UNIVERSITY OF CALIFORNIA
College of Engineering
Department of Electrical Engineering and Computer Sciences

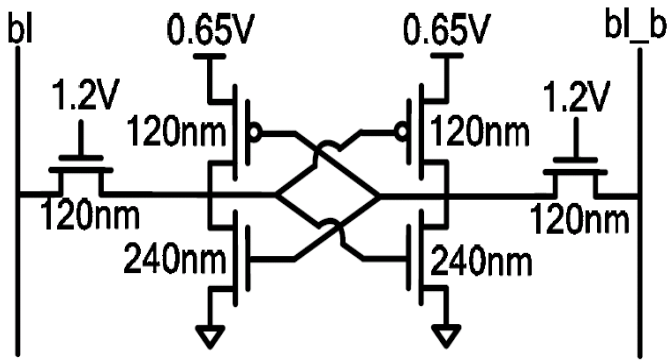
Jan M. Rabaey | *Homework #9* | *EECS 141*
(Optional, Not Graded)

Problem 1 – SRAM Design (from F'08 Final Exam)

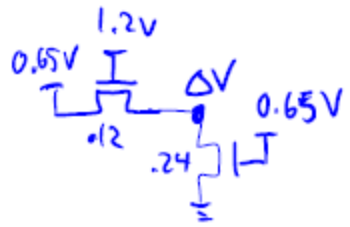
For this problem, you should use the velocity saturated transistor model. You can assume that all transistors have a channel length of 100nm and the following parameters (unless otherwise mentioned):

NMOS:
V_{Tn} = 0.2V, μ_n = 400 cm^2/(V·s), C_{ox} = 1.125 μF/cm^2, v_{sat} = 1e7 cm/s, L=100nm, γ=λ=0
PMOS:
$|V_{Tp}|$ = 0.2V, μ_p = 200 cm^2/(V·s), C_{ox} = 1.125 μF/cm^2, v_{sat} = 1e7 cm/s, L=100nm, γ =λ=0

a) Shown below is an SRAM cell during a read, where the power supply of the SRAM has been reduced to 0.65V while the VDD of the wordline is 1.2V. Note that the bitlines have also been precharged to 0.65V. With the device sizing shown below, what is the read ΔV? (Hint: How much larger is I_{DSAT} for a transistor with V_{GS} = 1.2V than with V_{GS} = 0.65V?)



Read circuit:



$E_c L = \frac{2v_{sat}}{\mu} L = 0.5V \quad V_{TN} = 0.2V$

$$\frac{I_{DSAT}\,(V_{GS}=1.2V)}{I_{DSAT}\,(V_{GS}=0.65V)} = \frac{(1.2V-0.2V)^2/(1.2V-0.2V+0.5V)}{(0.65V-0.2V)^2/(0.65V-0.2V+0.5V)} \approx 3.13$$

So, even though access device is half as wide, its current would be higher than the pull-down device if $\Delta V \approx 0$. So, lets guess that both are in sat:

$$\frac{1}{2}\frac{(1.2V-\Delta V-0.2V)^2}{(1.2V-\Delta V-0.2V)+0.5V} = \frac{(0.65V-0.2V)^2}{(0.65-0.2V)+0.5V} \rightarrow \boxed{\Delta V \approx 278mV}$$

Check our guess:

$V_{GTaccess} = 1.2V - 278mV - 200mV = 722mV$

$V_{DSaccess} = 0.65V - 278mV = 372mV$

$V_{GTpd} = 0.65V - 0.2V = 450mV$

$V_{DSpd} = 278mV$

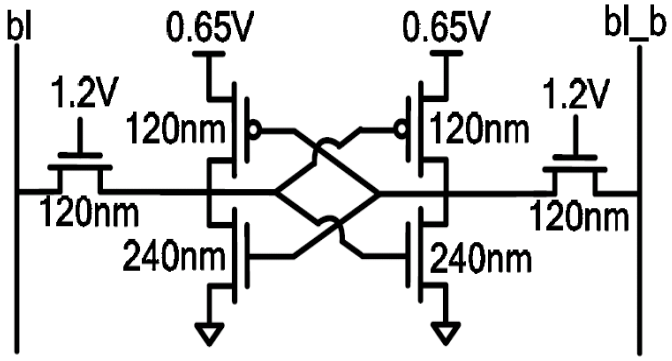
$V_{DSAT_a} = \frac{722mV \cdot 500mV}{722mV + 500mV} \approx 295mV$

access saturated ✓

$V_{DSAT_{pd}} = \frac{0.45V \cdot 0.5V}{0.45V + 0.5V} \approx 237mV$

pull down saturated ✓

b) Assuming that in your answer to part a) you calculated that the pull-down device is saturated, how much faster does the SRAM cell pull down the bitline when the wordline is driven to 1.2V compared to if the wordline was driven to only 0.65V? (Note that most of the credit on this problem will be given for finding the right regions of operation and setting up the equations.)



