University of California
College of Engineering
Department of Electrical Engineering
and Computer Sciences

E. Alon **Tuesday, December 15, 2009**
5:00-8:00pm

EECS 141: FALL 2009—FINAL EXAM

For all problems, 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

NAME	Last Solutions First

GRAD/UNDERGRAD	

Problem 1: _____/ 26

Problem 2: _____/ 19

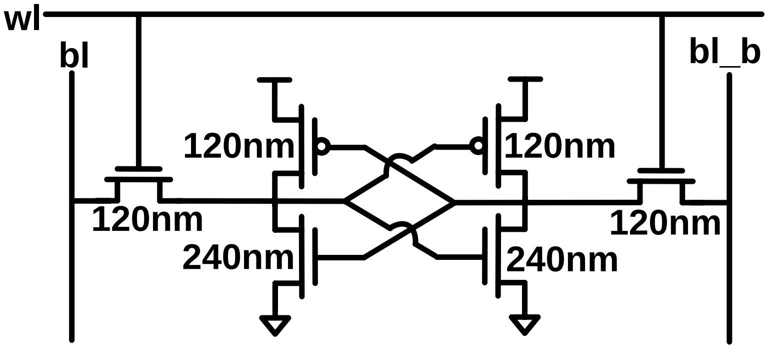
Problem 3: _____/ 22

Problem 4: _____/ 26

Total: _____/ 93

PROBLEM 1: SRAM Design (26 pts)

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.

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

$F = 15.36fF / 2fF = 7.68$

$\Pi B = 256/2 = 128$

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

$\Pi LE = \frac{4}{3} \cdot \frac{4}{3} \cdot \frac{4}{3} \approx 2.37$

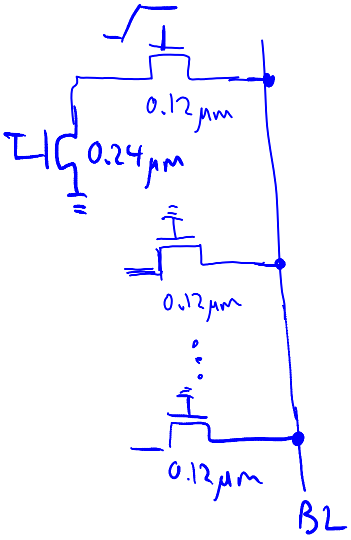
$PE \approx 2330.2$

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

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

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

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?



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

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

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

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

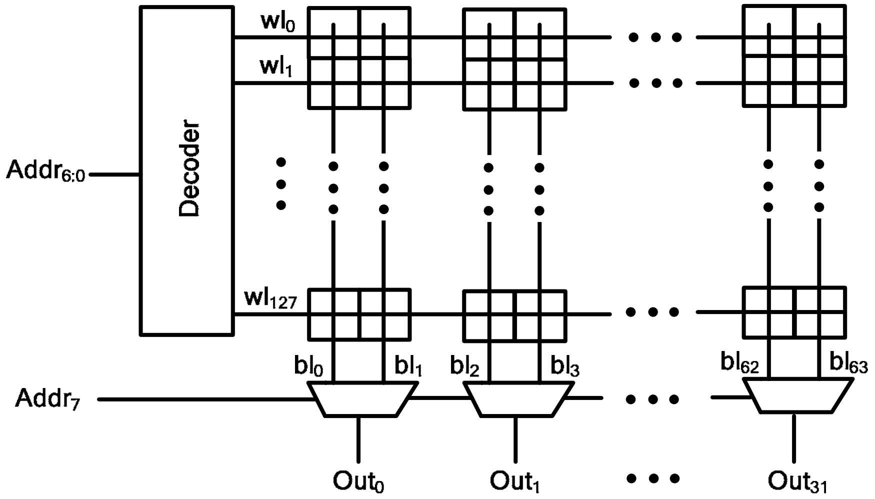
(including slope effect)

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

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

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

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: $PE_{dec,128\times64} = PE_{dec,256\times32}$

$\hookrightarrow t_{p,dec,128\times64} = t_{p,dec,256\times32} = 5.59\, t_{FO4}$

* 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). $t_{p,dec} = 5.59\ t_{FO4}$

