The UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Comp 411 Computer Organization
Fall 2012

Problem Set #4 Solutions

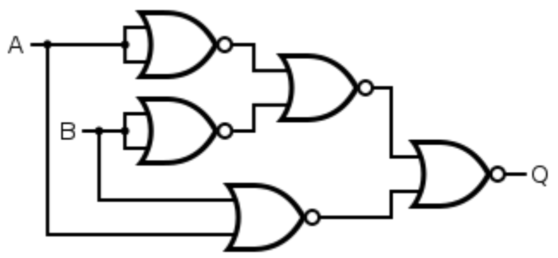
Issued Monday, 11/5/12; Due Monday, 11/12/12

Note: You may enter your answers in the space provided, or attach additional sheets of paper.

Problem 1. Circuits to Truth Table (16 points)

For each of the parts, simply draw a truth table corresponding to the given circuit. If there are more than one outputs, draw them as separate columns in the same truth table.

a)

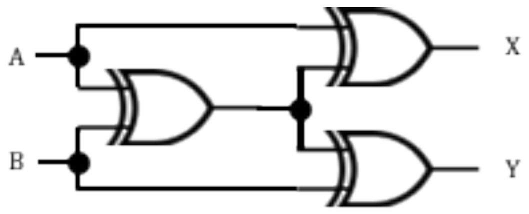


A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

From the truth table, do you recognize what this circuit does?

Answer: The circuit's output is the XOR of A and B.

b)



A	B	X	Y
0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	1

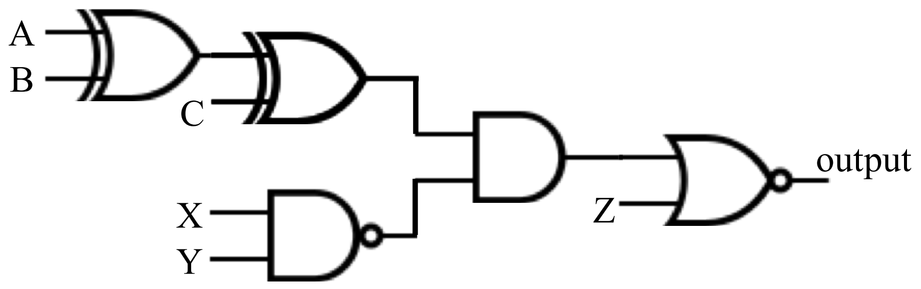
From the truth table, do you recognize what this circuit does?

Answer: The circuit swaps the inputs, so X=B and Y=A.

Problem 2. Equations to Circuits (12 points)

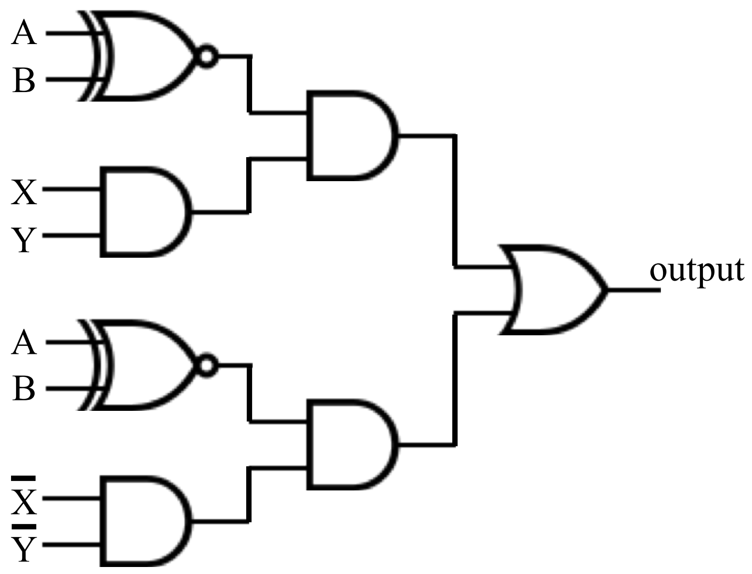
For each of the parts, simply convert the given Boolean expression directly into a circuit diagram consisting of basic gates. *Do not implement using transistors!* Please do not perform any simplification/optimization. Simply replace each Boolean operation by the appropriate gate. You can assume you have all of these gates available (with any number of inputs): AND, OR, inverter, NAND, NOR, XOR, XNOR. Also, you may assume that for each input, its complement is also available (e.g., both X and $\overline{X}$ are available).

