Solved Questions for Midterm Exam - Digital Design Fundamentals | CSE 120

Computer Science

Arizona State University (ASU) - Tempe

8 pag.

CSE/EEE120 Fall 2011
Midterm

Name:_____ **ASU ID:**_____

You have 75 minutes to complete and turn in this quiz. Each question is worth 10 points. Please read questions carefully and provide the answers in the form requested. SHOW YOUR WORK. Good luck!

---

1. Given the following function,

$$g = xy' + yz' + x'yz$$

(a) Write out the truth table for ***g***.

	x	y	z	g
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

(b) Write a SOP expression for ***g*** in *short-hand (numerical) notation.*

g = Σm(2,3,4,5,6)

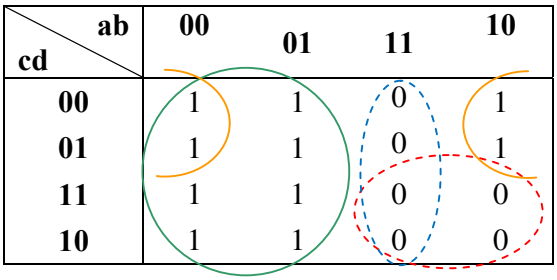
(c) Derive a canonical SOP expression for ***g'*** .

g' = x'y'z' + x'y'z + xyz

2. You are supposed to design a "blanking" detection circuit for a display. The display should show the decimal numbers 0_{10} ... 9_{10} , which are represented in a 4-bit binary word. Any number larger than 9_{10} should render the output function ***f*** false. Using the 4-input truth table shown on the right,

   (a) Draw, label, and fill-in the correct Karnaugh Map (K-map) for the function ***f***, and write the minimum SOP expression for ***f***.

	a	b	c	d	f
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0



f = a' + b'c'

   (b) Extract the minimum POS form out of the K-map that you prepared in (a).

The dashed circles represent the essential prime implicants for the POS form, resulting in the equation

f = (a' + b') (a' + c')

3. Given the following function in canonical SOP form,

$$h = ab'cd' + ab'cd + abc'd' + abc'd + abcd' + abcd$$

Use Boolean algebra to simplify ***f*** down to a minimum SOP expression that contains only **2 terms and 4 literals**. Document your steps! You will not get full credit if intermediate steps are missing!

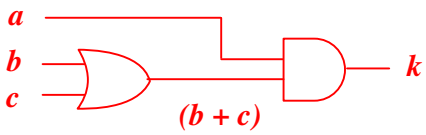
$h = ab'cd' + ab'cd + abc'd' + abc'd + abcd' + abcd$	
$= a(b'cd' + b'cd + bc'd' + bc'd + bcd' + bcd)$	distributive
$= a(b'c(d' + d) + bc'(d' + d) + bc(d'+d))$	distributive
$= a(b'c + bc' + bc)$	identity
$= a(b'c + b(c' + c))$	distributive
$= a(b'c + b)$	identity
$= a(c + b)$	simplification
$= ac + ab$	distributive
$= ab + ac$	commutative

N. B.: x + x'y = x + y (simplification rule, P10a in Marcovitz)

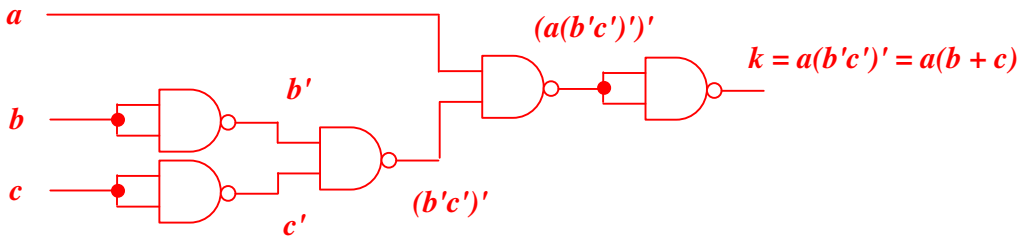
a + 0 = a	a * 1 = a	Identity
a + a' = 1	a * a' = 0	Complement
a + a = a	a * a = a	Idempotency
a(b + c) = ab + ac	a + bc = (a + b)(a + c)	Distributive
a + a'b = a + b	a(a' + b) = ab	Simplification
(a + b)' = a'b'	(ab)' = a' + b'	DeMorgan

4. Using the simplified function ***k = a(b + c),***

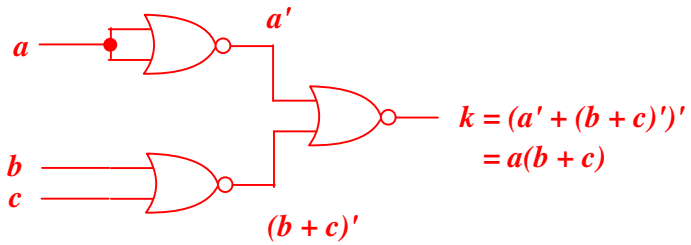
(a) Draw the logic circuit using AND/OR/NOT gates. Only uncomplemented inputs are available.