* $t_{p,BL} = t_{p,BL,256\times32} / N_{part} = \dfrac{25.6\ t_{FO4}}{N_{part}}$

* $t_{p,mux} = N_{part}\ t_{FO4}$

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

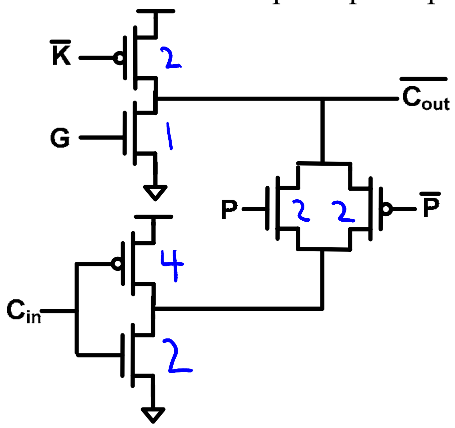
$$\frac{\partial t_{p,tot}}{\partial N_{part}} = -\frac{t_{p,BL,256\times32}}{N_{part}^2} + t_{FO4} = 0$$

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

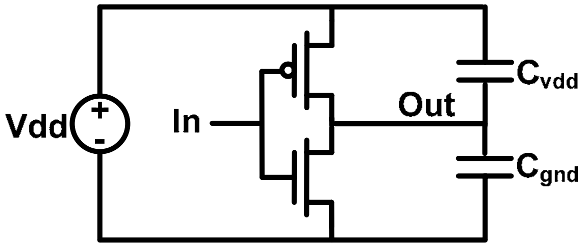
$$\boxed{N_{part,opt} \approx 5.06}$$

PROBLEM 2: Miscellaneous (19 points)

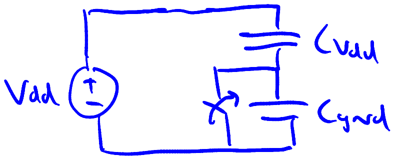
a) **(5 pts)** Assuming $R_{sqp} = 2*R_{sqn}$ and quadratic devices, size the gate shown below so that the worst-case pull-up and pull-down resistances are equal.



b) **(6 pts)** As a function of Vdd, C_{vdd}, and C_{gnd}, how much energy is pulled out of the supply voltage Vdd when In transitions from low to high? How about when In transitions from high to low? Note that you can ignore all capacitors associated with the transistors except those explicitly drawn in the figure.



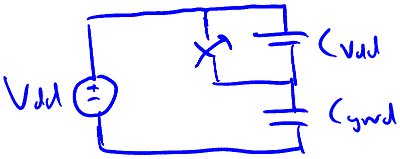
In 0→1 :



Charging Cvdd up to Vdd, so:

$$\boxed{E_{In\,0\to1} = C_{vdd}\cdot V_{dd}^2}$$

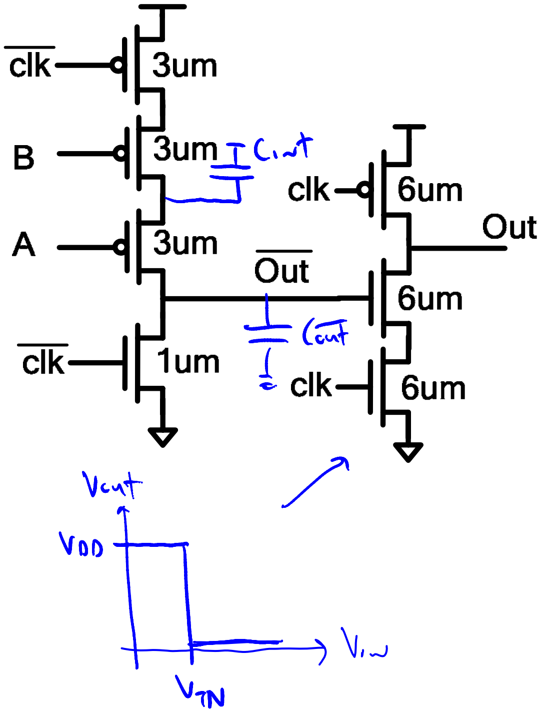
In 1→0 :