a) $Y = \overline{(A \oplus B \oplus C) \cdot (\overline{XY}) + Z}$

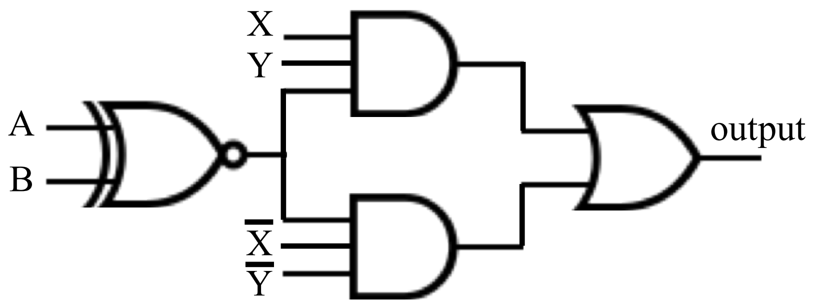


Note: Other variations are possible, including those that use fewer gates. For example, the two 2-input XOR gates can be replaced by a single 3-input XOR gate.

b) $Y = (A \overline{\oplus} B) \cdot XY + (A \overline{\oplus} B) \cdot \overline{X} \cdot \overline{Y}$



Note: Several other variations are possible, including those that use fewer gates (but you were not required to consider any of these simplifications). For example, each of the top and bottom halves uses two 2-input AND gates in sequence; each such pair can be replaced by a 3-input AND gate. Further, XNOR(A,B) is calculated twice; a simpler implementation would compute it only once. The picture below shows these optimizations. (Even further simplification is possible, but let's not worry about that.)



Problem 3. Complex CMOS Gates (26 points)

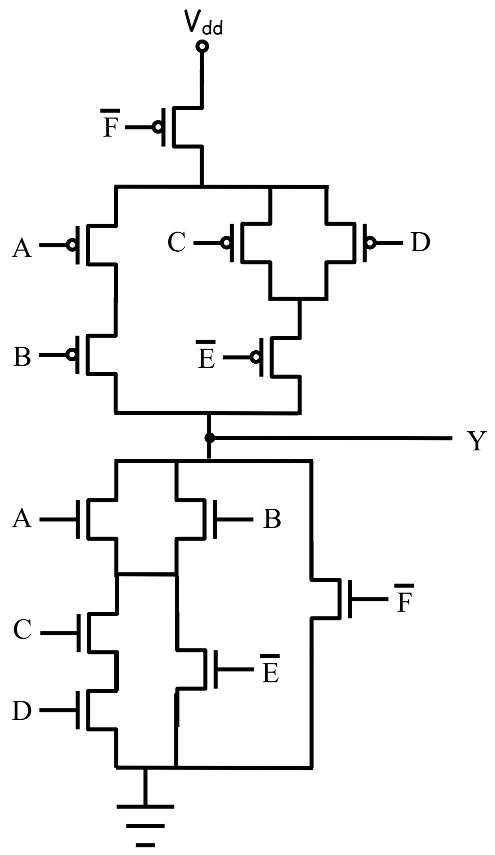
For each of the parts, you are to draw a single CMOS gate that implements the given function, *using transistors*. That is, draw a single gate with complementary pull-up and pull-down networks (using p-type and n-type transistors, respectively). Be sure that the circuit you draw corresponds exactly to the expressions given, i.e., do not perform any simplification. You may assume that for each input, its complement is also available (e.g., both A and $\overline{A}$ are available).

a) $Y = \overline{(A+B)(CD+\overline{E})+\overline{F}}$

Note carefully: The *E* and *F* in the equation have bars above them (i.e., they are complemented), and the entire expression for *Y* also has a bar above it.

Answer: The CMOS gate implementation is shown to the right. For this function, it is easier to draw the pull-down network first, and then the pull-up network is simply its complement.

The key thing to remember is that when the pull-down network is activated, it forces the gate output to be '0'. Since the original equation for *Y* has a big bar over it, it allows us to easily determine that the Boolean expression underneath it, i.e., $(A + B)(CD + \ldots)\ldots$ must be true for the pull-down network to be activated. From this information, the pull-down network is easily determined as shown. Finally, the top half of the circuit is simply the complement of the bottom half.

