Homework #1 (v180404)

(Deadline: 11:59PM PDT, April 20, 2018)

Name (Last, First): Christina Oliveira

Student Id #: 204-803-448

INSTRUCTIONS

This homework is to be done individually. You may use any tools or refer to published papers or books, but may not seek help from any other person or consult solutions to prior exams or homeworks from this or other courses (including those outside UCLA). You're allowed to make use of tools such as Logisim, WolframAlpha (which has terrific support for boolean logic) etc.

You must submit all sheets in this file based on the procedure below. Because of the grading methodology, it is much easier if you print the document and answer your questions in the space provided in this problem set. It can be even easier if you answer in electronic form and then download the PDF. Answers written on sheets other that the provided space will not be looked at or graded. Please write clearly and neatly - if we cannot easily decipher what you have written, you will get zero credit

SUBMISSION PROCEDURE: You need to submit your solution online at Gradescope (https://gradescope.com/). Please see the following guide from Gradescope for submitting homework. You'd need to upload a PDF and mark where each question is answered.
http://gradescope-static-assets.s3-us-west-2.amazonaws.com/help/submitting_hw_guide.pdf

Problem #1

A 6-bit decoder takes in 6 address bits, *addr[5:0]*, and for all possible binary combinations of the address, only assert one of the outputs, *dec[n-1:0]*, HIGH where *n* is the number of outputs. We discuss this as a building block later as a common block for a memory. Note that in a memory often 10-20 bits of address are decoded, so this is a greatly simplified design problem.

(a) How many combinations of the 6 address bits are there (in other words, how many outputs does this logic have)?

Now, a pre-decoder is a logic block that processes only a subset of the input address bits. Let the 0th pre-decode output, *pdec[0]*, be asserted HIGH (1'b1) when *addr[5:2]* are all LOWs (4'b0000) and *addr[1:0]* are don't cares.

(b) Write the output, *pdec[0]*, as a Boolean function of *addr[5:0]*, in fully-disjunctive normal form.

(c) Write the output, *pdec[0]*, as a simplified Boolean function of inputs.

(d) In some decoder implementation, the output, *pdec[0]* for instance, is asserted LOW (1'b0). Write a simplified Boolean expression of the output, *pdec[0]*, in Normal Form.

(e) Implement the function in (c) using only 2-input NAND gates.

(f) Implement the function in (c) using 2-input NAND and NOR gates.

Answer the question for all parts in the space below.

f e d c b a = High

(0 0 0 0 X X)

(∧) ∨ (∧ ∧) fully disjunctive

a) 2^6 = 64 outputs ???

b) $pdec[0] = (\bar{f} \wedge \bar{e} \wedge \bar{d} \wedge \bar{c} \wedge b \wedge a) \vee (\bar{f} \wedge \bar{e} \wedge \bar{d} \wedge \bar{c} \wedge \bar{b} \wedge a) \vee (\bar{f} \wedge \bar{e} \wedge \bar{d} \wedge \bar{c} \wedge b \wedge \bar{a}) \vee (\bar{f} \wedge \bar{e} \wedge \bar{d} \wedge \bar{c} \wedge \bar{b} \wedge \bar{a})$

fully-disjunctive normal form

c) $pdec[0] = (\bar{f} \wedge \bar{e} \wedge \bar{d} \wedge \bar{c}$)

simplified form

(IMPORTANT: ~~Keep this page~~ in submission even if left unused)

