

TSPE Practice Test Solutions: EE 2310, Digital Systems, Test #1

Name: _____ Student Number: _____

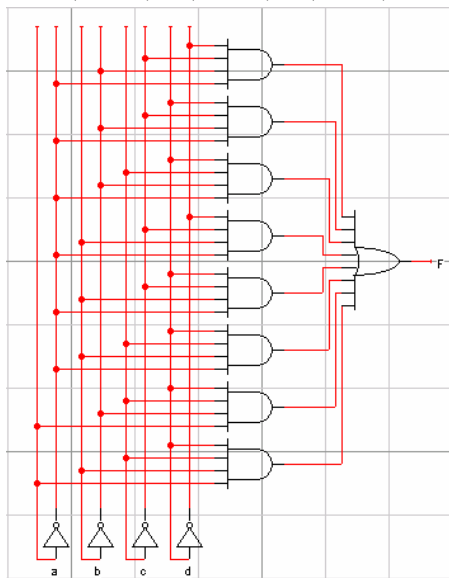
Put answers on the problem statement page, as for homework assignments. Staple work sheets in back if necessary.

1. Find the Boolean expression in SOP form from the truth table. Draw the corresponding circuit. Then, draw a Karnaugh Map and simplify the Boolean Expression. Draw the new circuit, check for hazards, and redesign if necessary.

a	b	c	d	f
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Complete Equation:

$$F = (\bar{a}\bar{b}\bar{c}\bar{d}) + (\bar{a}\bar{b}\bar{c}d) + (\bar{a}\bar{b}cd) + (\bar{a}b\bar{c}\bar{d}) + (\bar{a}b\bar{c}d) + (\bar{a}bcd) + (ab\bar{c}d) + (abcd)$$

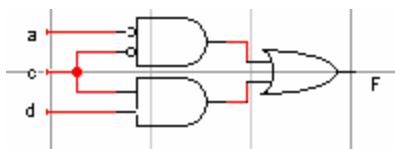


With Hazard

ab \ cd	00	01	11	10
00	1	1	1	0
01	1	1	1	0
11	0	0	1	0
10	0	0	1	0

Simplified Equation:

$$F = (\bar{a}\bar{c}) + (cd)$$

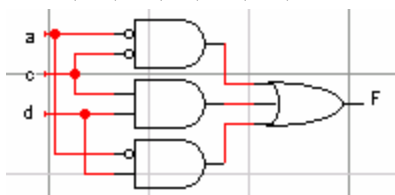


Without Hazard

ab \ cd	00	01	11	10
00	1	1	1	0
01	1	1	1	0
11	0	0	1	0
10	0	0	1	0

Simplified Equation w/ hazard removed:

$$F = (\bar{a}\bar{c}) + (cd) + (\bar{a}d)$$

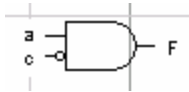


2. The Karnaugh map for a Boolean function is described by: $F = \sum m(8,9,12,13)$

Write the original expression. Use the Karnaugh map to simplify the expression.
Write the simplified expression and draw the circuit.

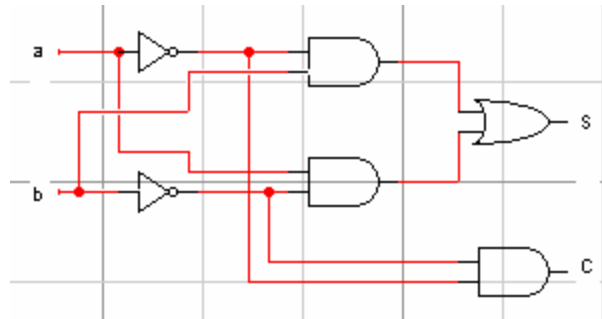
Original: $F = (ab\bar{c}\bar{d}) + (ab\bar{c}d) + (a\bar{b}\bar{c}\bar{d}) + (a\bar{b}\bar{c}d)$

Simplified: $F = (a\bar{c})$

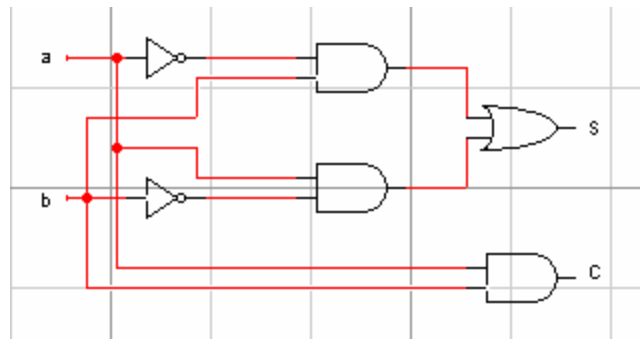


		cd			
	ab	00	01	11	10
	00	0	0	0	0
	01	0	0	0	0
	11	1	1	0	0
	10	1	1	0	0

3. The following half-adder was designed incorrectly. Fix it by redrawing the correct circuit.



Corrected Diagram:



Analysis: The correction is due to the fact that 'a' and 'b' must both be '1' for there to be a carry-out. Thus, 'a' and 'b' must be ANDed together to form the carry-out.

