EECS 151/251A Homework 7

Due Monday, April 5th, 2021

For this Assignment

Please include a short (1-2 sentence) explanation of your answer with each response unless otherwise directed to by the problem.

Problem 1: Technology Survey (No short explanation needed)

Take a look at your phone. What is its battery capacity in kW h and in J. How often do you charge your phone, and from what battery level do you start charging from each day? Estimate your average power consumption from those values. Repeat this exercise for your laptop.

Solution:

I use an iPhone X, which has a 10.35 W h (37.26 kJ) battery. I tend to leave my phone on the charger, but when I don't have access to one, I charge up from 20% by the end of the day. I therefore use roughly 8 Wh (28.8 kJ) per day.

My laptop is a MacBook Pro 15" 2017 Edition, which has a 76 Wh (273.6 kJ) battery. I keep my laptop plugged in constantly, but according to a power management tool I have installed on it, I consume about 12 W while idling, and I spend about 10 hours a day on it, so that would equate to 432 kJ per day.

Problem 2: Parallelization & Pipelining

One of your co-workers has designed an accumulator that takes a running sum of 8 integers. The integers arrive as an 8-wide integer array and the accumulator cycles through each element of the array and returns the final sum. They aggressively designed their accumulator to meet a throughput target of 64×10^7 sums/sec by pushing their accumulator to be as fast as possible. However, in doing so, their design has a dynamic power consumption of 75 W, which is unacceptably high for a simple accumulator. At your next weekly meeting, they sketch their design on the board and bring up their dilemma and you immediately recall learning in EECS151 about two ways to achieve the same throughput while using less power.

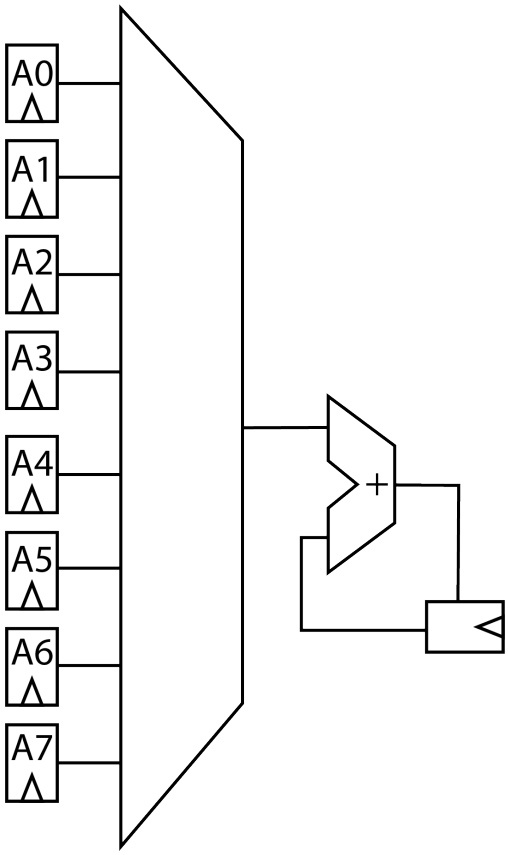
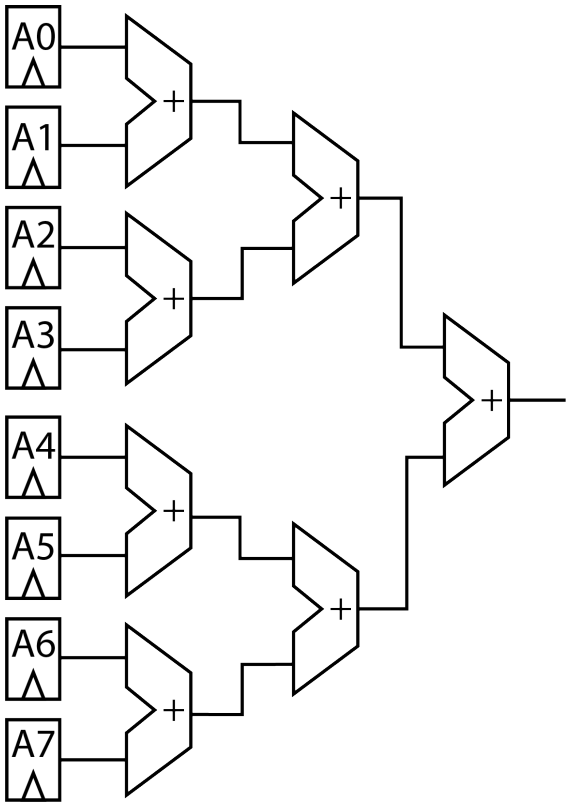