Charging Cgnd up to VDD, so:

$$\boxed{E_{In\,1\to0} = C_{gnd}\cdot V_{dd}^2}$$

c) **(8 pts)** One of your colleagues (who didn't take EE141) shows you the circuit below and complains that "something is wrong with HSPICE" because their simulations show that the circuit functions correctly for Vdd = 0.6V, but that the output isn't always correct when Vdd=1.2V. Assuming that C_G=2fF/µm, C_D=1.5fF/µm, and that V_{TN}=|V_{TP}|=0.3V, explain why the circuit really doesn't work at Vdd=1.2V, and calculate the maximum supply voltage for which the circuit will function correctly. (Hint: Using the simple switch model, what are the VTCs of the dynamic gates shown below?)



* The problem is charge sharing between C_{int} (which may get charged up to V_{DD}) and $C_{\overline{out}}$ (which is pre-discharged to ground).

* When charge share happens, if $V_{\overline{out}} > V_{TN}$, second gate fires and get wrong output.

Q conservation: $V_{DD}\cdot C_{int} = V_{\overline{out}}\cdot(C_{int}+C_{\overline{out}})$

$$V_{\overline{out}} = \frac{C_{int}}{C_{int}+C_{\overline{out}}}\cdot V_{DD}$$

$$C_{\overline{out}} = 6\mu m\cdot C_G + 4\mu m\cdot C_D = 18fF$$

$$C_{int} = 6\mu m\cdot C_D + 3\mu m\cdot C_G = 15fF$$

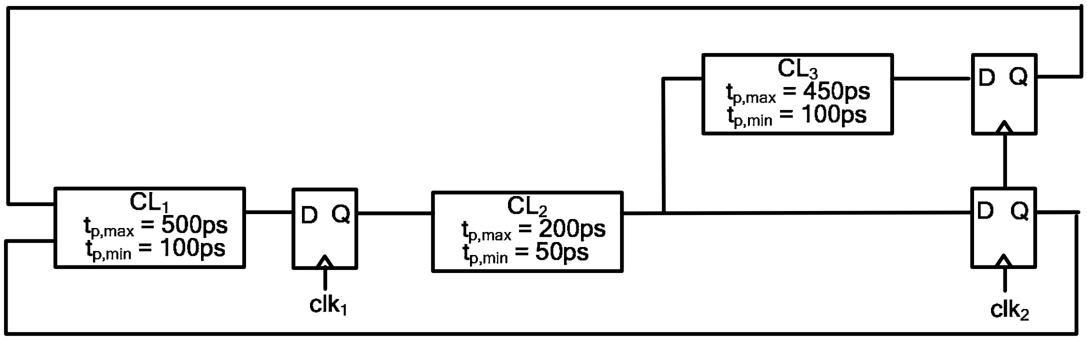
(Note that when $V_{DD} < 0.66V$, you actually get incomplete charge sharing)

So: $$V_{DD}\cdot\frac{15fF}{15fF+18fF} < V_{TN}$$

$$\boxed{V_{DD} < 0.66V}$$

PROBLEM 3: Timing and Clock Distribution (22 points)

In this problem we will be examining the pipeline shown below. The minimum and maximum delays through the logic are annotated on the figure, and the flip-flops have the following properties: $t_{clk\text{-}q}$ = 50ps, t_{setup} = 50ps, and t_{hold} = 50ps. You can assume that the clock has no jitter.



a) (6 pts) Assuming there is no skew between clk_1 and clk_2, what is the minimum clock cycle time for this pipeline? Are there any minimum delay (hold time) violations?