Another possible correction would be to add two more NOT gates to the inputs of the carry-out AND gate. However, this is just redundant as you are just getting $\neg a$ and $\neg b$ going into the gate. This is the same as sending a and b into the gate.

4. Design a simple multiplexer that will output one of four data inputs D0, D1, D2, D3 depending upon a 2-bit address A1, A0 using NOT gates and three input AND gates.

Use the following output table:

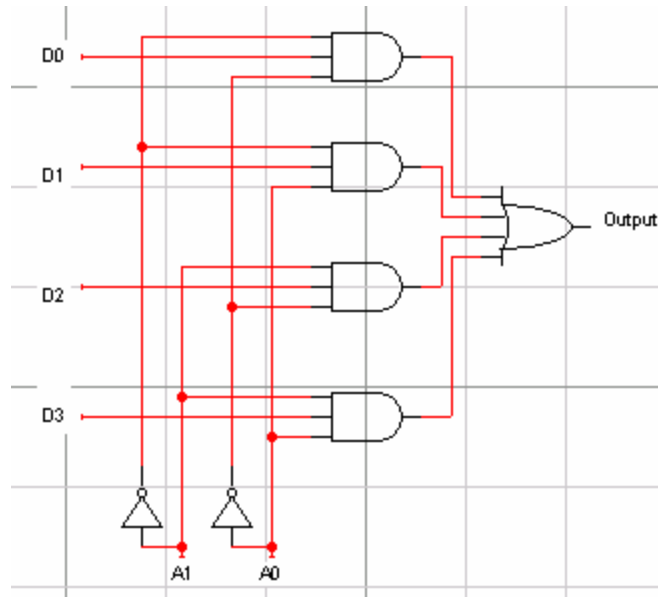
A1	A0	Output
0	0	D0
0	1	D1
1	0	D2
1	1	D3

Analysis:

Think of each 3-input AND gate as a real gate. When two of the inputs turn '1' the gate lets whatever's in the third input through. Therefore, if we want D0 to appear at the output, we need to make sure that whatever gate D0 is attached to is turned "on." I.E. We need to make sure that the other two inputs are both '1'.

Since D0 is supposed to be output when A1 and A0 are both 0 and because the gate needs '1' inputs to open, we will negate both A1 and A0 and connect them to the gate that D0 is connected to. That way, when [0,0] appears in the address lines, D0 is let through and travels to the OR gate. Since all the other gates are "closed" and output '0', D0 appears at the output. I.E. $(D0) + 0 + 0 + 0 = D0$.

The same is true for D1, D2, and D3 except that we need to provide the appropriate negations for each of the gates that they are connected to in order to make sure that the gate, and only that gate, opens when it's address is called.



5. Construct a four bit adder/subtractor that performs twos complement addition or subtraction of positive numbers. Assume that A is always larger than B. The circuit should perform $A+B$ for $op=0$ and $A-B$ for $op=1$. Indicate when an overflow has occurred by setting an output OVFL to one. Use full adder modules. (You don't have to show the inner circuitry)

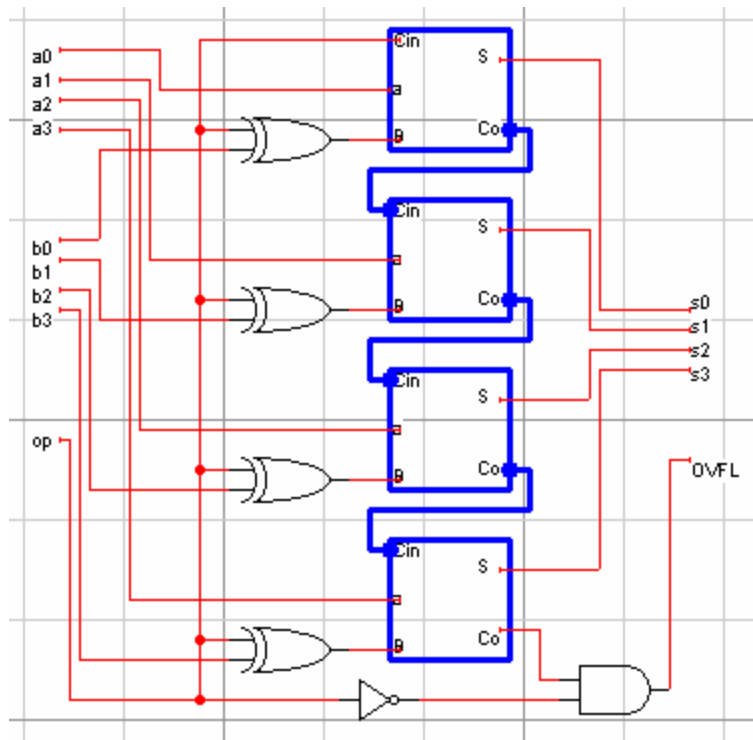
Analysis:

$A+B$ can be performed directly. Each input of A: a_3, a_2, a_1, a_0 can be directly added to each input of B: b_3, b_2, b_1, b_0 with carry-outs going to the carry-in of the next adder module.