Figure 1: Sketch of original design

(a) The first idea that comes to your mind is just to parallelize the design. How would you modify the design to parallelize the hardware as much as possible while maintaining the same throughput? How and why does this improve the dynamic power consumption? (*Hint: Think of a binary tree*)

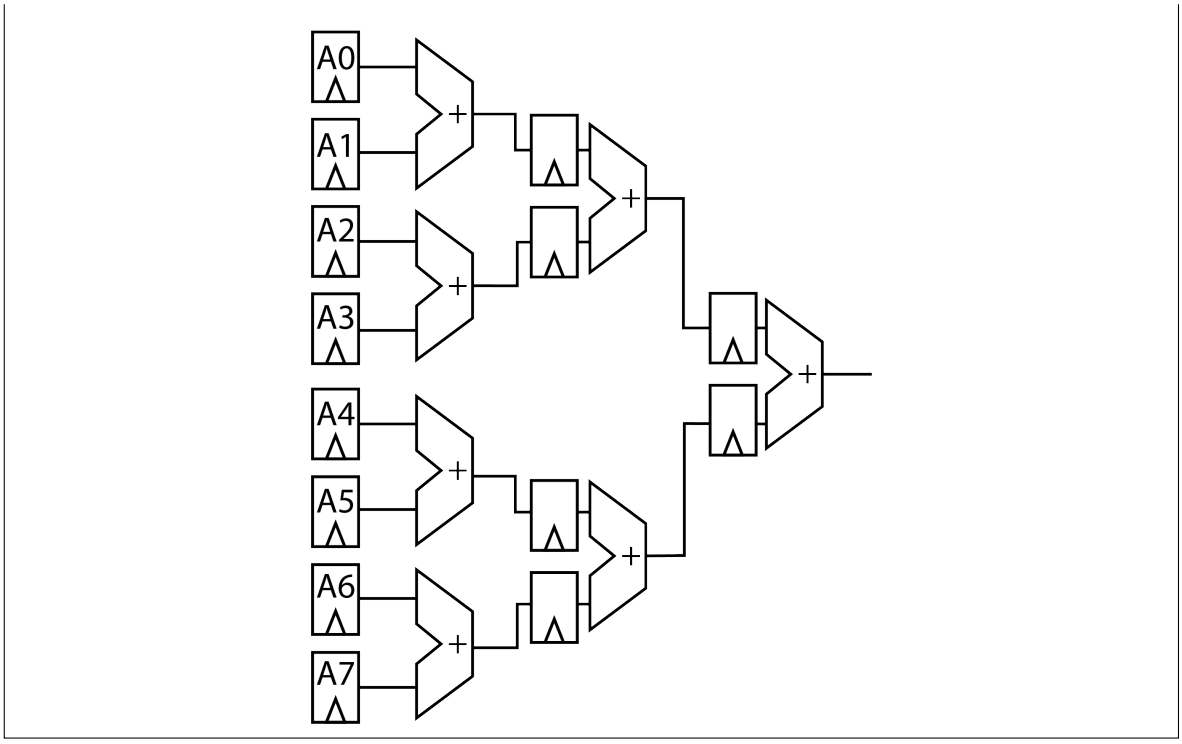
Solution:



We can make a binary adder tree to perform the addition. Since the critical path is now 3 adders long and the sum is performed within one clock cycle, we can relax the clock speed significantly while still maintaining the same throughput.

(b) Now as you consider your parallelized approach, it dawns on you that you can also leverage the other technique you learned and pipeline the resulting design. How would you pipeline the parallelized design that you came up with in the last part? How and why does this further improve the dynamic power consumption?

Solution:



Problem 3: Race to Halt

One scheme for potentially improving energy efficiency if static power is a significant proportion of the total power consumption is a technique known as "race to halt". Basically, we run the circuits at maximum speed to finish the computation as quickly as possible, then cut off the power so that we don't suffer the static power loss.

Suppose we have a CPU that takes 10 seconds to run a particular application, consuming 12 W, where some propotion δ of the total power is consumed by dynamic power, with the remaining $\sigma = 1-\delta$ the proportion lost to static power consumption. Assuming there are no other applications running on the CPU and that these two proportions cover the entire power budget of the CPU, we would like to find a scheme that would minimize power consumption.

As an alternative to the race to halt method, we could also consider a more traditional frequency/voltage scaling method for reducing power consumption. The technology we are working with can tolerate a V_{DD} reduction by at most 25%, and we can assume that the voltage to delay scaling is linear (e.g. a 2x reduction in supply voltage will need a 2x decrease in clock frequency).

