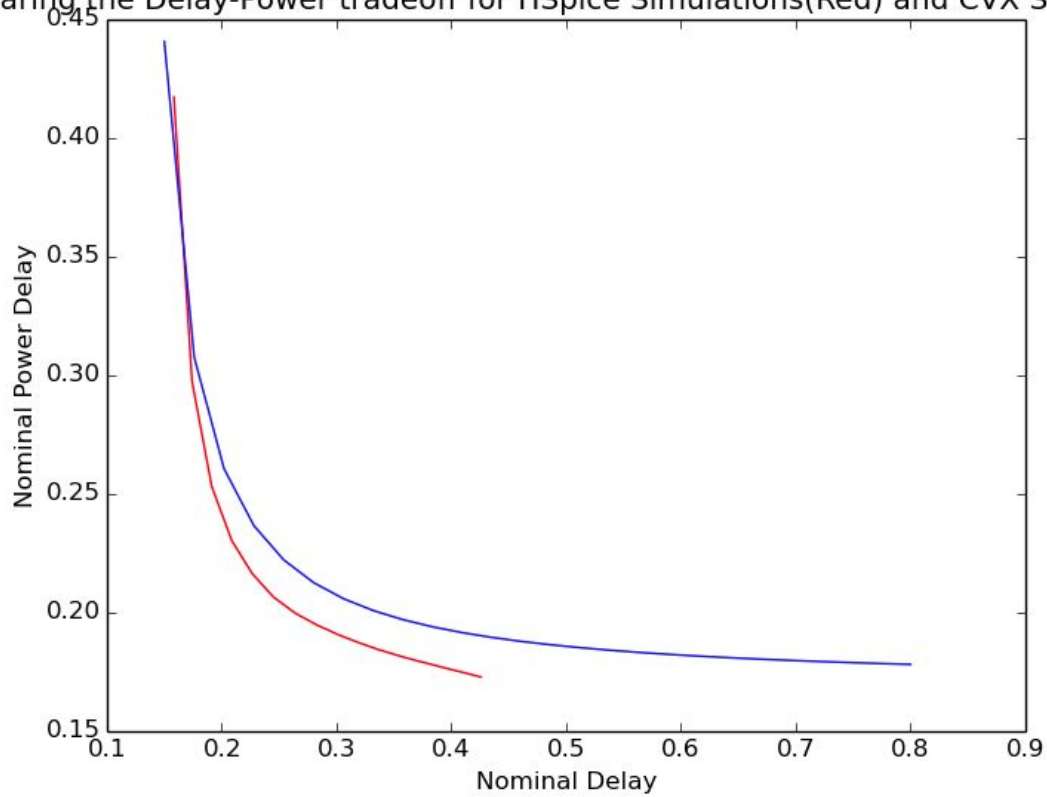
In the above diagram, we plot the power consumption for the optimal sizing, given the constraint of a delay and under a particular voltage. Sizing of the gates is done for the worst case i.e. variation of 3 sigma is taken into account so that a 99.7% coverage is obtained.



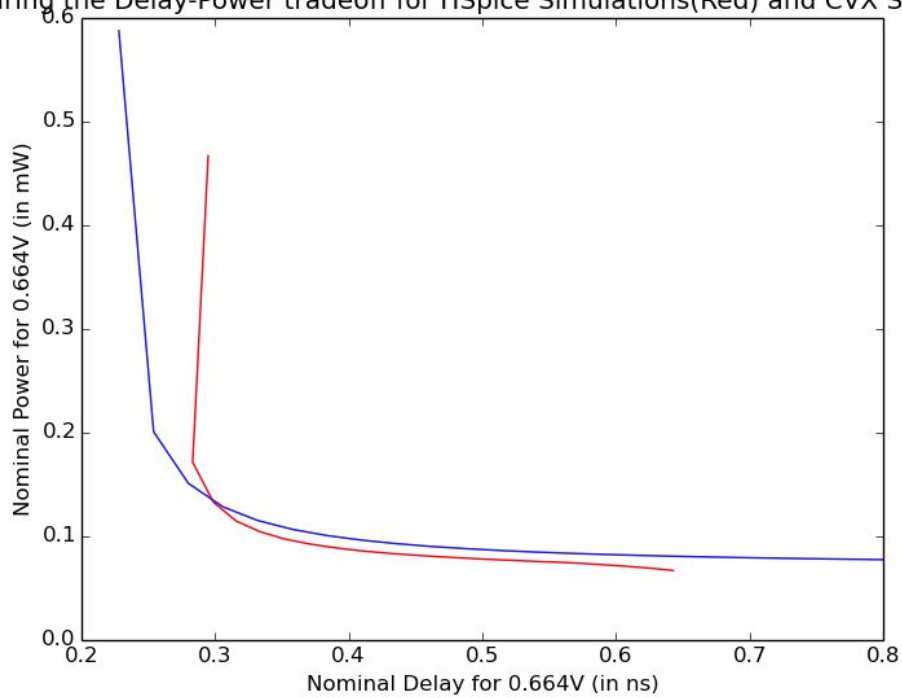
Energy HeatMap for Voltage (in V) vs Nominal Delay (in ns)

