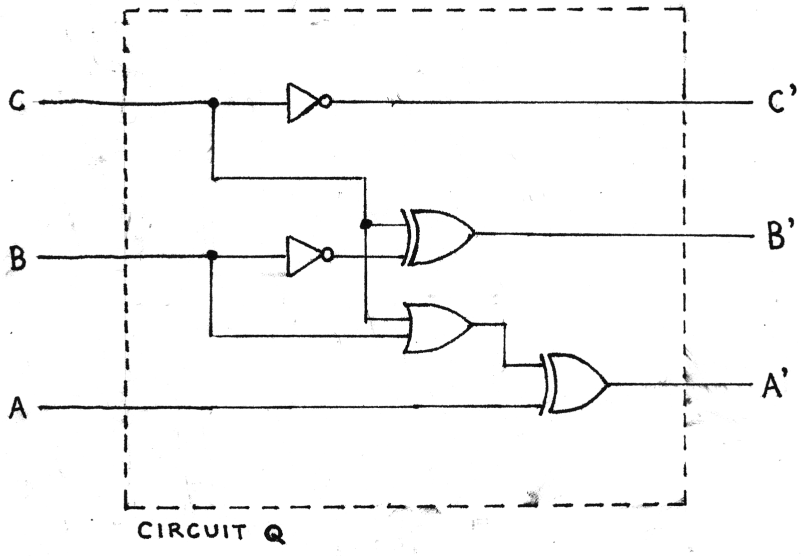
6·004 RECITATION VI

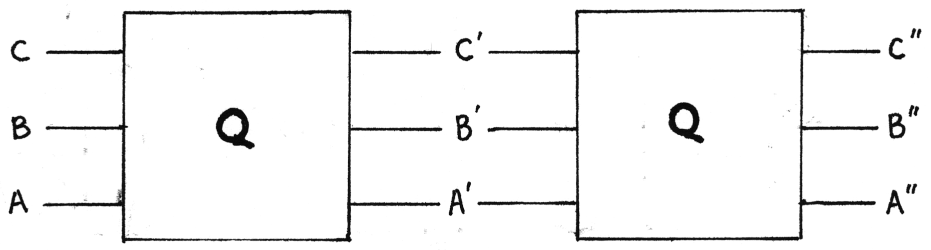
1 Mount Lassen to Mount Lytton

Calculating propagation delays

Consider the circuit **Q** drawn below. Its specification states that, if one input is changed or multiple inputs are changed simultaneously, then the output will be settled after 170 ps; in other words, the propagation delay of the critical path is 170 ps.



Is this enough information to determine the propagation delay of the following cascade? If so, provide an argument. If not, provide an argument or construct a counterexample.



No. The critical path from the inputs (with propagation delay of 170 ps) to the middle layer is either INV + XOR (from input to B’) or OR + XOR (from input to A’). The critical path from the inputs to the output layer is either 2 INV + 2 XOR (from input to B”) or INV + OR + 2 XOR (from input to A”). In either case, knowing that either 170 = INV + XOR or 170 = OR + XOR is not enough to determine the value of 2 INV + 2 XOR or INV + OR + 2 XOR.

PROPAGATION DELAY FROM INPUTS TO EACH LABEL

C	**C’**	**C”**
0	INV	2 INV
B	**B’**	**B”**
0	INV + XOR	max(INV, INV + XOR + INV) + XOR
		= 2 INV + 2 XOR
A	**A’**	**A”**
0	OR + XOR	max(max(INV, INV + XOR) + OR, OR + XOR) + XOR
		= max(INV + XOR + OR, OR + XOR) + XOR
		= (INV + XOR + OR) + XOR
		= INV + OR + 2 XOR

2 Reach Into the Cookie Jar

Revisiting the take-home problem from lecture

Find a minimal sum of products for the expression $A\bar{B}\bar{C} + A\bar{B}C + \bar{A}\bar{B}C + \bar{A}BC$. Then, using the following gate library, design an implementation that minimizes area.

GATE	INV	AND2	NAND2	OR2	NOR2	AND3	NAND3	OR3	NOR3
DELAY	20	50	30	55	35	90	70	100	80
AREA	10	25	15	26	16	40	30	42	32

TODO

Curious about the section titles in the labs, and the question titles in the worksheets? Some are just references to songs, games, or literature; some are puzzles of a sort. What's the connexion between revisiting a problem from lecture and reaching into a cookie jar? "Follow-through." We're following-up with you after lecture, and pretending to reach into the cookie jar is common advice for proper follow-through when you're shooting a basketball.