$A - B$ is different. $A - B$ is same as $A + (-B)$. Thus, twos complement negation must be performed on B before the result is added to A. All B inputs: b_3, b_2, b_1, b_0 must be negated and a one must be added.

This can be accomplished by allowing op to negate the B input and allowing a '1' to be input into the LSB full-adder's C-in. Since $op=1$ indicates a subtract operation, op can be tied to the LSB full-adder's C-in. This will add a '1' to the entire operation, giving $A + (!B) + 1$ which is the same as $A + (-B)$.

OVFL is detected by whether there is a carry-out of the MSB full adder. Remember that the carry-out of the MSB is ignored for two's complement subtraction. Thus, we can tie OVFL to the carry-out of the MSB through an AND gate that is controlled by op . Whenever op is '1' and subtraction is indicated, OVFL must remain at '0' and, whenever, op is '0' and addition is indicated, OVFL will function normally be tied directly to the carry-out of the MSB.



6. Convert the following decimal numbers to signed binary (2's complement byte form):

-17	-100	21
	1. Convert to binary (disregard sign)	
0001 0001	0110 0100	
	2. Negate each bit (if negative)	
1110 1110	1001 1011	
	3. Add 1 (if negative)	
1110 1111	1001 1100	0001 0101

7. Convert the following 2's complement binary numbers to decimal:

1101 0110	0100 1110	1100 1101
	1. Negate each bit (if negative)	
0010 1001		0011 0010
	2.. Add 1 (if negative)	
0010 1010		0011 0011
	3. Convert to decimal and put sign in front	
-42	78	-51

8. Convert the following positive binary numbers to hexadecimal:

0100 1011	1010 1010	1111 1110
0x 4b	0x aa	0x fe

9. Convert the following hexadecimal numbers to unsigned binary (byte form):

0x f3	0x 8a	0x ae
1111 0011	1000 1010	1010 1110

10. Convert the following positive binary numbers to decimals.

110.11011	101.00101	0001 1.101
110.11011 => 4(1)+2(1)+1(0)+.5(1)+.25(1)+.125(0)+.0625(1)+.03125(1) = 6.84375		
101.00101 => 4(1)+2(0)+1(1)+.5(0)+.25(0)+.125(1)+.0625(0)+.03125(1) = 5.15625		
0001 1.101 => 16(0)+8(0)+4(0)+2(1)+1(1)+.5(1)+.25(0)+.125(1) = 3.625		

11. Convert the following decimals to binary. (Use 2x the number of binary places after the binary point as there are decimal places after the decimal point if non-exact)

12.3	101.125	3.12
12.3 => 16(0) + 8(1) + 4(1) + 2(0) + 1(0) + .5(0) + .25(1) + .125(0) =>		
0011 00.01 = 12.25		
101.125 => 64(1)+32(1)+16(0)+8(0)+4(1)+2(0)+1(1)+.5(0)+.25(0)+.125(1) =>		
0011 0010 1.001 = 101.125		
3.120 => 4(0)+2(1)+1(1)+.5(0)+.25(0)+.125(0)+.0625(1)+.03125(1)+.015625(1) =>		
11.00 0111 = 3.109375		

12. Simplify the following expression: $F = \overline{(\overline{a}b)} \bullet \overline{(\overline{a}\overline{b})}$

1) Working from outside-in:

Use DeMorgan's Law: $F = \overline{(\overline{a}b)} + \overline{(\overline{a}\overline{b})} = (\overline{a}b) + (\overline{a}\overline{b})$

Pull out (de-distribute): $F = \overline{a}(b + \overline{b})$

Since $b + \overline{b}$ always = 1: $F = \overline{a}$

2) Working from inside-out:

Use DeMorgan's Law: $F = \overline{(\overline{a} + \overline{b})} \bullet \overline{(\overline{a} + b)} = \overline{(a + b)} \bullet (a + b)$

Use DeMorgan's Law again: $F = \overline{(a + b)} + \overline{(a + b)}$

Use DeMorgan's Law again: $F = (\overline{a}\overline{b}) + (\overline{a}\overline{b}) = (\overline{a}b) + (\overline{a}\overline{b})$

Pull out (de-distribute): $F = \overline{a}(b + \overline{b})$

Since $b + \overline{b}$ always = 1: $F = \overline{a}$

13. Simplify the following expression: $F = (ab\overline{c}\overline{d}) + (ab\overline{c}d) + (\overline{a}\overline{b}\overline{c}\overline{d}) + (\overline{a}\overline{b}\overline{c}d)$

Pull out common abc's: $F = ab\overline{c}(\overline{d} + d) + \overline{a}\overline{b}\overline{c}(\overline{d} + d)$

Since $d + \overline{d}$ always = 1: $F = ab\overline{c} + \overline{a}\overline{b}\overline{c}$

Pull out common ac's: $F = a\overline{c}(b + \overline{b})$

Since $b + \overline{b}$ always = 1: $F = a\overline{c}$ (This checks with problem #2)