In the above figure, we plot the power consumption for the optimal sizing, given the constraint of a delay and under a particular voltage. This allowed us to choose our Vmin and delay constraint for the decoder. Note dark red denotes infeasible designs.

Comparing the Delay-Power tradeoff for HSpice Simulations(Red) and CVX Sizings(Blue)



Comparing the Delay-Power tradeoff for HSpice Simulations(Red) and CVX Sizings(Blue)



The above figures compare the actual and predicted power vs delay of the sizings found using CVX at a fixed voltage of 664V. This allowed us to estimate how well our model fit HSPICE simulation.

7.1.5 Convex Optimization Problem Modeling

A major hurdle to using the above described methodology for optimum gate sizing was modeling the problem and reducing it to a convex problem. We derived ideas from Dinesh Patil's PhD thesis[1] and Prof. Stephen Boyd's paper on "Heuristics for Robust Gate Sizing"[2] and combined them to design the optimization problem.

In order to accurately model the CMOS transistors and delay, we used the velocity saturated model of the PMOS and NMOS. Assuming that the transistor spends more time in saturation than the linear region, we used the below fact to calculate delay through a gate:

$$\begin{aligned}C_{total_k} &= C_{junc} + C_{load} \\ids_gate_k &= \\&((size_gate_k) * \lambda * v_{sat} * C_{ox} * (V_{dd} - V_{th})^2) / ((V_{dd} - V_{th}) + E_{cL}) \\Delay_gate_k &= C_{total_k} * V_{dd} / (ids_gate_k)\end{aligned}$$

Power can be modeled as using the total switching capacitance when one address line is asserted (or deasserted). We get that:

$$Power_gate_k = 0.5 * C_{total_k} * V_{dd}^2 * f$$

By virtue of knowing the capacitances of each stage, we can find the sizings of each gate, using the velocity saturated model. Our total delay and power in now:

$$\begin{aligned}Decoder\ Delay &= \sum_{k=1}^8 Delay_gate_k \\Decoder\ Power &= \sum_{k=1}^8 0.5 * (C_{total_k}) * (V_{dd}^2 * f)\end{aligned}$$

We want to keep our clock frequency at 1 GHz. This means that at a particular voltage, we have the delay for the Memory Cell and Sense Amp, thus giving us the max delay we can get for the Decoder. For this delay, we want to minimize the power consumed. Thus, our optimization problem is

Minimize

$$Decoder\ Power$$

Where

$$Decoder\ Delay \leq (1ns - T_{cell_read} - T_{senseAmp})$$

Where our optimization variables are the sizes of each gate ($size_gate_k$).

Now, we also wanted to incorporate variations in the transistor and its affect on delay and power, so that we can size the decoder for robustness. For this, we used Pelgrom's model of sizing. We assumed that the variation in delay is normally distributed, with the mean at

the nominal delay and with some standard deviation. We obtained the standard deviation as follows:

$$\text{std_dev_k} = \text{gamma} * 1 / \sqrt{x} * \text{Delay_gate_k}$$

Where $\text{gamma} = 0.2$ i.e. one standard deviation is 20% of the delay of the gate and x is the size of the NMOS of the gate.