$$* \quad T_{cyc,min} = t_{clk\text{-}q} + t_{p,max2} + t_{p,max3} + t_{setup}$$

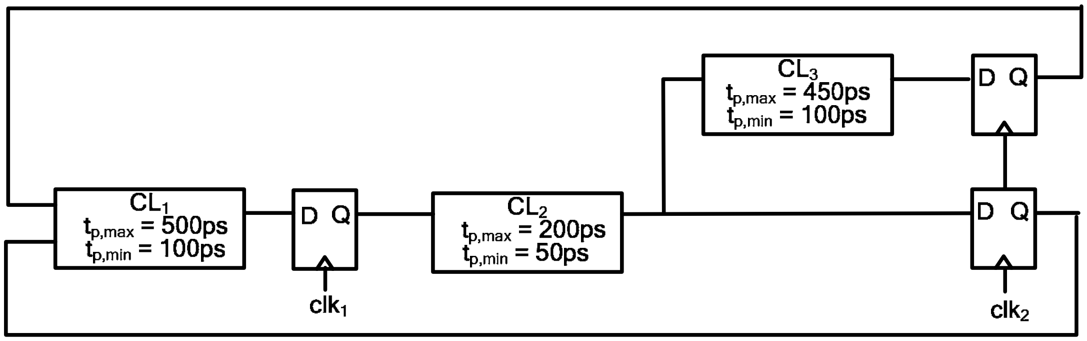
$$T_{cyc,min} = 50ps + 200ps + 450ps + 50ps$$

$$\boxed{T_{cyc,min} = 750ps}$$

$$* \quad t_{clk\text{-}q} + t_{p,min2} \geq t_{hold}$$

$$50ps + 50ps \geq 50ps \quad \checkmark$$

↳ No minimum delay violations.



b) (6 pts) Now let's assume that clk₁ and clk₂ can be skewed relative to each other by up to +/-60ps. Now what is the minimum clock cycle time? Are there any minimum delay violations?

* $T_{cyc,min} - t_{skew} = t_{clk-q} + t_{p,max2} + t_{p,max3} + t_{setup}$ (clk₂ early)

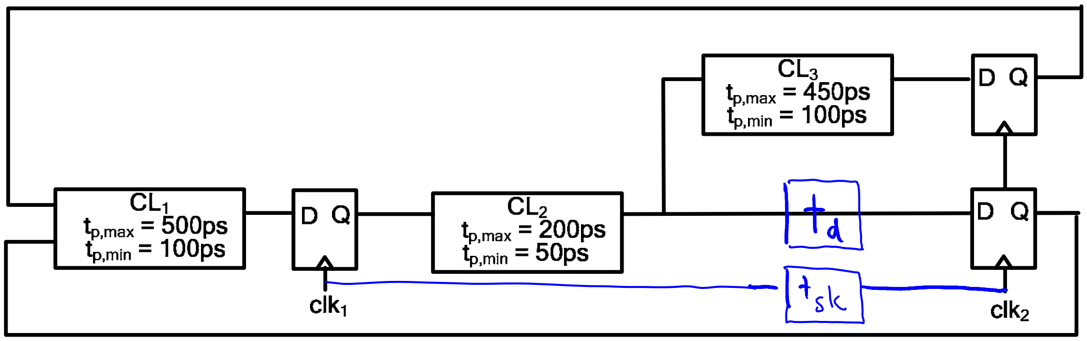
$T_{cyc,min} - 60ps = 50ps + 200ps + 450ps + 50ps$

$$\boxed{T_{cyc,min} = 810ps}$$

* $t_{clk-q} + t_{p,min2} > t_{hold} + t_{skew}$ (clk₂ late)

$50ps + 50ps > 50ps + 60ps$

$100ps > 110ps$ ← No! Have a hold time violation now.



c) **(10 pts)** Continuing to assume that there can be +/-60ps of random skew between clk$_1$ and clk$_2$, can you introduce any intentional skew and/or delay into the circuit in order to reduce the minimum cycle time without causing any hold time violations? If so, you should indicate where in the pipeline you want to add skew or delay, calculate what the new cycle time is, and prove that you will not have any hold time violations. If not, you should explain why this it isn't possible to improve the cycle time without introducing a hold time violation.

* What we need to do is fix the hold time violation without slowing down the critical path.

* As shown above, simplest way to do this is to add a delay td between CL_2 and the following flop.

* How much td is needed? Want to steal enough time that max. paths through CL_1 and CL_2/CL_3 have zero margin. To do this, look at difference between $t_{p,max2}$ and $t_{p,max1}$:

$$\Delta t = (t_{p,max2} + t_{pmax3} - t_{p,max1}) = 150ps$$

* Splitting the difference equally means we want clk_2 to arrive $\boxed{t_{sk} = \Delta t / 2 = 75ps}$ after clk_1.