Explore race to halt versus frequency/voltage scaling. Assume that the voltage and frequency scaling will only affect dynamic power consumption, and does not affect the amount of static power consumption (which is not an unrealistic approximation). For what δ would race to halt be better than frequency/voltage scaling?

Solution:

We can express the energy consumed by the race to halt processor as follows:

$$(12)(10)(\delta)$$

The energy consumed by the scaling processor is expressed as

$$(12)(10/0.75)(1-\delta)+(12)(\delta)(0.75)^3$$

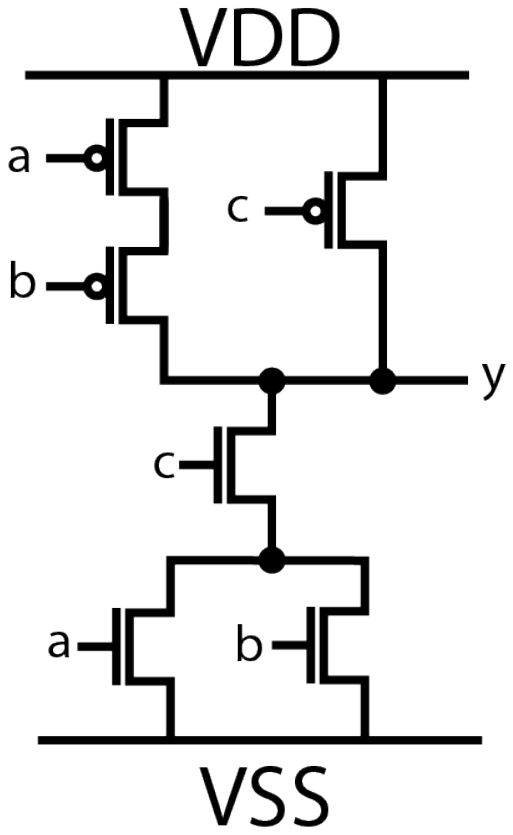
To find the crossover point, we set the two equal and solve for δ,

$$(12)(10)(\delta)=(12)(10/0.75)(1-\delta)+(12)(\delta)(0.75)^3$$

This converges at $\delta \approx 0.58$.

Problem 4: Logic Gate Dynamic Power

Consider the OR-AND-INV (OAI) gate shown below, operating at $V_{DD} = 1\,\text{V}$, with a load of $C_L = 20\,\text{fF}$, and $C_D = 0.5\,\text{fF/nm}$.



(a) Size the devices in the OAI gate such that the input capacitance is equal to a unit-sized inverter, with NMOS width $W_N = 12\,\text{nm}$. Make sure the rise and fall time of the gate is balanced for the worst case.

Solution:

In the worst case, the output must be driven through both a and b PMOS devices, so these will each be twice the size of the NMOS devices. The a and b PMOS devices are therefore 24 nm wide, and the NMOS devices are each 12 nm wide. The c PMOS device can be sized to be 12 nm.

(b) What is the dynamic power dissipation of the gate, assuming the design runs at a clock frequency of 4 GHz and the gate output has an activity factor of $\alpha = 0.4$?

Solution:

The total capacitance at the output is

$$(0.5\,\text{fF/nm})(12\,\text{nm} + 24\,\text{nm} + 12\,\text{nm}) + (20\,\text{fF}) = 44\,\text{fF}$$