3 If You Can't ———, Reuse; If You Can't Reuse, Recycle

Designing an incrementer by simplifying a ripple-carry adder

A 4-bit ripple-carry adder can be built from four full adders by forwarding the carry-out of one digit to the carry-in of the next. If $A_{3...0}$ and $B_{3...0}$ are two four-digit binary numbers, and $S_{3...0}$ is their sum (modulo 16), then the circuit for each full adder is

$$S_n = A_n \oplus B_n \oplus C_n$$

$$\text{Overflow}_n = A_n B_n + (A_n \oplus B_n) C_n$$

where the carry-in, C_n, for digit n is equal to the carry-out, Overflow_{n-1}, of digit $n-1$.

But suppose we wished to design an incrementer – a circuit that always adds 1 to its input. We can do this with our ripple-carry adder by tying $B_{3...0}$ to 0000 and setting C_0 to 1.

Now simplify the circuit using the properties of Boolean algebra. In terms of gate count, what percentage of the size of the original ripple-carry adder is the size of the incrementer?

Less than 40%. See the following page.

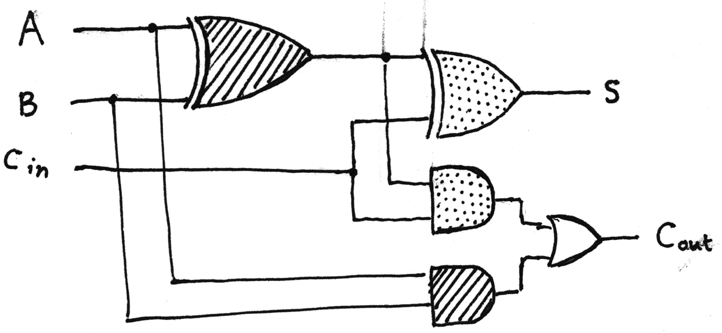
$S = A \oplus B \oplus C_{in}$

$C_{out} = AB + BC_{in} + C_{in}A$

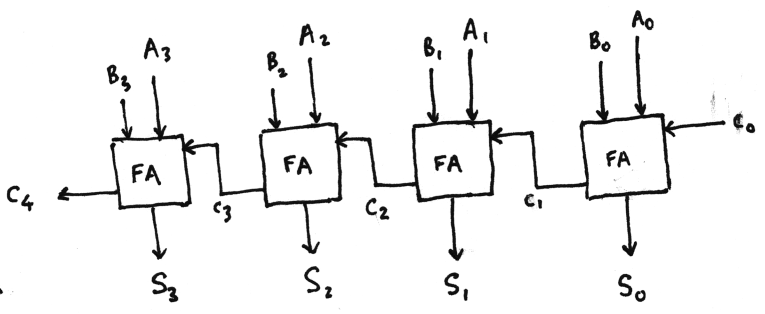
$= AB + (A \oplus B)C_{in}$

▨ Half-Adder №1

⬚ Half-Adder №2



FULL ADDER



4-BIT
RIPPLE CARRY ADDER

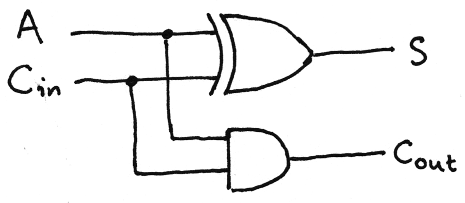
As an incrementer, $B_3B_2B_1B_0 = 0000$ and $C_0 = 1$.

$S = A \oplus \overset{0}{\cancel{B}} \oplus C_{in} = A \oplus C_{in}$

$C_{out} = A\overset{0}{\cancel{B}} + \overset{0}{\cancel{B}}C_{in} + C_{in}A = AC_{in}$

$= A\overset{0}{\cancel{B}} + (A \oplus \overset{0}{\cancel{B}})C_{in} = AC_{in}$

Now it is just a half-adder, and our circuit is 40% the $\overset{\text{previous}}{\vee}$ size!



(Actually even less if you implement the XOR as AND-OR-NOTs, since you can reuse one of the ANDs to generate C_{out}).