* To meet hold time: $t_{clk-q} + t_{min2} + t_d \geq t_{hold} + t_{sk} + t_{skew}$

↳ $\boxed{t_{d,min} = 85ps}$

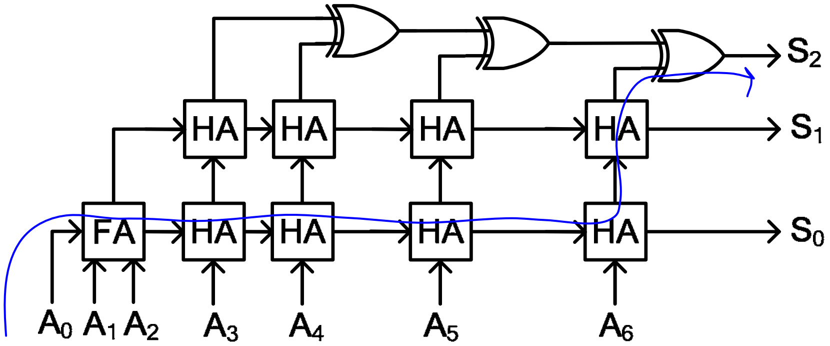
* Now: $T_{cyc,min} = t_{clk-q} + t_{p,max2} + t_{p,max3} + t_{setup} + t_{skew} - t_{sk}$

$$\boxed{T_{cyc,min} = 735ps}$$

PROBLEM 4: Arithmetic (26 pts)

In this problem we will look at designing a circuit that adds together 7 1-bit binary numbers $A_{6:0}$ into one 3-bit output $S_{2:0}$ (whose value ranges from 0 to 7).

a) **(4 pts)** Shown below is a simple implementation of this circuit that uses only half adders (HA), full adders (FA), and XOR gates. Assuming that the sum and carry delays of the half and full adders are equal to each other and to the delay of the XORs, how many gate delays are there on the critical path for this circuit? (Please mark this path on the schematic.)



* Critical path : $\boxed{\text{7 gate delays}}$

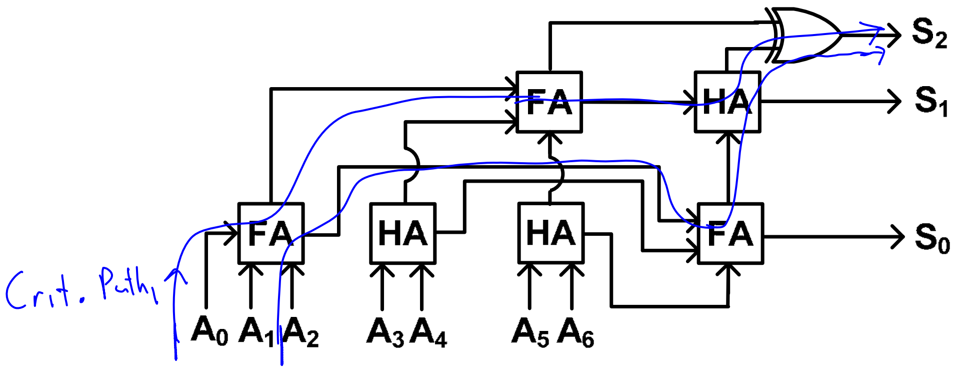
b) **(4 pts)** If each FA requires $10\mu m^2$ of area, each HA requires $6\mu m^2$, and each XOR requires $4\mu m^2$, how much total area is occupied by the circuit from part a)?

* Area : 1FA + 8HA + 3XOR

↳ Area = $10\mu m^2 + 8 \cdot 6\mu m^2 + 3 \cdot 4\mu m^2$

$\boxed{\text{Area} = 70\mu m^2}$

c) (8 pts) Ace says she has a better implementation for this circuit and shows you the schematic below. Under the same assumptions as part a) and b), how many gate delays are on the critical path of this circuit, and how much area does it consume? Once again, please highlight the critical path on the schematic.



Crit. Path2 ...

(There are actually multiple critical paths - any correct one will receive full credit.)

