Boolean Circuit Problems

Problem 1. Adding It Up

How could we use boolean circuits to build a calculator? Let's investigate...

a) Suppose that you have a one-bit number A, and a one-bit number B. What truth table represents the product of multiplying A times B? What circuit could you build to compute that truth table?

A	B	Product
0	0	0
0	1	0
1	0	0
1	1	1

Circuit:

A and B

b) Suppose now that you want to add A and B instead of multiplying them. However, 1+1 is 2, which is bigger than either 0 or 1 – so now we need two circuits: one to compute the *sum* bit, and one to compute a *carry* bit. Fill in the rest of the truth table, and write the two circuits.

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Circuit for *Sum*:

A xor B

Circuit for *Carry*:

A and B

c) An "adder" circuit component, as used to add up numbers longer than just one bit, actually needs to add up three bits – A, B, and C. However, we still only have the *sum* bit and the *carry* bit as the two outputs. Write the two circuits for this truth table.

A	B	C	Sum	Carry
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Circuit for *Sum*:

Sum =

In the most straightforward-to-solve form:
(not-A and B and not-C) or (A and not-B and not-C) or (not-A and not-B and C) or (A and B and C)

Alternatively, the most concise form:
(A xor B) xor C

Circuit for *Carry*:

Carry =

In the most straightforward-to-solve form:
(A and B and not-C) or (not-A and B and C) or (A and not-B and C) or (A and B and C)

Alternatively, the most concise form:
((A xor B) and C) or (A and B)

Problem 2. Functional Completeness

A set of logic gates is called ***functionally complete*** if you can use those logic gates to construct any other logic gate.

a) {AND, OR, NOT} is a set of functionally complete logic gates. Demonstrate this by example: construct an XOR gate using only AND, OR, and NOT. Draw either a circuit or a formula for XOR in the space below.

XOR = (not (A and B)) and (A or B)

b) NAND is functionally complete: all you need are NAND gates in order to make *any* other gate. Demonstrate this by constructing {AND, OR, NOT} from NAND gates. Draw either a circuit or a formula for each logic gate.

NOT

A nand A

AND

NOT (A nand B)
… using NOT from above

OR

(not A) nand (not B)

Assuming {AND, OR, NOT} is functionally complete, you've just proved that NAND is, also!

Problem 3. No Inverters?

from 6.045 Problem Set 1, Spring 2013

A Boolean circuit *C* is called ***monotone*** if changing any of the *n* input wires $x_1 \ldots x_n$ from 0 to 1 can only ever change the output $C(x_1 \ldots x_n)$ from 0 to 1, never from 1 to 0.

a) Argue that any circuit made of AND and OR gates must be monotone. (Discuss with partner)

b) Make an example of a truth table for a monotone circuit which has three inputs. Write in the truth table below. (There are many possible correct answers.)

x_1	x_2	x_3	$C(x_1,x_2,x_3)$
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

c) Show that your truth table can be constructed from AND and OR gates – no NOT gates needed! Draw the circuit that creates this truth table, below. You may use the constants 0 and 1 as "freebies" where necessary, as inputs instead of x_1, x_2, and x_3.

d) Argue that any monotone truth table must be able to be made by a circuit of AND and OR gates. (Discuss with partner)

Problem 4. Inverter Alchemy!

*This problem is **very challenging**. If you decide to tackle it, you might want to download Logicly <http://logic.ly/> and use that to play around (instead of solving on paper).*

Sometimes, people just expect the impossible. You were given only two inverters (aka NOT gates) to work with. But your assignment is to take in three inputs – A, B, and C – and produce their inversions: not A, not B, and not C.

Strange though it sounds, this task is *actually possible!* Prove it by figuring out what circuit will accomplish the task.