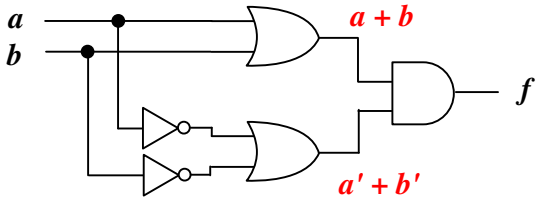
(b) Using gate equivalency implement ***k*** using only NAND gates (do not use Boolean Algebra!). Only uncomplemented inputs are available.

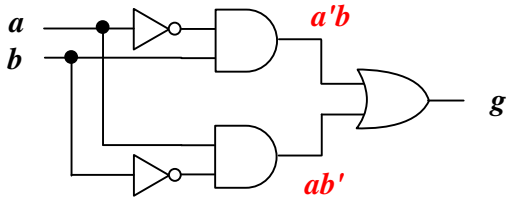


(c) Using gate equivalency implement ***k*** using only NOR gates (do not use Boolean Algebra!). Only uncomplemented inputs are available.



5. Given the following two circuits





(a) Label the internal signals and write down the logic equation for each of them.

f = (a + b) (a' + b') , **g = a'b + ab'**

(b) Show using a truth table which shows every term in each of the functions that both circuits perform an identical logic operation.

	a	b	a + b	a' + b'	f	a'b	ab'	g
0	0	0	0	1	0	0	0	0
1	0	1	1	1	1	1	0	1
2	1	0	1	1	1	0	1	1
3	1	1	1	0	0	0	0	0

6. Perform the following conversions (show all of your work):

(a) Convert $\mathbf{110110_2}$ to a decimal number (assuming unsigned representation).

$\mathbf{110110_2 = 2^5 + 2^4 + 2^2 + 2^1 = 32 + 16 + 4 = 54}$

(b) Convert $\mathbf{73_{10}}$ to an unsigned 7-bit binary number.

$\mathbf{73_{10} = 64 + 8 + 1 = 2^6 + 2^3 + 2^0 = 1001001_2}$

(c) Convert $\mathbf{0011100100110110000_2}$ to a hexadecimal number.

1110010011011000 = 1110 0100 1101 1000 = $\mathbf{E4D8_{16}}$

7. Answer the following questions (explain your answers):

   (a) What is the maximum number that can be represented using a 5-bit, two's complement format?

   In 5-bit two's complement form, the smallest number that we can represent is 01111 = +**15**.

   (b) What is the minimum number that can be represented using a 4-bit, two's complement format?

   In 4-bit two's complement form, the smallest number that we can represent is 1000 = **-8**.

   (c) How many bits would it take to represent **(-45)**$_{10}$ using an n-bit, two's complement format?

   For 7 bits, the range is [-64,+63], so we need **7** bits to represent -45.

8. Compute the sum of the following pairs of two's complement, signed 6-bit binary numbers. **In parallel, show the operands and the answer in decimal**. Indicate <u>if or if not</u> an overflow occurs.

   (a)

```
    101110
  + 110000

   100000
    101110     - 18
+   110000     - 16
   0011110     - 34  < -32
```

2's complement of 101110:

```
   000001
    010001
+   000001
    010010    = +18
```

2's complement of 110000:

```
   001111
    001111
+   000001
    010000    = +16
```

The carry-in (green) and the carry-out (blue) of the left most column are not equal, so there is overflow. The expected result is -34 but because the range is [-32,31], the answer is +30.

   (b)

```
    010000
  + 110001

   000000
    001110      16
+   110001     - 15
   0111111      1
```

2's complement of 110001:

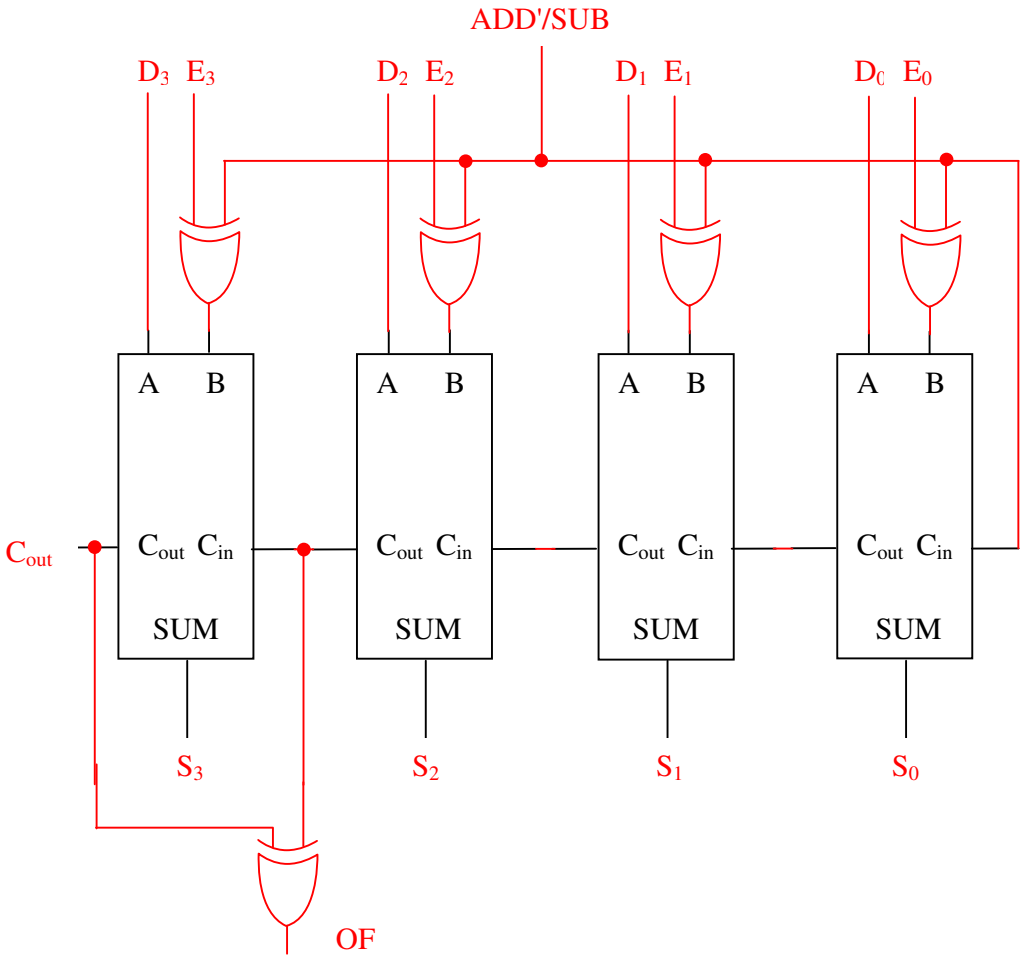
```
   000000
    001110
+   000001
    001111    = +15
```

The carry-in (green) and the carry-out (blue) of the left most column are equal, so there is no overflow.

9. Answer the following questions about adder circuits:
(a) Explain the primary difference between a half adder circuit and a full adder circuit.

A full adder has a carry-in input, while a half adder does not.

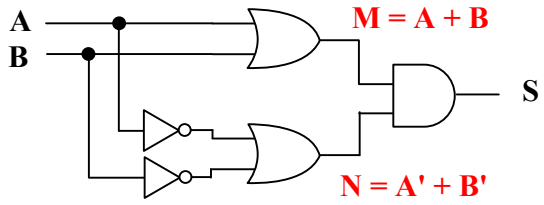
(b) Complete the partial circuit shown below in order to properly implement a 4-bit adder/subtracter with overflow detect. Make sure to label all lines correctly. Assume you are adding/subtracting $E_3E_2E_1E_0$ to/from $D_3D_2D_1D_0$and the result is $S_3S_2S_1S_0$.



10. In Hardware lab 1 you built half and full adder circuits. During the Debugging step, somebody messed up your product-of-sums circuit for the SUM generation of the half adder. Using a logic probe, you obtain the following truth table (M and N are the inputs to the AND gate generating the SUM) :

A	B	M	N	SUM	CRY
0	0	1	1	1	0
0	1	1	1	1	0
1	0	1	1	1	0
1	1	1	0	0	1

(a) Draw the gate-level schematics of the POS implementation of the SUM function.



(b) Indicate what could be wrong with the circuit, i.e. which of the connections to the gate(s) are faulty, resulting in the truth table values shown above.

The SUM function is displaying an erroneous '1' for **A** = **B** = '0'. Checking the inputs **M** and **N** it turns out that **N** is behaving correctly, based on the logic equation **N = A' + B'**, while **M** is permanently stuck at '1'. **Thus, there is something wrong with the top OR gate feeding the AND gate.** Most probably, **one of the inputs is permanently wired to '1'** and not to the respective input. Since we used a logic probe, we can be sure that the output of the OR gate is connected to the AND gate, since otherwise the probe would have indicated an open circuit. An open **M** input to the TTL AND gate would have the same effect on the SUM output as described.