$\boxed{\text{Critical path: 4 gate delays}}$

Area: 3FA + 3HA + 1XOR

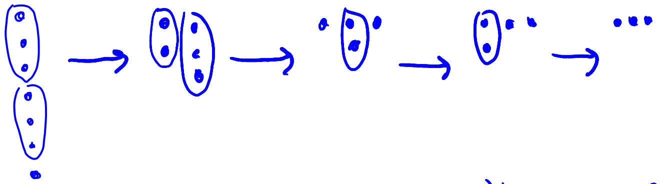
Area = $3 \cdot 10\mu m^2 + 3 \cdot 6\mu m^2 + 4\mu m^2$

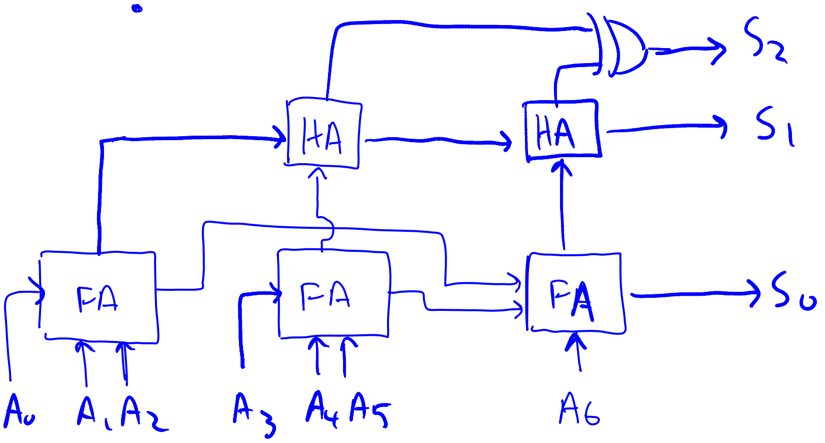
$\boxed{\text{Area} = 52\mu m^2}$

d) (10 pts) Still using only full adders, half adders, and XORs, and without increasing the delay, draw an implementation for this circuit that uses even less area than the one from part c). How much area does this implementation require?

Solution #1:

*Rather than consume all of the input bits right away 1 FA and 2HA's, we can do a little bit better by just using 2 FAs in the first stage. We can then replace an FA with an HA in the second stage.





*Critical path still $\boxed{4 \text{ gates}}$

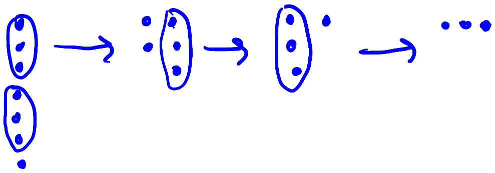
*Area: 3 FA + 2 HA + 1 XOR

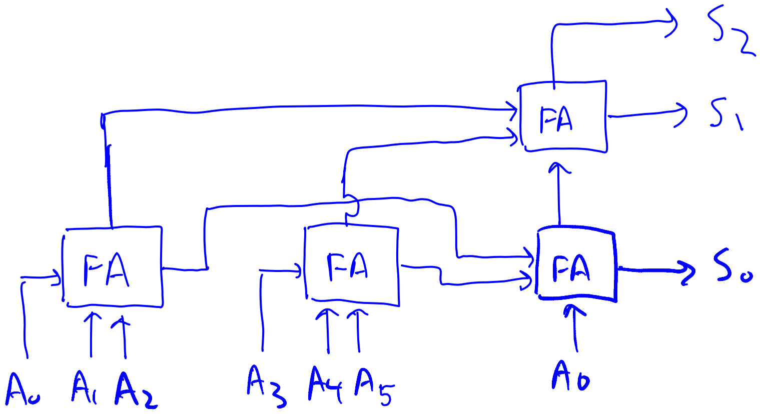
$$\text{Area} = 3 \cdot 10\,\mu m^2 + 2 \cdot 6\,\mu m^2 + 4\,\mu m^2$$

$$\boxed{\text{Area} = 46\,\mu m^2}$$

Solution #2:

Can actually do even better than this by once again replacing some HA's with FA:





Critical path now $\boxed{\text{3 gate delays}}$

Area: 4FA

$$\boxed{\hookrightarrow \text{Area} = 40\,\mu m^2}$$