EECS 151/251A Homework 10

Due Sunday, April 22nd, 2018

Problem 1: LFSR

A particular linear feedback shift register (LFSR) is built using the primitive polynomial $x^{20}+x^3+1$. How many unique states does it have?

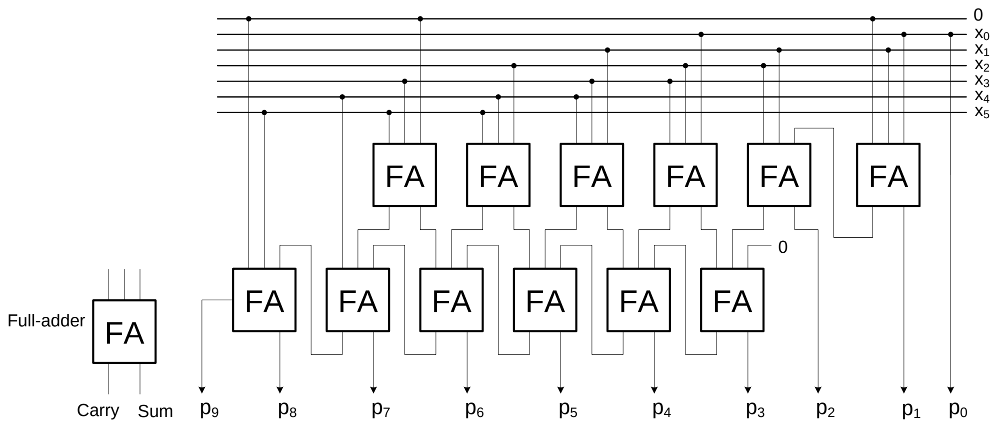
Problem 2: Constant Multiplier

Consider the design of a circuit for multiplying a constant, C, with a signed two-complement variable, X, such that $Y = C \times X$.

In this problem, let $C = 13_{10}$, and assume X is a 6-bit variable. Using only *full-adder blocks* (1-bit adders) draw a multiplier circuit. Your design objective is to minimize the total number of full-adder blocks, as well as the delay from input to output. Give priority to cost over delay.

Problem 3: Another Constant Multiplier

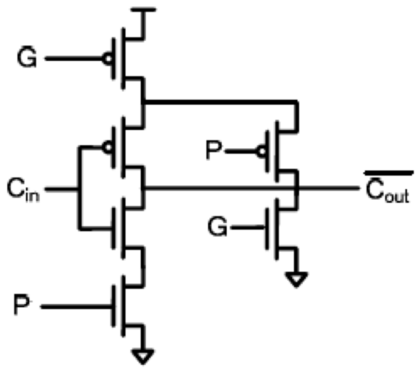
The circuit shown below is used to multiply the 6-bit number X by a 6-bit constant value, C. It is made up of instances of a full-adder cell. The full-adder takes as input 3 1-bit signals and outputs a 1-bit sum and a 1-bit carry.



What is the value of C?

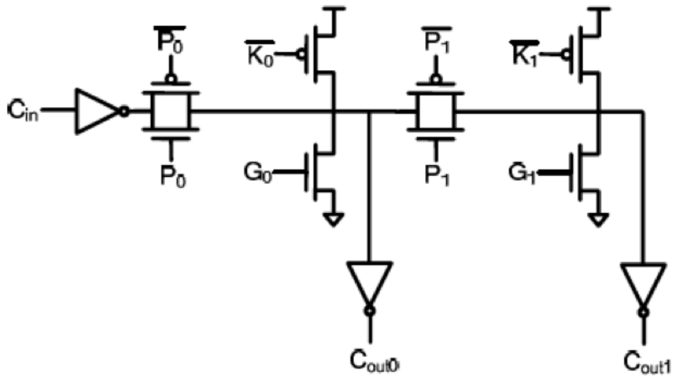
Problem 4: Adders

(a) Shown below is a static CMOS implementation of a gate that computes the carry-out at a particular bit position. If the P signal fed into the gate is calculated using $P = A+B$ (instead of $P = A \oplus B$, which is usually the case) would the output of this gate still be correct? Why or why not? If not, suggest a modification that gives the right output.



