EECS 151/251A Homework 10

December 6th, 2019

Problem 1: Hidden Caches... [6 pts]

You're given an L1 direct mapped cache design and another L2 2-way cache design with the following performance metrics:

Cache	Hit Rate	Miss Penalty	Hit Time
L1	0.9	5 ns	1ns
L2	0.95	10 ns	3 ns

Calculate and compare the average memory access times (AMAT) for an L1 cache architecture and an L1 + L2 architecture.

1. L1 only AMAT:
2. L1 + L2 AMAT:

Looking at the two architectures below, can you explain the difference in hit times? What are the critical paths in each cache?

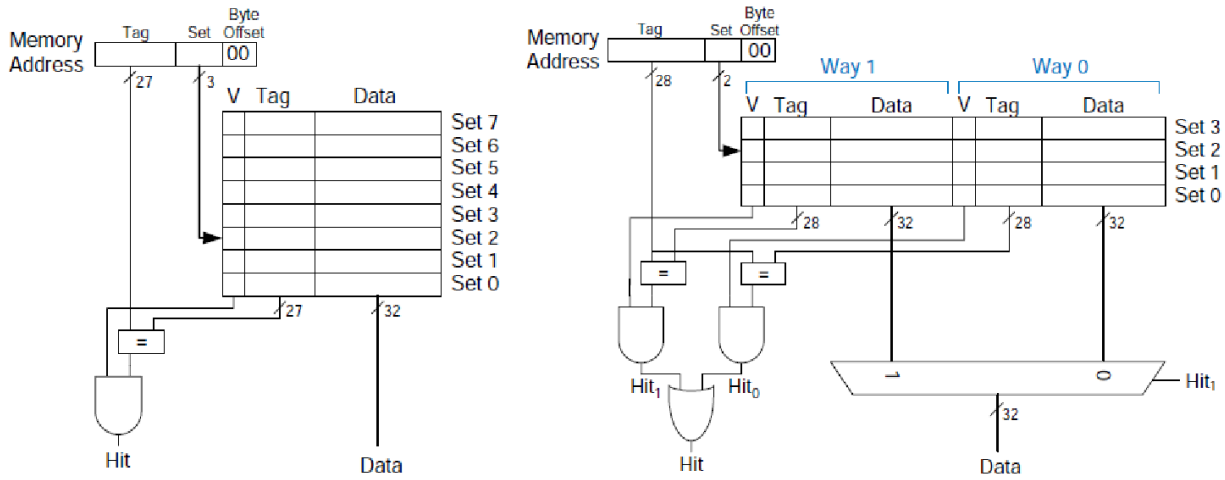


Figure 1: Direct mapped and 2 way cache architectures

Problem 2: Yea, I've got time... [9 pts]

Your ASIC needs to be supplied by an off-chip clock (meaning it'll come from a pad).

1. Draw 3 levels of an H tree + extra wires on the figure below such that the two flip flops (endpoints) see minimal skew.
2. Given that it takes 1.25mm of wiring to connect one flip flop, and 1.26mm of wiring for the other. What is the wire delay from the pad to both flip flops? What's the skew between the two flip flops? Assume a wire resistance of $0.1\Omega/mm$ and a $C_{pp} = 20aF/\mu m^2$.
3. How many inverters are needed to minimize the delay along both paths? What's the skew now? (You can ignore branching)
4. What would the delay be if you placed an inverter at each branching node of the H-Tree? (Account for branching now)

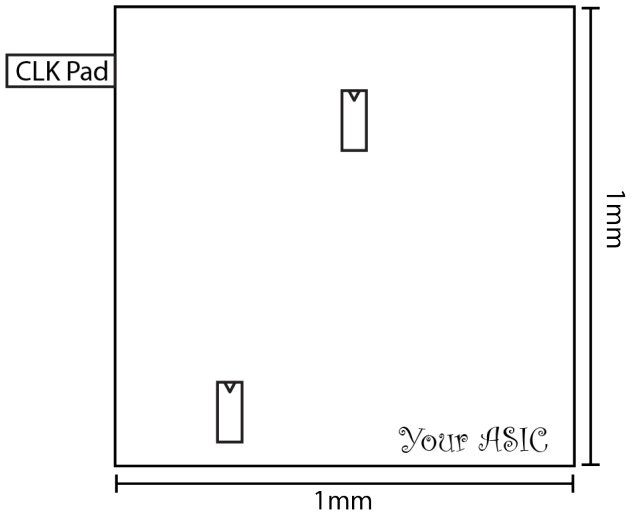
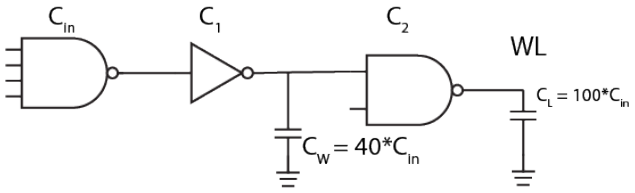