When $WL = 1.2V$:

$I_{pulldown} = I_{DSAT}$ of the pull-down device

$I_{pulldown} \propto 0.24 \cdot \frac{(0.65V - 0.2V)^2}{(0.65V - 0.2V) + 0.5V}$

When $WL = 0.65V$:

pull-down much bigger, so it should be in linear region.

0.65V
0.12 — 0.65V
ΔV
0.24 — 0.65V

Solve for ΔV:

$$0.12\mu m \cdot 1e7\,cm/s \; \frac{(0.65V - 0.2V - \Delta V)^2}{0.65V - 0.2V - \Delta V + 0.5V} = \frac{0.24\mu m}{0.1\mu m} \cdot 400\,cm^2/V\,sec\,(0.65V - 0.2V - \Delta V/2)\Delta V$$

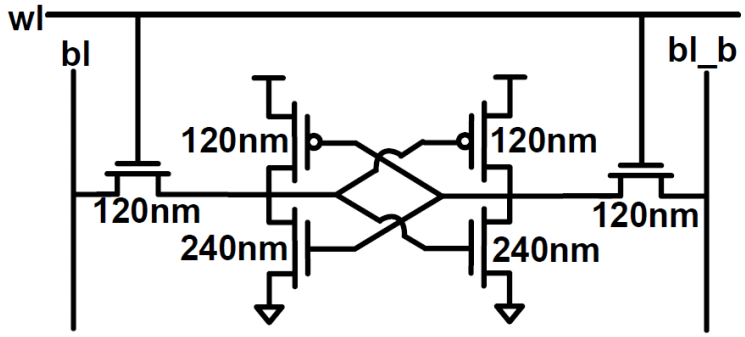
$\hookrightarrow \Delta V \approx 52mV$

So, $$\frac{I_{pd}(WL = 1.2V)}{I_{pd}(WL = 0.65V)} = \frac{0.24\,(0.65V - 0.2V)^2/(0.65V - 0.2V + 0.5V)}{0.12\,(0.65V - 0.2V - 0.052V)^2/(0.65V - 0.2V - 0.052V + 0.5V)}$$

$$\frac{I_{pd}(WL = 1.2V)}{I_{pd}(WL = 0.65V)} \approx 2.417$$

Problem 2 – SRAM Design (from F'09 Final Exam)

For this problem we will be looking at a 256x32 SRAM (i.e., each wordline drives 32 cells, and each bitline has 256 cells on it), with each cell sized and implemented as shown below. You can assume that $C_G = C_D = 2fF/\mu m$, $R_{sqn} = 10k\Omega/\square$, $R_{sqp} = 20k\Omega/\square$, that the transistors are quadratic (i.e., long-channel), and that you can ignore all wire parasitics.



a) **(6 pts)** Assuming you use a static decoder with only NAND2's and inverters, and that the input capacitance on each address input must be less than 2fF, what is the minimum delay of the decoder for this 256x32 memory? You should provide your answer in units of $t_{FO4} = (4+\gamma)t_{inv}$, and you can assume that the parasitic delay of all of the NAND gates is γt_{inv} (i.e., the same as an inverter), and you can ignore the fact that you can't build a fractional number of stages.

$C_{WL} = 32 \cdot 2fF/\mu m \cdot 0.24\mu m = 15.36\,fF$

$F = 15.36\,fF / 2fF = 7.68$

$TB = 256/2 = 128$

8 address lines, so need 3 stages of AND'ing:

$TLE = \frac{4}{3} \cdot \frac{4}{3} \cdot \frac{4}{3} \approx 2.37$

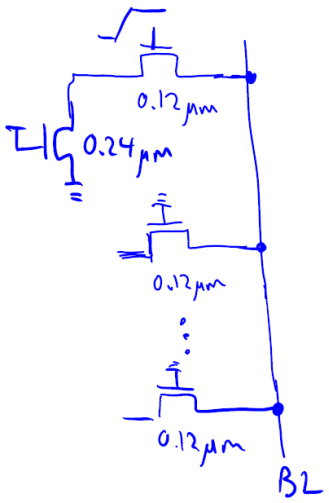
$PE \approx 2330.2$

$N_{stages,opt} = \log_4(PE) \approx 5.59$

Each stage has EF=4 and parasitic delay of γt_{inv}:

$t_{p,dec} \approx 5.59\, t_{FO4}$

b) **(4 pts)** For this same 256x32 SRAM, what is the delay (still in units of t_{FO4}) from the wordline rising to the bitline crossing Vdd/2?



$$R_{cell} = L R_{sqN} \cdot \left(\frac{1}{0.12\mu m} + \frac{1}{0.24\mu m} \right)$$

$$C_{BL} = 0.12\mu m \cdot C_D \cdot 256$$

$$t_p = R_{cell} C_{BL} = L R_{sqN} C_D \cdot 256 \cdot \left(1 + \frac{1}{2}\right)$$

$$t_p = \frac{3}{2} \cdot 256 \cdot L \cdot R_{sqN} \cdot C_D$$

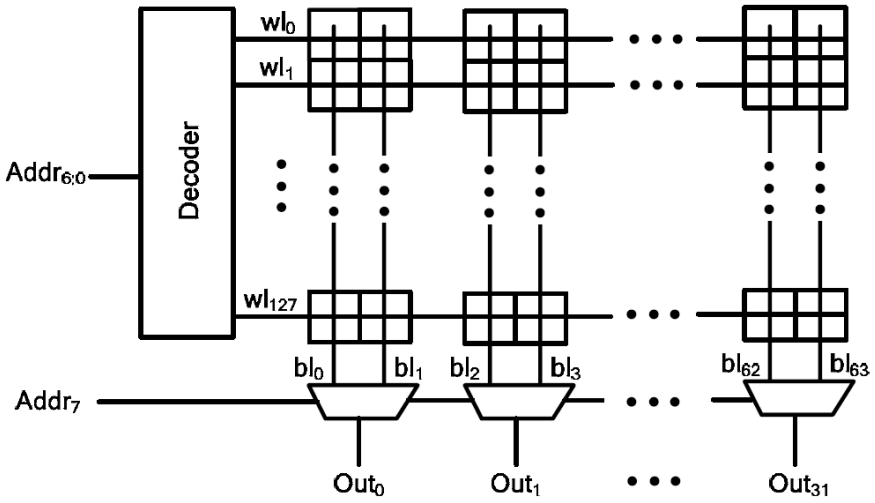
(including slope effect)

$$t_{inv} = 3 L R_{sqN} C_G$$

$$t_{FO4} = (4+\delta) t_{inv} = 15 L R_{sqN} C_G$$

$$\hookrightarrow \boxed{t_{p,BL} = 25.6\, t_{FO4}}$$

c) **(8 pts)** Now let's examine the effect of repartitioning the SRAM into a 128x64 array with 2-input MUXes selecting the appropriate final output, as shown below. Using your answers from parts a) and b), ignoring the capacitive loading of the MUX on the bitlines, and assuming that the delay of the MUX is 2 t_{FO4}, now what is the total delay of the SRAM from Addr to Out?



* For decoder, $F = 2 \cdot F_{256\times32}$, but $\Pi B = \frac{\Pi B_{256\times32}}{2}$.
  Since 7-input AND still needs 3 levels of NAND2's, $\Pi LE = \Pi LE_{256\times32}$.
  So: $PE_{dec,128\times64} = PE_{dec,256\times32}$
  ↳ $t_{p,dec,128\times64} = t_{p,dec,256\times32} = 5.59\, t_{FO4}$

* Half as many cells on the bitline, so:
  $t_{p,BL,128\times64} = \frac{1}{2} t_{p,BL,256\times32} = 12.8\, t_{FO4}$

* $t_{p,tot} = t_{p,dec} + t_{p,BL} + t_{p,mux}$
  $t_{p,tot} = 5.59\, t_{FO4} + 12.8\, t_{FO4} + 2 t_{FO4}$
  $\boxed{t_{p,tot} = 20.39\, t_{FO4}}$

d) (8 pts) Now assuming that the SRAM is partitioned into a $256/N_{part}$ x $32*N_{part}$ array, what value of N_{part} results in the minimum total delay for the SRAM? You should assume that the overall logical effort of the decoder is independent of N_{part}, and that the delay of an N-input MUX is N t_{FO4}.