The dynamic power formula is

$$P_{dyn} = \frac{1}{2} C V_{DD}^2 f \alpha$$

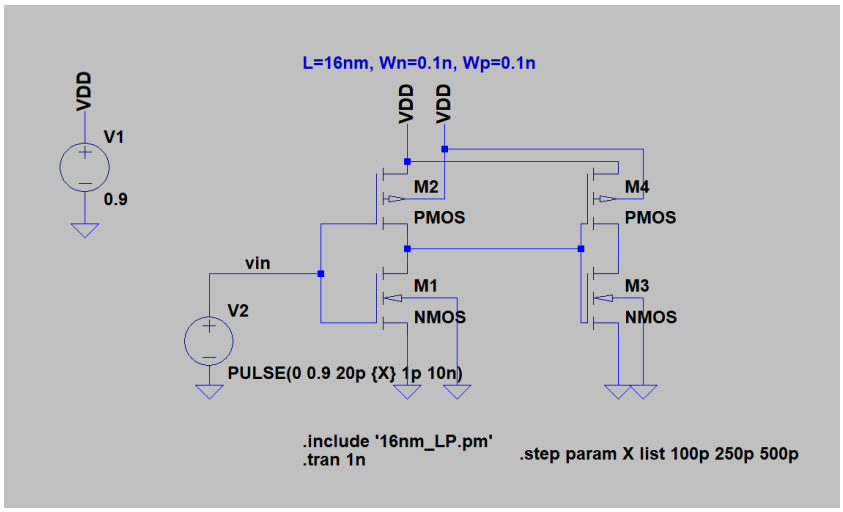
plugging these in,

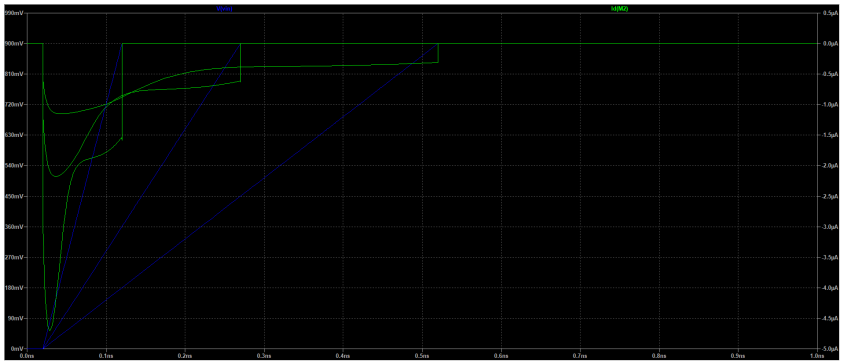
$$P_{dyn} = \frac{1}{2}(44\,\text{fF})(1\,\text{V})^2(4\,\text{GHz})(0.4) = 35.2\,\mu\text{W}$$

Problem 5: Short Circuit Power

Set up a simulation in LTSpice to demonstrate short-circuit power. Build an inverter with $W_P = 2W_N = 0.5\,\text{nm}$ and $l_N = l_P = 16\,\text{nm}$ at a $V_{DD} = 0.9\,\text{V}$. Set up the schematic to have input transitions of 10 ps, 25 ps, and 50 ps. To measure the short-circuit current, consider this: In an ideal inverter with no short-circuit current, for a high-to-low transition the only device that should be conducting is the pull-down device. Any additional current through the pull-up device is unwanted current, and thus contributes to the short-circuit power that we would like to avoid. Therefore to measure the short circuit current, we just need to see how much current is conducting through the undesired device. Please turn in a screenshot of your schematic as well as a waveform of the three short-circuit currents.

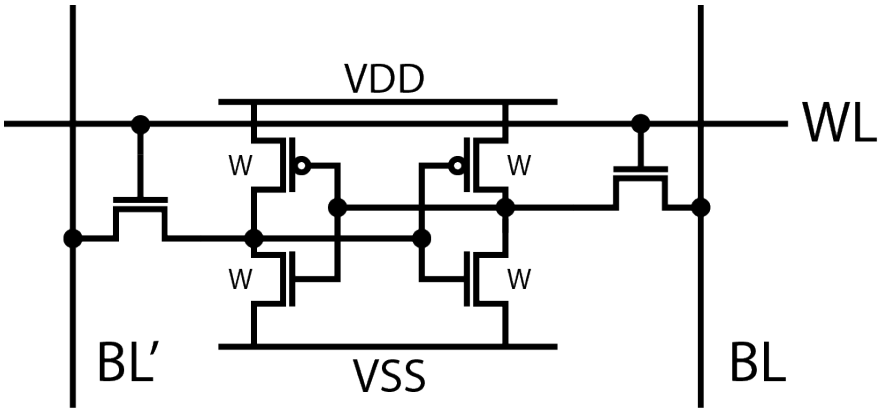
Solution:





Problem 6: 6T SRAM Cell

Consider the 6T SRAM Cell below, with the inverter device sizes labeled. In this technology, $R_P = R_N$. The cell operates at a supply voltage of 0.9 V



(a) How would you size the access transistors to ensure a successful write? Assume for this technology that the voltage needs to be more than 20% V_{DD} away from the switching voltage

($V_{DD}/2$) for a successful write. You may assume the column drivers perfectly drive the bit lines, and the only significant resistances are from the SRAM cell and the access transistors.

Solution:

The write must make it to 70% VDD to write a 1, or 30% VDD to write a 0. Since the drivers are able to drive the bit lines as ideal voltage sources, this is simply a resistor divider network. The access transistor must therefore be $\frac{7}{3}W$.