Assert Low

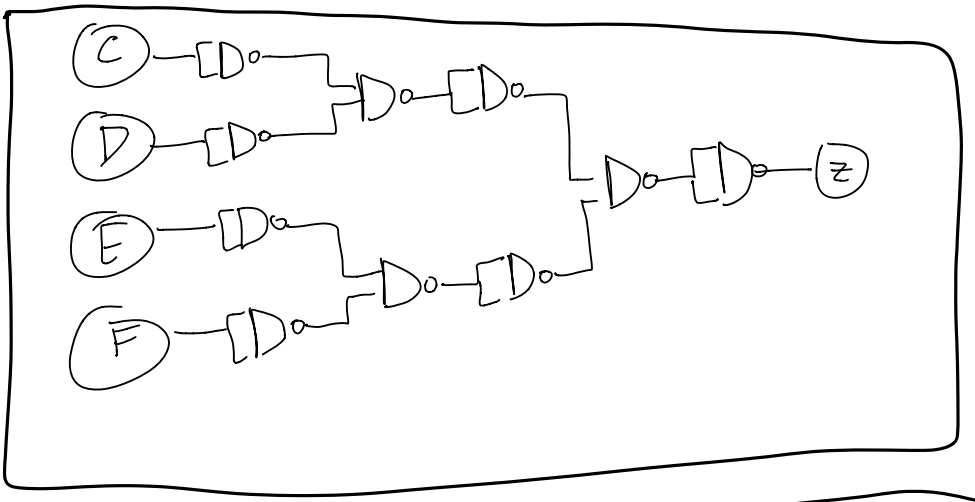
d) $prdec[0] = (f \wedge e \wedge d \wedge c \wedge b \wedge \bar{a}) \vee$
$(f \wedge e \wedge d \wedge c \wedge \bar{b} \wedge a) \vee$
$(f \wedge e \wedge d \wedge \bar{c} \wedge b \wedge a) \vee$
$(f \wedge e \wedge \bar{d} \wedge c \wedge b \wedge a)$
.....
and so on → (60 total terms)

fully-disjunctive normal form

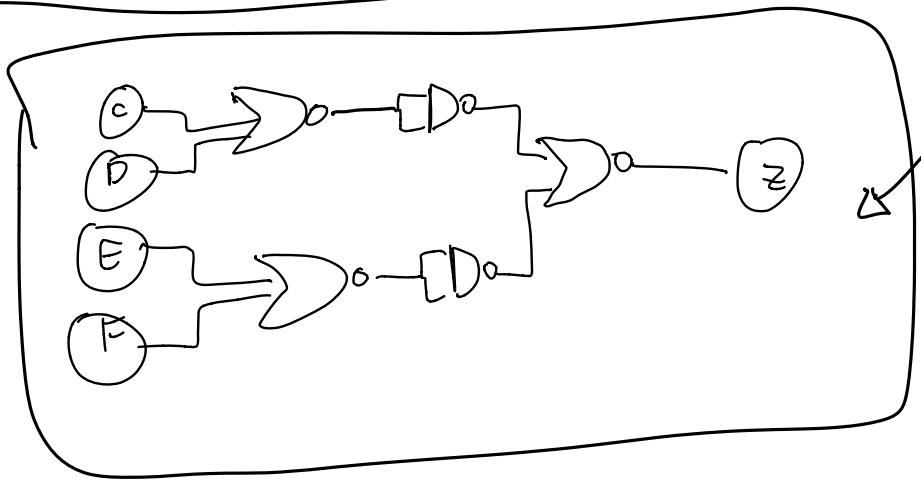
all poss comb. besides the 4 described in b)

e) WANT THIS:
$prdec[0] = (\bar{f} \wedge \bar{e} \wedge \bar{d} \wedge \bar{c})$ using only 2 input NAND gates

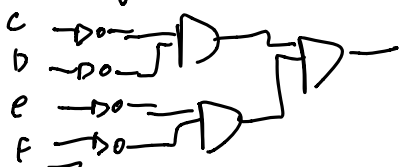
$\overline{C \wedge D \wedge E \wedge F}$



f)



from logisim:



Problem #2

ASCII (American Standard Code for Information Interchange) is a standard way to represent characters in binary. It takes 7-bits to represent an assortment of different characters including the alphabet (both lower and upper case), numbers from 0-9, and various symbols. In this problem, the input, *ascii[6:0]*, can only be one of the alphanumeric characters (no symbols). Design the logic for the three outputs, *alphaCap*, *alphaNoCap*, and *numbers*.

(a) Write the truth table for each of the three outputs. Since there are many don't care conditions, condense the table as needed.

(b) Based on your result in (a), write the simplified Boolean expression as a function of *ascii[6:0]*. Feel free to use any logic simplification methods such as K-map.

a)

numbers

6 5 4 3 2 1 0	output
0 1 d d d d	1
else	0

where d = DON'T CARE

alphaNoCap

6 5 4 3 2 1 0	output
1 1 d d d d	1
else	0

alphaCap

6 5 4 3 2 1 0	output
1 0 d d d d	1
else	0

(IMPORTANT: Keep this page in submission even if left unused)

b)

Boolean for numbers:

$\overline{ascii[6]} \wedge ascii[5]$

Boolean for alphaNoCap:

$ascii[6] \wedge ascii[5]$

Boolean for alphaCap:

$ascii[6] \wedge \overline{ascii[5]}$

Problem #3

Consider the two Boolean functions below. Try to implement the logic using the fewest number of n-input NAND gates n-input NOR. Only true inputs (a,b,c,d) are used. You may need to simplify the logic.