To obtain a robust modeling, we allowed for a variation of 3 standard deviations in the delay, giving us a 99.7 percentil delay value. For power, we used the argument made by Prof. Boyd in his paper that the independence in variations of power for each gate cancel each other in the limit, leading to the limit of the sum of power consumption being the nominal power. Thus, we did not account for variations in power, since those would be insignificant.

Taking all these into account, we obtain the following optimization problem:

Minimize

Decoder_Power

Where

$\text{Sum}(\text{Delay_gate_k} + 3 * \text{std_dev_k}) \leq (1\text{ns} - T_{\text{cell_read}} - T_{\text{senseAmp}})$

This formulation lead to larger sizings, but more robust predictions of delay. The power calculated is still much lower than what we found in Part2 of the project.

7.2 Redundent column optimization

7.2.1 Descibe what you hoped the change would accomplish

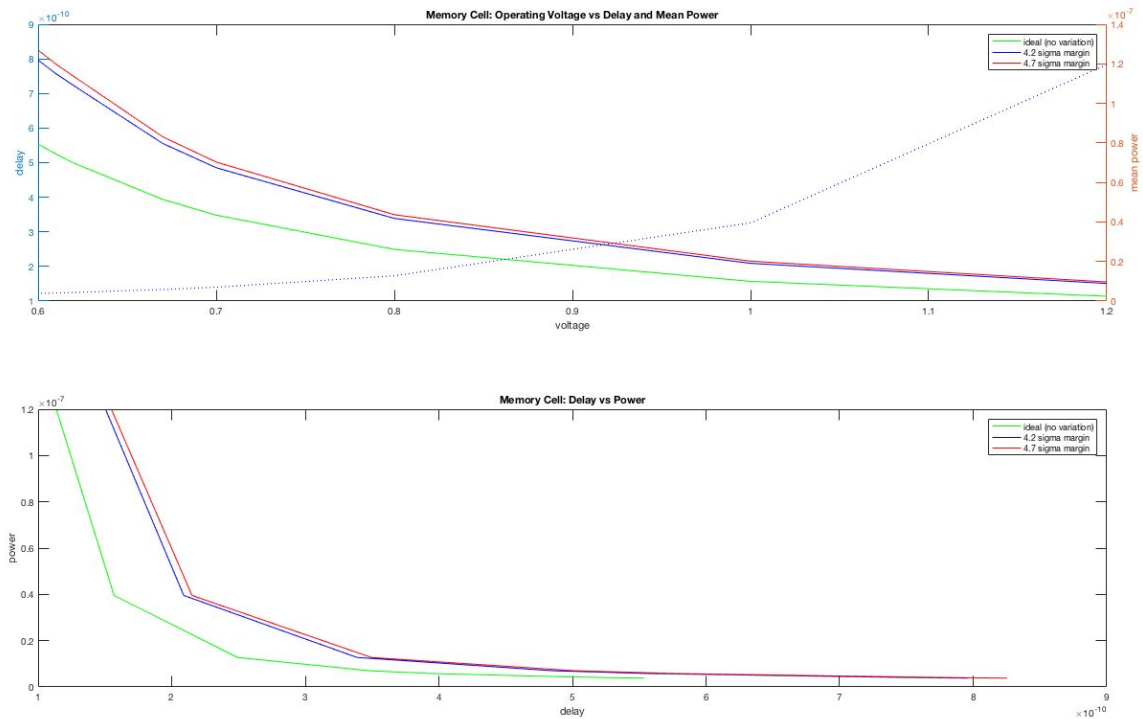
We hoped that by adding additional columns we would be able to lower our V_{min} (and thus our power consumption) while not hurting our delay too much.

7.2.2 Describe what the change actually accomplished

It turns out that additional columns didn't allow us to improve V_{min} as much as we hoped. This optimization did allow us to reduce our V_{min} , but only by a very small amount (see section 7.2.4)

7.2.3 Describe how you implemented this change

First, we estimated the number of columns we would need to reduce our V_{min} by a reasonable amount. We found diminishing returns after about 4 columns. We also estimated that the amount of additional power used by the extra column would outweigh the extra margin we got from adding more columns, so we decided to use 4 columns for the rest of our design. We then used simulation to find how power and delay changed as voltage changed, taking into account variation by performing monte carlo with 400 samples.