(b) A write-only SRAM cell is not very useful, so we will also need to read from this SRAM cell. However, you just sized the devices so that the SRAM cell is easy to write to, which means the typical read procedure has a high risk of corrupting the memory values stored in the SRAM cell. How would you resize the devices to reduce the chance of read corruption while still maintaining a successful write? You may now resize the devices in the SRAM cell itself if you wish to do so. Again, you may assume the only significant resistances are that of the SRAM cell and the access transistors. You can also assume that the bitlines have extremely high (effectively infinite) capacitance so the voltage on them will not change significantly during the read operation, but the sense amplifier will still be sensitive enough to register a read.

Solution:

The key realization for this problem is that during a read, the only SRAM device that does any work is the NMOS. Since both bit lines are precharged to VDD, the PMOS on the side holding a 1 does not actually contribute anything to the read operation, while the NMOS on the side holding a 0 must pull the bit line down enough for the sense amp to read. This means we can just make the NMOS significantly larger than the access transistor so during a read, it will maintain the value in the cell such that there is no risk of a read corruption. In this case, if we assume the access transistor is still $\frac{7}{3}W$, the NMOS must be the smaller resistance in the resistor divider, with a size of $\frac{49}{9}W$.

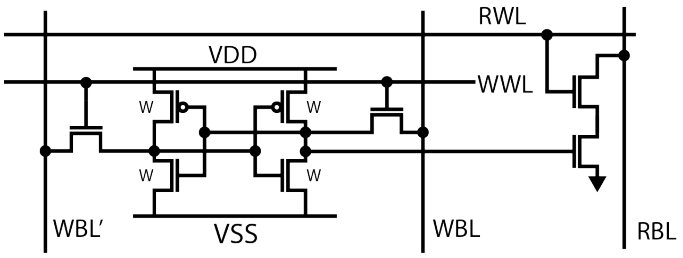
This won't cause a problem with the write however, as the PMOS devices are still small, and therefore during the write operation, only the side writing a 0 needs to overcome the PMOS device. This will pull the input of the side holding a 0 low enough to weaken the NMOS and allow for a successful write.

(c) **251A only — *Optional* Challenge Question for 151**

If you could redesign the SRAM cell, is there a way you could break the read vs. write sizing tradeoff? You may add or remove devices from the classic 6T structure to achieve this new design.

Solution:

One potential solution is what is known as an 8T SRAM cell. We can decouple the read and write operations by reading using the gate of an NMOS transistor rather than having the SRAM cell drive the bit line directly. This allows for the access transistors to only be sized for write, and it is now almost impossible to cause a read corruption. This comes at the cost of additional hardware though, as we now need a new dedicated word/bit line for the read, as well as 2 extra transistors.



Problem 7: Building Bigger Blocks

As modern processors are able to handle more and more computations per second, the associated memories must also be able to hold more and more data to keep up. However, one major problem with making huge SRAM blocks is that the bit lines will become extremely long as the memory increases in capacity, which causes issues with readability and write speed. One way to get around this problem is to make the overall memory block out of smaller sub-blocks.

For this problem, you have access to an SRAM block that is 32 words deep with a 32-bit word length. The basic SRAM block has a single read port and a single write port.

(a) Describe how you would make a 128KB, 32-bit wide dual-port memory (single read, single write) using these sub-cells. You will need to describe your periphery circuits. How would you assign the address bits for this design (row/column arrangement)?

Solution:

Since these are self-contained SRAM blocks, they can be arranged more flexibly. However, we will still want to stick to a 1:1 aspect ratio as much as possible. With 128KB of total memory, we can build the array with a 32x32 grid of our SRAM blocks, which would necessitate 5 bits of row and column address bits each, for a total of a 10-bit address.

(b) Describe how you would implement a dual-read SRAM (1 write port, 2 read ports).

Solution:

To implement an additional read, we will replicate the access transistors, the word line, and the bit linnes. This will allow a double read. We will need to careful that reading the same cell at the same time will not cause a read corruption, however, so the access transistors may need to be resized appropriately.

Problem 8: Address Decoding

Consider a 16-bit wide 2KB SRAM. How many rows and columnns are in your design? Design an address decoder using the predecoder technique from the lecture. You may use only logic gates of no more than 4 inputs.

Solution:

Adhering to a 1:1 aspect ratio, the SRAM will have 128 rows and 128 columns, divided into 8 words fo 16-bits each. The row decoder must therefore decode a 7-bit row address, while the column address can be just 3 bits (only need to select 8 words).

Building the predecoder will follow the same procedure as in slide 12 in lecture 18, but because we now have an additional address bit, we can implement this as either predecoding in groups of 3, 2, 2, or implementing the final decoder with 4-input gates to accomodate the extra bit.