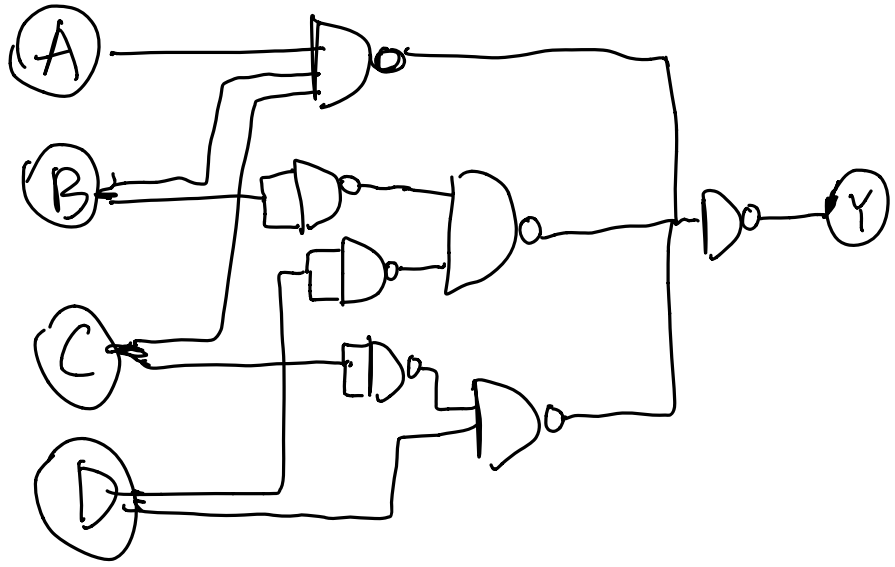
(a) $Y = \neg\big((\neg a \wedge c) \wedge (d \vee b)\big) \wedge (\neg b \vee c \vee d) \wedge \big(\neg\big((a \wedge \neg b) \vee (\neg a \wedge b)\big) \vee \neg c \vee \neg d\big)$

(b) $Z = \neg\big(\neg\big((a \wedge b) \vee \neg c\big) \vee (d \vee e) \vee \neg a\big)$

Answer the question for all parts in the space below.

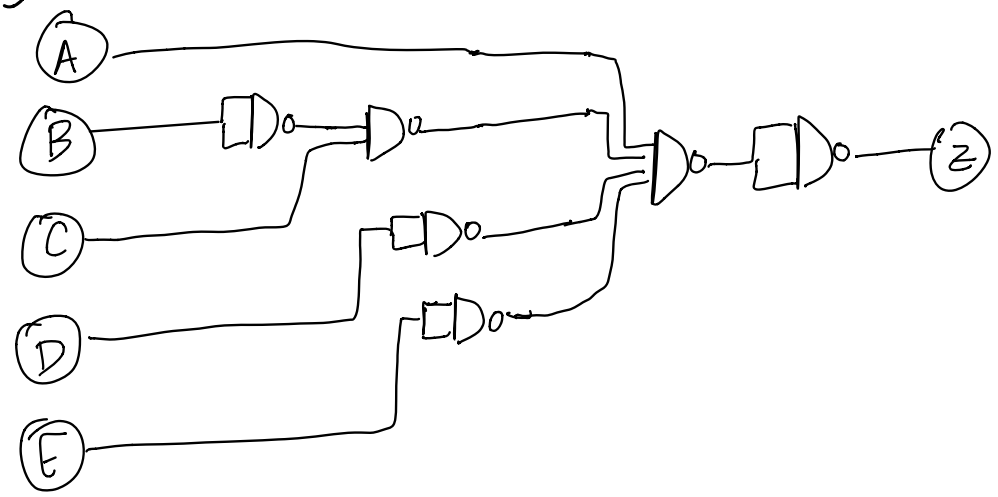
a) using NAND:

$(A \bar{\wedge} B \bar{\wedge} C) \bar{\wedge} (\bar{B} \bar{\wedge} \bar{D}) \bar{\wedge} (\bar{C} \bar{\wedge} D)$



(IMPORTANT: Keep this page in submission even if left unused)

b) $Z = \neg\big(\neg\big((a \wedge b) \vee \neg c\big) \vee (d \vee e) \vee \neg a\big)$



Problem #4

Time is often represented with HOUR:MINUTE:SECOND {AM,PM}. There are many ways of representing this information. Let's consider two. First is to use a separate binary number for each of the four fields. A second is to use a single binary number representing the number of seconds from midnight.