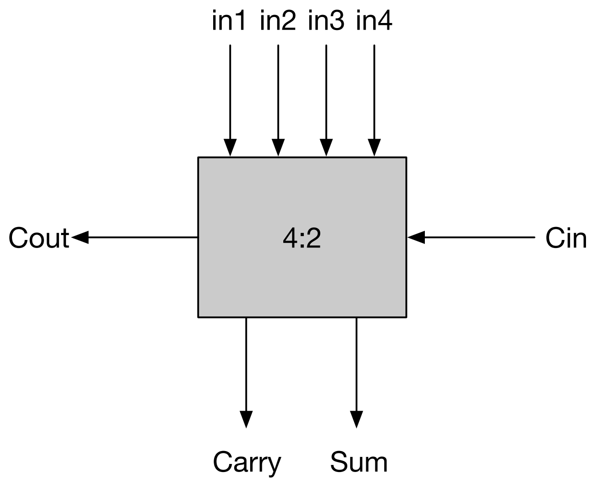
(b) Show the static CMOS implementation of a gate that computes the sum at a particular bit position ($S = P \oplus C_{in}$). Again, if the P signal fed into the gate is calculated using $P = A+B$, is the output of the gate correct? Why or why not? If not, suggest a modification that gives the right output.

(c) The gate shown below is called a Manchester carry chain, and it computes the carry-out for two bit positions. Does this gate give the correct output if $P = A + B$? Why or why not?



Problem 5: Multipliers

(a) Implement the 4:2 compressor described in the truth table below using NOT, AND, OR, and XOR gates. C_{out} and $Carry$ are both weight 2, while Sum and all inputs are weight 1. In the truth table, N refers to the total number of in signals at logic 1.



N	C_{in}	C_{out}	$Carry$	Sum
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	1	0	1
4	0	1	1	0
0	1	0	0	1
1	1	0	1	0
2	1	0	1	1
3	1	1	1	0
4	1	1	1	1

(b) Use this 4:2 compressor to implement a 6x6 unsigned Wallace-tree multiplier.

(c) Now use a 3:2 compressor (a full adder) to implement a 6x6 unsigned Wallace-tree multiplier. How many more reduction layers do you need?

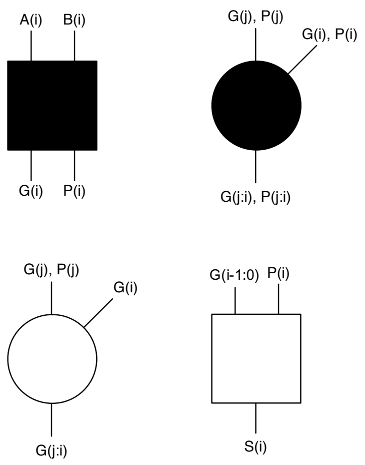
(d) Convert your multiplier from (b) to a signed multiplier. How many more compressors did you use?

Problem 6: Tree Adders

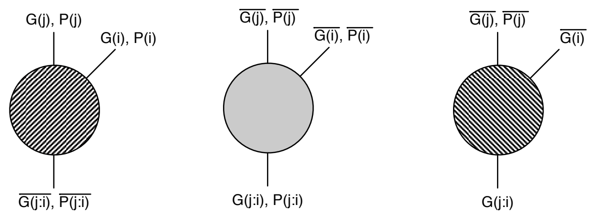
The goal of this problem is to design a 10-bit Kogge-Stone adder optimized for delay:

(a) Design the following logic blocks at a gate level. You may draw the gates or write the Boolean functions. Review your notes or the textbook to determine what goes inside each block. There are also numerous resources available online about these logarithmic adders. Use the given inputs and outputs as hints.

(b) Using the logic blocks you designed in part (a), design a 10-bit logarithmic adder with a carry input and a carry output. Use a radix-2 Kogge-Stone implementation. What is the critical path of your design? Give a block-level estimate, assuming that more complex blocks have more delay.



(c) **EECS 251A Only.** To save logic depth, we can create inverting logic stages instead of non-inverting logic stages. This removes the inverter at the output of each prefix block (the block that creates G(j:i) and P(j:i)). Design the following logic blocks and use them to modify your logarithmic adder from before. How many inverters were removed from your critical path?


G(j), P(j)
G(i), P(i)
$\overline{G(j:i)}$, $\overline{P(j:i)}$
$\overline{G(j)}$, $\overline{P(j)}$
$\overline{G(i)}$, $\overline{P(i)}$
G(j:i), P(j:i)
$\overline{G(j)}$, $\overline{P(j)}$
$\overline{G(i)}$
G(j:i)