* Need equation for total delay as a function of N_{part}.
* Under these assumptions, decoder delay is independent of N_{part} since each address always needs to eventually drive half of the array (i.e., $\Pi B \cdot F$ is constant). $t_{p,dec} = 5.59\, t_{FO4}$
* $t_{p,BL} = t_{p,BL,256\times32} / N_{part} = \frac{25.6\, t_{FO4}}{N_{part}}$
* $t_{p,mux} = N_{part}\, t_{FO4}$

So: $t_{p,tot} = t_{p,dec} + \frac{t_{p,BL,256\times32}}{N_{part}} + N_{part}\, t_{FO4}$

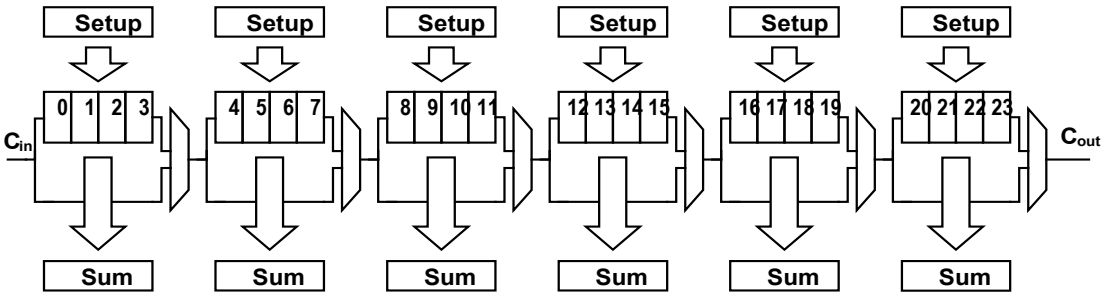
$\frac{\partial t_{p,tot}}{\partial N_{part}} = -\frac{t_{p,BL,256\times32}}{N_{part}^2} + t_{FO4} = 0$

↳ $N_{part,opt} = \sqrt{t_{p,BL,256\times32} / t_{FO4}}$

$\boxed{N_{part,opt} \approx 5.06}$

Problem 3 – Variable Block Carry Bypass Adder (from S'06 HW)

Consider a 24-bit, 6 stage carry-bypass adder with the following delays: $t_{setup}=4$, $t_{carry}=1$, $t_{sum}=4$, $t_{bypass}=2$



a) Identify the critical path through the adder. List the delays for each block along the critical path and give the total delay. Assume that each stage bypasses the same number of bits.

The critical path is through the setup of the first stage (specifically, the first bit of the first stage), through all four bits of the first stage's carry chain, through the first five bypass multiplexers, three bits of the last stage's carry chain, then through the final stage's sum. The last carry bit of the final stage does not affect the sum, only the carry out. Carry out through the final multiplexer is not on the critical path as the sum is slower.

The delays for each component of this are t_{setup}, $4t_{carry}$, $3t_{bypass}$ $+2t_{bypass}$, $3t_{carry}$, and t_{sum} respectively. These add up to the expression given in equation 11.9 in the text, and substituting the given delays gives a total of 25. The critical path is grayed in figure 11-14.

b) Consider the setup delay and carry propagation of the second and third stages. These are not on the critical path and can be made slower without affecting performance. If we allow each stage to handle a different number of bits, what is the relationship between the number of bits per stage and the respective carry propagation delay? How many bits would you assign to each of the first three stages to minimize the delay from inputs to the carry output for the first 12 bits of the adder?

The worst case delay from the first stage's inputs through the setup, carry propagation and bypass to the start of the fourth stage is

$t_{setup} + M_0 * t_{carry} + 3t_{bypass}$ where M_0 is the number of bits in the first stage.

The delay for the second and third stages are similarly

$t_{setup} + M_1 * t_{carry} + 2t_{bypass}$ and

$t_{setup} + M_2 * t_{carry} + t_{bypass}$

Making all of these equal, we get that $M_1 - M_0 = M_2 - M_1 = t_{bypass}/t_{carry} = 2$. Thus, the first, second, and third stages should add 2, 4 and 6 bits respectively.

The original critical path is now 2 carries shorter, for a total delay of 23 at the final sum output (it is also acceptable to just give the delay to the end of the third stage carry being now 12 instead of 14). The second and third stages are now also critical paths as well, with the same delay.

c) How many bits would you assign to each stage in the second half of the adder? What is/are the delays along the critical path(s) now?