(a) How many bits does one need to represent the first method? Show the result for the start time of the lecture.

(b) How many bits does one need to represent the second method? Show the result for the end time of the lecture.

(c) To calculate time difference most easily, what would be the better choice? Explain your answer.

(d) To display the time on a clock, what would be the better choice? Explain your answer.

(e) An additional field is actually needed to represent time zone. Time zone can be expressed as deviation from GMT and can be +14 to -11:45. What is the minimum number of bits one needs to indicate the shift due to time zone?

a)

	HOUR	MINUTE	SECOND	AM/PM
# of items	12	59	60	2
2^n	2^4	2^6	2^6	2^1
	=16	=64	=64	=2

and 2^n = # of items to represent where n is the number of bits so:

4+6+6+1 = **17 bits needed**

start time:

2:00:00 PM

00010:000000:000000:1

b) seconds from midnight – 86400 seconds in a day so:

# of items	86400
2^n	2^{17} = 131,072

17 = **17 bits needed**

(IMPORTANT: Keep this page in submission even if left unused)

b continued) ending time: 3:50:00 which is 57000 seconds from midnight

so: 00|01|11|0|10|000

c) the better choice is the second method, because you can just subtract the two binary numbers.

d) the better choice would be the first method because you would have to interpret the bits rather that just translate them, unless your clock simply displayed minutes from midnight.

e) if we assume 15-minute increments then there are 104 possible "steps" between -11:45 and 14, so to represent 104 possibilities we need 7 bits because $2^7 = 128$.

(IMPORTANT: Keep this page in submission even if left unused)

Problem #5

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	X
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	0
1	1	0	0	1
1	1	0	1	X
1	1	1	0	1
1	1	1	1	1

(a) Draw K-Map of the function corresponding to the truth table shown above. Identify on the K-Map the prime implicants of the function and identify which of the prime implicants (if any) are essential.

(b) Write the minimum cover as a sum-of-product Boolean function using the prime implicants found in (a)

(c) Draw the K-Map of the complement function, Y'. Identify on the K-Map the prime implicants of the function and identify which of the prime implicants (if any) are essential.

(d) Write the minimum cover as a sum-of-product Boolean function using the prime implicants found in (c).

(e) Write the complement function, Y', as a minimal product-of-sum.

(f) Draw the truth table of the dual of the function Y, Y^D.

(IMPORTANT: Keep this page in submission even if left unused)

a)

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

prime implicants: 5

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

essential prime implicants: 4

b) MIN COVER: Boolean Function:

$$(\bar{b} \wedge \bar{c}) \vee (a \wedge b) \vee (\bar{c} \wedge d) \vee (b \wedge c \wedge \bar{d})$$

$X00X + 11XX + XX01 + X110$

ab cd

c)

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

prime implicants =
ess. prime implicants = 3

all prime implicants here are essential.

(IMPORTANT: Keep this page in submission even if left unused)

d) sum-of-products Boolean function

$$(\bar{a} \wedge b \wedge \bar{c} \wedge \bar{d}) \vee (\bar{a} \wedge c \wedge d) \vee (\bar{b} \wedge c)$$

0100 V 0X11 V X01X

e) write this as product-of sum

$$(b \vee c) \wedge (\bar{a} \vee \bar{b}) \wedge (c \vee \bar{d}) \wedge (\bar{b} \vee \bar{c} \vee d)$$

f) draw truth table of DUAL of Y

$$Y = (\bar{b} \wedge \bar{c}) \vee (a \wedge b) \vee (\bar{c} \wedge d) \vee (b \wedge c \wedge \bar{d})$$

$$Y_D = (\bar{b} \vee \bar{c}) \wedge (a \vee b) \wedge (\bar{c} \vee d) \wedge (b \vee c \vee \bar{d})$$

← just need to write form on next page

(see next page)

(IMPORTANT: Keep this page in submission even if left unused)

$f^D(\bar{a}, \bar{b}, \bar{c}, \bar{d}) = \neg f(a, b, c, d)$

$f(0,0,0,0) = 0$

$f^D(1,1,1,1)$

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	X
0	0	1	1	0
0	1	0	0	1
0	1	0	1	X
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	X
1	1	1	1	0