From this plot, we were able to examine what the best V_{min} for the overall system to run. We chose 664mV because the read delay shoots up drastically after $\sim 700\text{mV}$, and the extra energy needed to make the decoder operate fast enough made lowering V_{min} further not worth it.

Note that this particular decision was a result of wanting the memory to operate at 1GHz. If we had wanted a faster or slower decoder (or wanted the best combination of delay and energy) we would have chosen a different voltage (higher for better delay, lower for better overall power).

Finally, we added additional muxes to the write circuitry and sense amplifiers. This added very little delay over the inverter that was already present in both circuits ($\sim 40\text{ps}$ for the mux vs $\sim 20\text{ps}$ for the inverter, with a fanout of two).

7.2.4 Performance and power of your solution

If the delay of a given cell is normally distributed, then the amount of margining you need to meet our delay specification is ~ 4.2 sigma with 4 additional columns, and ~ 4.7 sigma with no additional columns (math to back this up is given in 7.2.5). From simulation, we found that at our chosen Vdd of 664mV, the standard deviation of the cell delay is about 40ps. This is equivalent to reducing Vdd of the memory cells by about 10mV. Thus, the reduction in Vdd we get from adding extra columns is very small since the standard deviation of the cell's delays is not large.

7.2.5 Yield estimations

Suppose we have a SRAM array with 64 bit words, 256 columns and 256 rows. Additionally suppose the read margin for a single cell is normally distributed with mean μ_v and standard deviation σ_v . What must the read margin be for the SRAM to have a parametric yield of 99.7%?

Assuming each cell is independent, we need the individual failure rate to be

$$\begin{aligned} X_c &= 1 - Y^{(1/N)} \\ &= 1 - 0.997^{(1 / 65536)} \\ &= 4.58452 * 10^{-8} \end{aligned}$$

This corresponds to ~ 5.5 sigma. Thus the SRAM voltage needs to be

$$V_c = \mu_v + 5.5 * \sigma_v$$

Next, suppose we add N redundant columns (for a total of $256 + N$). What must the read margin be for the SRAM to have a parametric yield of 99.7%?

From equation 7.30 in the book we can calculate the minimum yield required for a column (this equation is solved numerically)

$$0.997 = \sum_{i=0..N} \text{choose}(256 + N, i) * Y_c^{(256 + N - i)} * (1 - Y_c)^i$$

Next, note that for a column to function all cells must work correctly (assuming no extra rows). For a given Y_c we get a required cell failure rate of:

$$X_c = 1 - Y_c^{(1/256)}$$

For $N = 1..4$, that gives the following Y_c , X_c and V_c :

N	Y_c	X_c	V_c
1	0.999690	$1.21112 * 10^{-6}$	$\mu_v + 4.7 * \sigma_v$

2	0.998907	4.27186×10^{-6}	$\mu_v + 4.5 \times \sigma$
3	0.997746	8.81459×10^{-6}	$\mu_v + 4.3 \times \sigma$
4	0.996310	1.44406×10^{-5}	$\mu_v + 4.2 \times \sigma$