Figure 2: A floormap of YOUR ASIC!

Problem 3: Decoders [4 (6) pts]

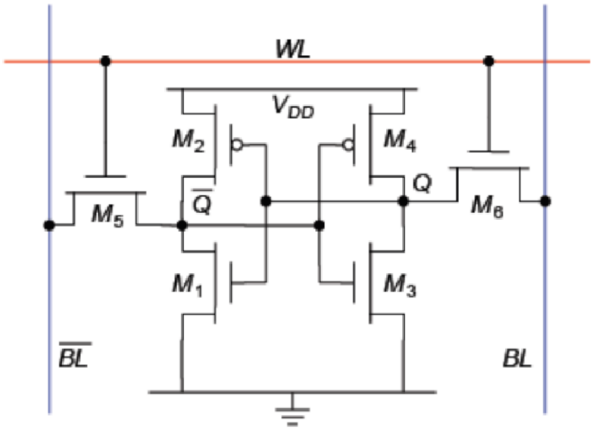
Lets size a memory decoder that consists of a 4-bit pre-decoder and a 2-bit final decoder. $C_in = 1fF$, and C_W is a model for the large wire capacitance between the pre-decoder and the final decoder.



1. Assume that $C_W = 0$. Optimally size the decoder by finding the input capacitances of gates C1 and C2 to drive the wordline load $C_L = 100C_{in}$.
2. (251A students only) Derive the equation for the delay of this chain in terms of the input capacitances of the three gates (C_{in}, C_1, C_2, the capacitances, C_W and C_L and t_{inv}.
3. (251A students only) Using the values for C_W and C_L given in the figure and your equation from part (b), determine the optimal sizing for the gates to minimize the total delay.
4. (251A students only) We will now explore an alternative chain sizing heuristic. First, note the optimal size for the last NAND2 gate without the wireload (as in part A). By leaving the sizing of this last gate constant and re-introducing the wire-load, you can now calculate the total fanout that the first two gates must drive. Based on this total fanout for the first two gates, you can now size the inverter using the standard method we learned in class. Show your work and include a gate level schematic of the new chain.
5. (251A students only) Compare the results from parts c) and d). Explain the differences.

Problem 4: I think I remember... (SRAM) [4 pts]

Cache lines are generally large (assume a line is 64 bytes or 512 bits), but accesses to these lines are small (only 8, 16, or 32 bits at a time). The cache line could be build out of narrow width SRAM macros (eg. 8 bits wide), but these are not area efficient because peripheral circuitry is amortized over less bitcells. Other techniques are therefore typically needed.



Given the 6T cell above, assume that a precharge operation between accesses precharges the bitlines high.

1. Given the data input to the cell din[x], an active high write enable signal WE, first write out the logic and then draw an efficient schematic of the circuitry required to write the cell. Hint: You'll want to add things to your bitlines. You should be able to use no more than 4 NMOS transistors and an inverter.
2. Now we'll add our mask. Given an additional active low signal mask, which prevents writes when mask=0, draw a transistor-level diagram of circuitry that only writes the cell when both mask and WE are high.

Problem 5: I think I remember something else... (DRAM) [2 pts]

As you know, DRAM, while significantly more dense than SRAM, requires refreshing. For a given DDR3 array implemented in some technology, assume a retention time of 64 ms at 85°C.

1. How often do you have to refresh your array?
2. Normally you'd break up your refresh operation into a row-by-row basis. If you have 8,192 rows, how often will you be refreshing a single row? Qualitatively, what does this refresh overhead mean in terms of memory availability?