Prof. C.K. Yang

Midterm Exam

Name (Last, First):

Student Id #:

Do not start working until instructed to do so.

1. *You must answer in the **space provided** for answers after every question. We will ignore answers written anywhere else in the booklet. **All pages in this booklet must be accounted** for otherwise it will not be graded.*
2. *You are permitted 1 page of notes 8.5x11 (front and back).*
3. *You may not use any electronic device.*

Following table to be filled by course staff only

	Maximum Score	Your Score
Question 1		
Question 2		
Question 3		
Question 4		
TOTAL	100	

Question #1

Consider the following Karnaugh Map for the Boolean function, Y. A blank truth table is provided for your convenience.

CD \ AB	"00"	"01"	"11"	"10"
"00"	0	0	1	1
"01"	1	1	X	X
"11"	0	0	1	1
"10"	1	1	X	0

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

(a) Circle the prime implicants on the map.
How many prime implicants are there? 5

(b) Write the Boolean (sum-of-product) expression of the *essential* prime implicants of (b) (if any).

EssentialPrimeImplicants = $(\bar{C}D) \vee (A\bar{C}) \vee (AD) \vee (\bar{A}C\bar{D})$

(c) Complete the following statement

$$\neg Y = \prod M(\quad)$$

This seems like the cover of 0's

How do you get inverse, product of sums

A B C D
0 X 0 0 $\bar{A}\bar{C}\bar{D} \vee \bar{A}CD \vee A\bar{C}\bar{D}$
0 X 1 1
1 X 1 0

(d) Express as a minimal sum of product, $\neg Y$.
The K-map is provided for your convenience.

AB

CD	"00"	"01"	"11"	"10"
"00"	0	0	1	1
"01"	1	1	X	X
"11"	0	0	1	1
"10"	1	1	X	0

C X 00 $\bar{A}C\bar{D}$
0X 11 $\bar{A}CD$
1X 10 $A C\bar{D}$

$\neg Y =$ $(\bar{A}C\bar{D}) \vee (\bar{A}CD) \vee (AC\bar{D})$

(e) Is the dual of this function itself? Circle one: Yes No
Justify your answer:

Is it a dual if there are don't care values?

If it weren't for don't care values it would obviously be a dual. It still is however, because you would get the identical logic Y, Y^D if you wrote a boolean expression and flipped $\vee \Leftrightarrow \wedge$, for don't cares we can assume they are any value, because it doesn't matter. Yes

(f) If the logic is implemented using its minimal sum-of-product expression, is there a risk of a static hazard?
Circle one: Yes No
Under what input conditions?

What product term(s) would you add to remove the risk? _____

What is a static hazard?

Question #2

(a) Rewrite the following Boolean equation as a sum-of-products or product-of-sum. Hint: apply DeMorgan's theorem.

$$W = \neg\Big(\neg(c \vee \neg(a \wedge \neg b)) \vee (d \wedge \neg e)\Big)$$

Answer:

$= (c \vee \neg(a \wedge \neg b)) \wedge (\overline{d \wedge \neg e})$

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

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

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

$((c + (ab')')' + (de)')'$

$= (C + (ab')')(de)'$

$= (C + (ab')')(d' + e)$

$(c + (a' + b))(d' + e)$

$(C + a' + b)(d' + e)$

SOP:

$cd' + a'd' + bd' + ce + a'e + be$

(b) Rewrite the following Boolean equation as a sum-of-products or product-of-sum.

$$X = (c \vee (a \wedge b)) \vee (d \wedge e)$$

Answer: Find logical equivalent of $c \vee (a \wedge b)$ in SOP form

$= (C \wedge (a \wedge b)) \vee (C \wedge \overline{(a \wedge b)}) \vee (\bar{C} \wedge (a \wedge b))$

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

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

$X =$ $(c \wedge b) \vee (c\bar{a}) \vee (c\bar{b}) \vee (\bar{c} \wedge b) \vee (de)$