7.3 Clock Generation

7.3.1 Describe how you generated the clock/control signals (BLPC, SAPC, SAE)

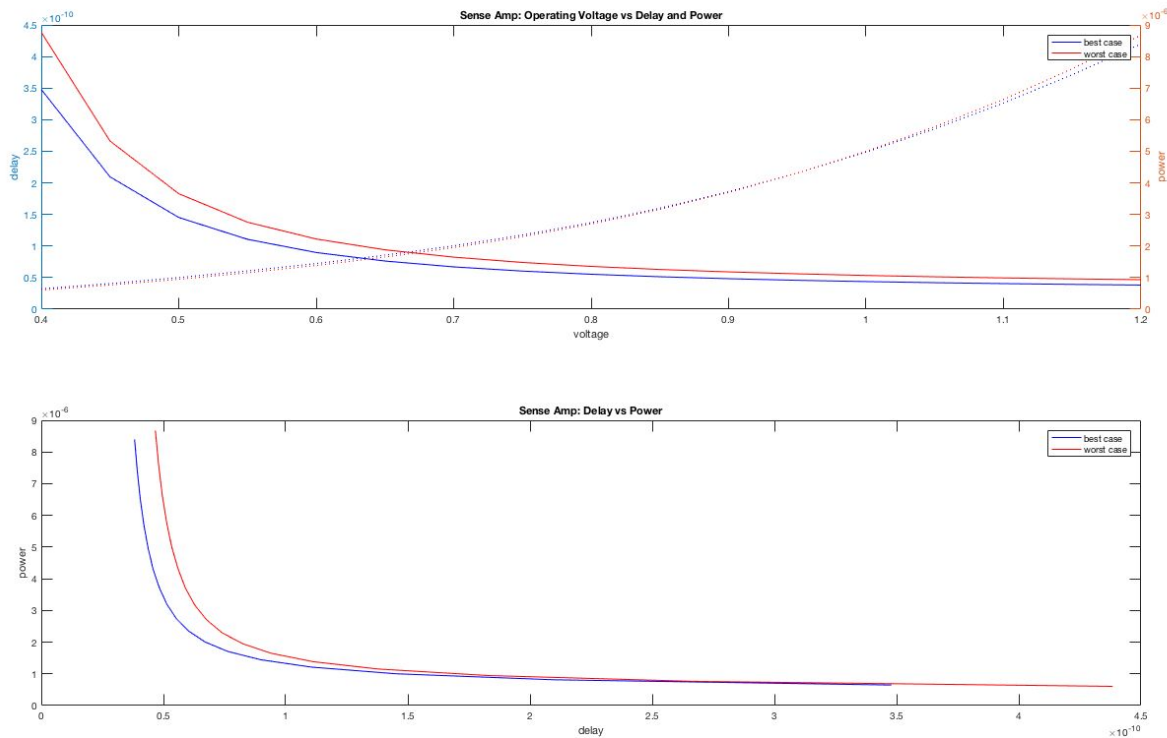
For BLPC and SAPC we used a chain of four inverters sized for optimal delay according to the method of logical effort.

For SAE, we used a redundant bitline with three inverters at the end of the bitline.

We did not limit bitline swing using SAE.

7.3.2 For each signal explain whether it is a delayed clock, the margin you gave it, and how you generated it.

To find the amount of margin needed we simulated the delay of the nominal and worst case sense amplifier.



From this, we found that a worst case sense amplifier at our chosen V_{min} of 664mV needed roughly 214ps to evaluate the bitlines.

We then designed SAE with a redundant bitline, precharged like the other bitlines, and pulled low by a transistor attached to each wordline. At the base of this bitline we had three inverters that were sized to let the entire system have exactly 788ps of delay. This gave enough margin that even a three sigma worst case memory attached to a sense amplifier with 50mV offset would be able to evaluate the bitlines within our 1ns constraint. The variation of the inverters driving the sense amplifier was simulated and found to be within this margin (<4ps at 3 sigma).

If we wanted to additionally margin SAE, we could have added a mux to choose between two nominally identical paths (with the path with the closest “correct” delay at test being chosen).

8. Analysis – The memory you wished you designed

The first thing we would have changed is to either include both redundant columns and rows, or to do a different optimization entirely. Doing just redundant columns improved our power numbers, but not by a significant amount to matter.

The second thing we would have changed is to use a better design for our clock generation circuits. These circuits were optimized for delay rather than power, and it dramatically increased the overall amount of power consumed by the design. With more appropriate sizing, we could have reduced our power consumption further.

Finally, we would have taken into better account the slew of the output when optimizing the decoder. Our final slew wasn’t terrible, but some designs we looked at had horrible slews due to bad constraints during optimization.

One thing that worked well was using convex optimization methods for sizing. The predicted delays and powers were a good match for the actual delays and allowed us change our timings between subsystems without having to redesign the decoder from scratch. It also produced a much lower power decoder than what we would have been able to design by hand.

9. References

- [1] “Design Of Robust Energy-Efficient Digital Circuits Using Geometric Programming”, Dinesh Patil
- [2] “A Heuristic Method for Statistical Digital Circuit Sizing”, Stephen Boyd, Seung-Jean Kim, Dinesh Patil, Mark Horowitz