Same approach as for part b, except the critical paths are now from the carry in from the third stage, to the sum outputs. Delays for each path are:

$2t_{bypass} + (M_5\text{-}1)t_{carry} + t_{sum}$
$t_{bypass} + (M_4\text{-}1)t_{carry} + t_{sum}$
$(M_3\text{-}1)t_{carry} + t_{sum}$

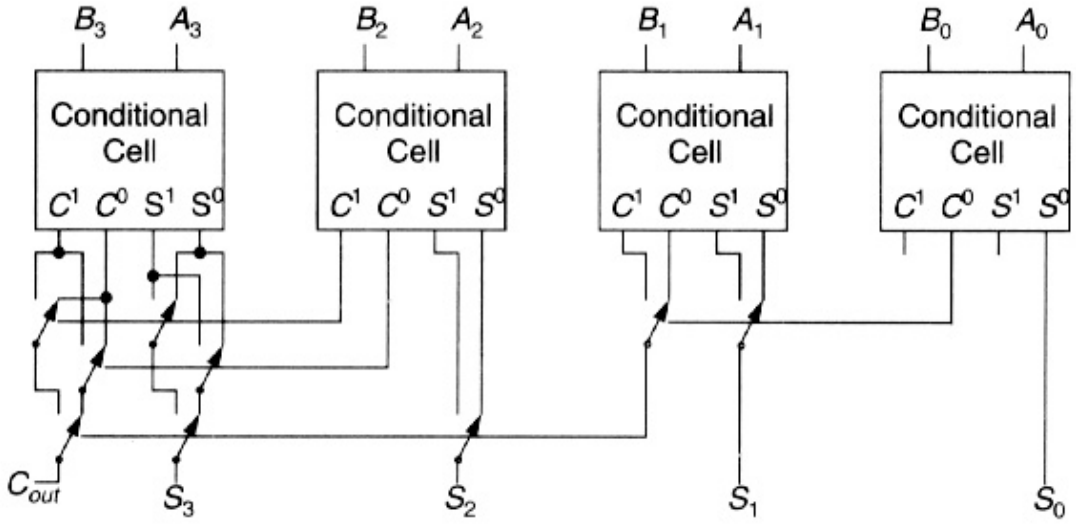
Making all of these equal, we get that $M_4\text{-}\ M_5 = M_3 - M_4 = t_{bypass}/t_{carry} = 2$. Thus, the fourth, fifth and sixth stages should add 6, 4 and 2 bits respectively.

The critical path to the final stage sum output is now another 2 carries shorter, for a total delay of 21. The fourth and fifth stage outputs are now also critical paths.

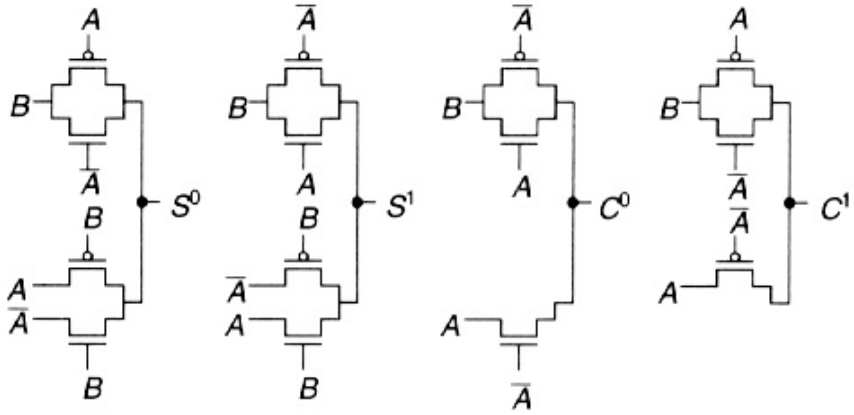
Note: parts a and c assumed that the sum logic for a bit has a delay of t_{sum} from its carry in to the sum out. From the structure of the mirror adder, one might consider t_{sum} to refer to the delay from the bit's own carry to its output, in which case the critical path delays would have one extra $t_{carry.}$ (M instead of M-1) This does not affect the choice of stage widths and is acceptable for the answers.

Problem 4 – Conditional Sum Adder (from F'06 HW)

Here is a neat adder structure called the conditional sum adder. Shown below is a 4-bit version of the circuit. Note that in the diagram, multiplexors are represented by switch-controlled arrows.



Using a pass-transistor implementation, the circuit schematic for each adder cell can be:



a) Derive Boolean equations for the four outputs of the one-bit conditional adder cell

S0 = A ⊕ B C0 = AB

S1 = (A ⊕ B)' C1 = A + B

b) Derive an expression for the propagation delay of the adder as a function of the number of Bits, *N*. Assume that the delay through each conditional cell is t_{cell} and that the delay of a MUX is t_{MUX}.

Basically, we need an additional level of multiplexors every time we double the number of bits. So…in the worst-case critical path, we must pass the carry-out from the least-significant bit all the way to the most-significant bit, which means that we must go through each MUX level. # of levels = ⌈log2 N⌉. On top of that, we always need to calculate C0/C1 and S0/S1, so we have one cell delay.

Delay = tcell + ⌈log2 N⌉ tMUX