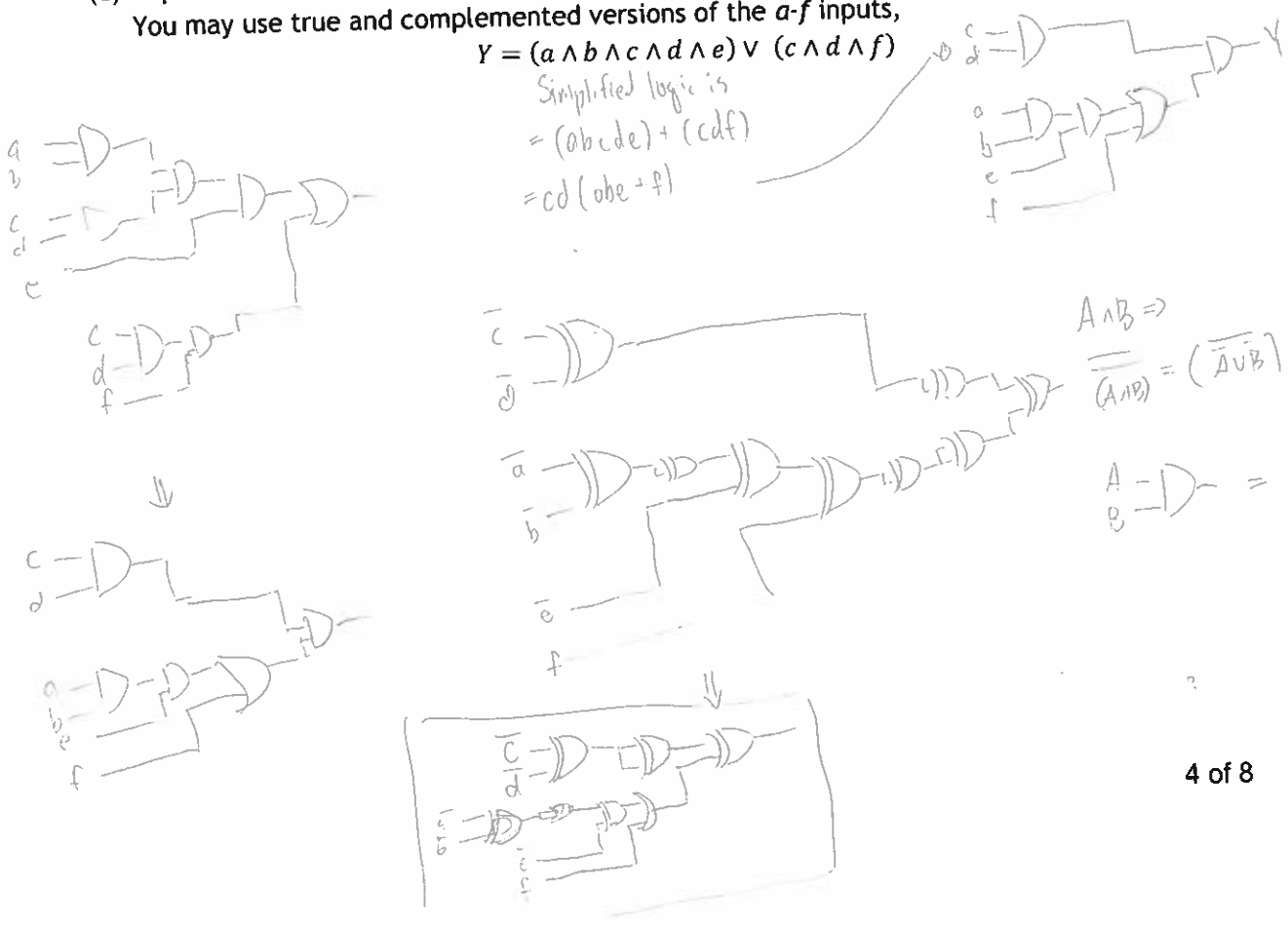
(c) Implement the following function using NOR gates only and the fewest NOR gates possible. You may use true and complemented versions of the *a-f* inputs,

$$Y = (a \wedge b \wedge c \wedge d \wedge e) \vee (c \wedge d \wedge f)$$

Simplified logic is

$= (abcde) + (cdf)$

$= cd(abe + f)$



Question #3

The following 12-bit word can be used to represent different numbers depending on the encoding

12b'1011_1101_0110

(a) If the word is 2's complement, what is the corresponding integer? -1066

(b) If we convert the word into base-4, what is the represented number? 2331 12

(c) If we take answer in (b), write the 3's complement of the base-4 number. 3100221

(d) If the word is unsigned fixed point 6.6, what is the corresponding number? 47.6875