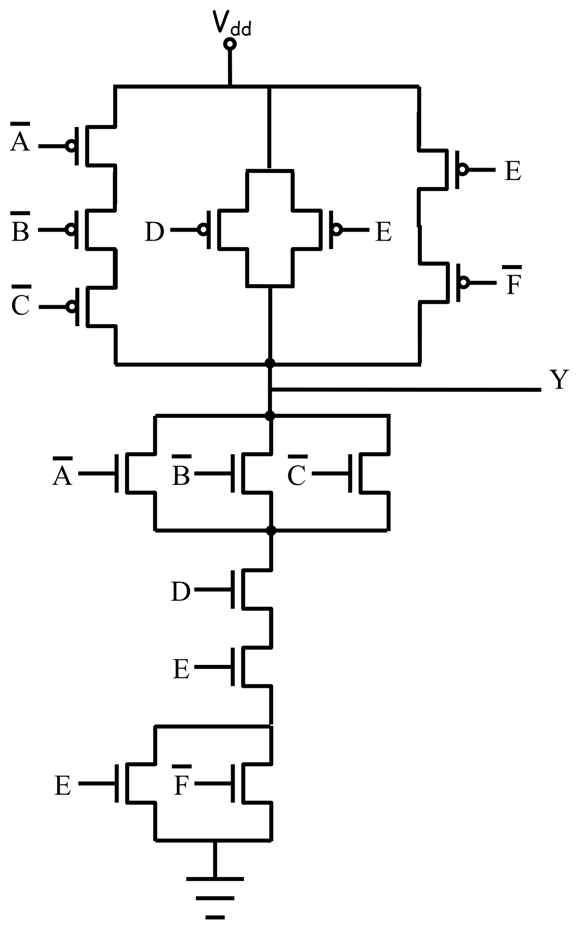


b) $Y = ABC + \overline{DE} + \overline{E}F$

Note carefully: The entire term *DE* has a bar over it; remember that $\overline{DE}$ is *not* the same as $\overline{D} \cdot \overline{E}$. Also, only *E* in the last term has a bar over it.

Answer: The CMOS gate implementation is shown to the right. For this function, it is easier to draw the pull-up network first, and then the pull-down network is simply its complement.

The key thing to remember is that when the pull-up network is activated, it forces the gate output to be '1'. The original equation for *Y* tells us when *Y* is '1', and therefore tells us when the pull-up network should be activated. From this information, the pull-up network is easily determined as shown. *Note here that a pfet is switched on when its input is zero, hence all the inputs to the circuit are complemented versions of the inputs in the given equation.* Finally, the bottom half of the circuit is simply the complement of the top half.



Problem 4. Sum of Products (26 points)

For each of the parts, do the following in this order: (i) first draw a truth table, (ii) then give the *sum-of-products* Boolean expression for the output, and (iii) finally, draw a circuit diagram. You may only use inverters, AND gates and OR gates for your circuit. The AND and OR gates can have two or more than two inputs, as many as you need. You do not need to simplify/optimize your Boolean expression. You may assume that for each input, its complement is also available (e.g., both A and $\overline{A}$ are available).

a) Implement a function *F* whose inputs are *A, B* and *C,* such that the value of *F* is the same as the majority of the inputs (i.e., at least two out of three inputs have that value).

Truth table:

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Sum of products expression:

$$F = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

Simplification is not necessary (and beyond the scope of this course), but you would get:

$$F = AB + BC + AC$$

Circuit diagram: Solution is not provided, but should be quite straightforward. In particular, for the unoptimized expression, you will use 4 AND gates, one for each term, and then one OR gate to combine the four terms together.

b) Implement a function *F* whose inputs are *A, B, C* and *D,* such that the value of *F* is the same as the majority of the inputs (i.e., at least three out of four inputs have that value), but if there is a 2-2 tie, then the value of *F* is simply the value that *A* has.

Truth table:

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Sum of products expression:

$$F = \overline{A}BCD + A\overline{B}\,\overline{C}D + A\overline{B}C\overline{D} + A\overline{B}CD$$
$$+AB\overline{C}\,\overline{D} + AB\overline{C}D + ABC\overline{D} + ABCD$$

Again, simplification is beyond the scope of this course, but a simpler sum-of-products express is:

$$F = AB + AC + AD + BCD$$

Circuit diagram: Solution is not provided, but should be quite straightforward. In particular, for the unoptimized expression, you will use 8 AND gates, one for each term, and then one OR gate to combine the eight terms together.

Problem 5. "Bits of Floating-Point" (20 points)

Represent the following in *single-precision* IEEE floating point. Give your answers in *hexadecimal.* Enter the answers in the table below.

a) -205.0
b) 60.125
c) (2^{11} -1)

Decimal	***S field***	***E field (binary)***	***F field (binary)***	***Complete Number (Hex)***
-205.0	1	10000110 (134-127 = 7)	~~1~~.100 1101 0000 0000 0000 0000	0xC34D0000
60.125	0	10000100 (132-127 = 5)	~~1~~.111 0000 1000 0000 0000 0000	0x42708000
2^{11} -1	0	10001001 (137-127 = 10)	~~1~~.111 1111 1110 0000 0000 0000	0x44FFE000

Convert the following single-precision floating-point number (given in hexadecimal) to decimal, and enter the answer in the table below:

d) 0x338c0000

After you determine the *S, E,* and *F* fields, compute the decimal value using a calculator.

Hex	***S field***	***E field (binary)***	***Significand (binary)***	***Decimal (using calculator)***
338c0000	0	01100111 (103-127 = -24)	~~1~~.000 1100 0000 0000 0000 0000	6.519e-8