What is the absolute accuracy of this representation? .015625

(e) What is this word in Hexadecimal? BD5

(f) If the word is 6E5 floating point number (IEEE format S+EEEEE+MMMMMM),

What is the bias? 2

What is the corresponding real number? $-22 * 10^{13}$ $(-1)^S (1.M) \cdot 2^{(E-bias)}$

(g) In base-20 system and using 3 "vigits" (base-20 digits) in 20's complement:

How would one represent a base-10 integer -1616? _____

What is the most positive value in base-10 integer that can be represented? 3999

Question #4

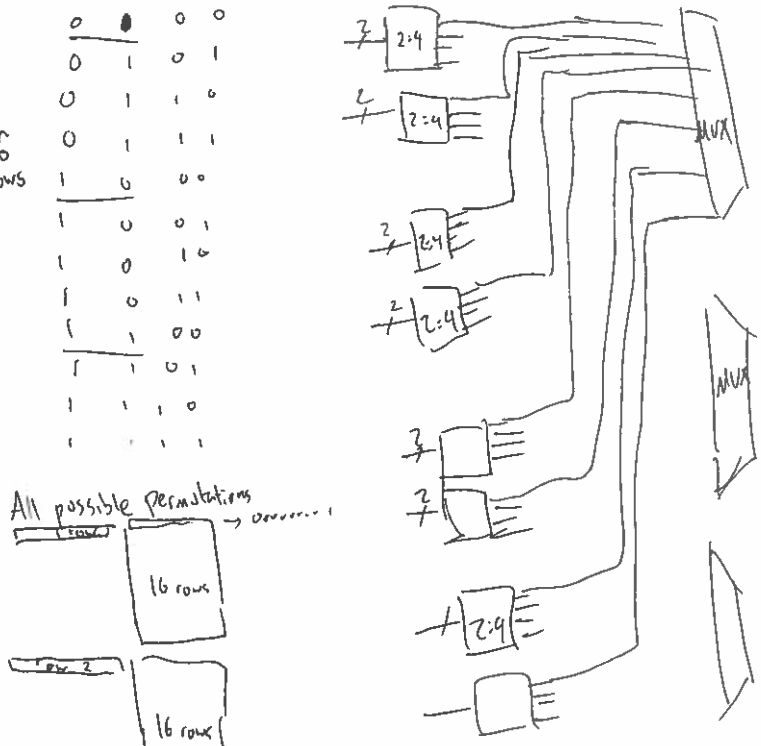
(a) Two 4-bit binary inputs, *b1[3:0]* and *b2[3:0]*, are ~~needs~~ needed to be decoded into a single 2-hot output vector, *twoh[15:0]*. This 2-hot vector has up to two positions of 1's and the remainder are 0's. If b1 is the same as b2, then only a single 1 is asserted. Design this logic. You may use 2:4 Decoders as building blocks and any number of AND, OR, or INV gates. Try to minimize the amount you use. You may describe connections to avoid an overly complex drawing of every signal and line.

00	0004	0001
01	0010	0010
10	0100	0100
11	1000	1000

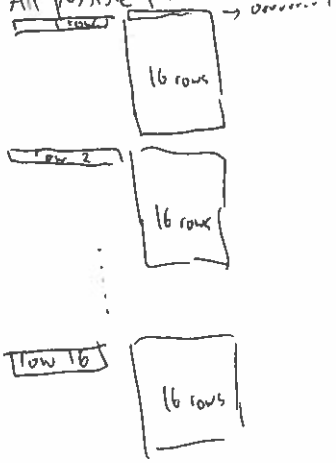
b1[3] b1[2] b1[1] b1[0] b2[3] [2] [1] [0]

~~For each decoder there~~ are 64

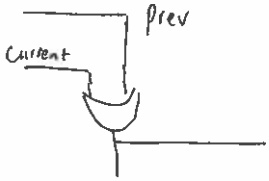
16 rows



All possible permutations → .....



(b) Next, you are tasked to code the *twoh[15:0]* signals into a vector that indicates a range, similar to a thermometer code. The 16-bit input, *twoh[15:0]*, produces 16-bit outputs, *rangeh[15:0]*, whereby the positions in between the 2 asserted signals of the *twoh[15:0]* are all 1's. For instance, with b1[3:0]=4'b0110 and b2[3:0]=4'b0010, rangeh[15:0]=16b'0000_0000_0011_1110. You may use any building block we have covered in lecture as well as any number of AND, OR, or INV gates. Again you can describe connections to avoid drawing every signal and line.



if previous is 0
and current is 1, propogate.

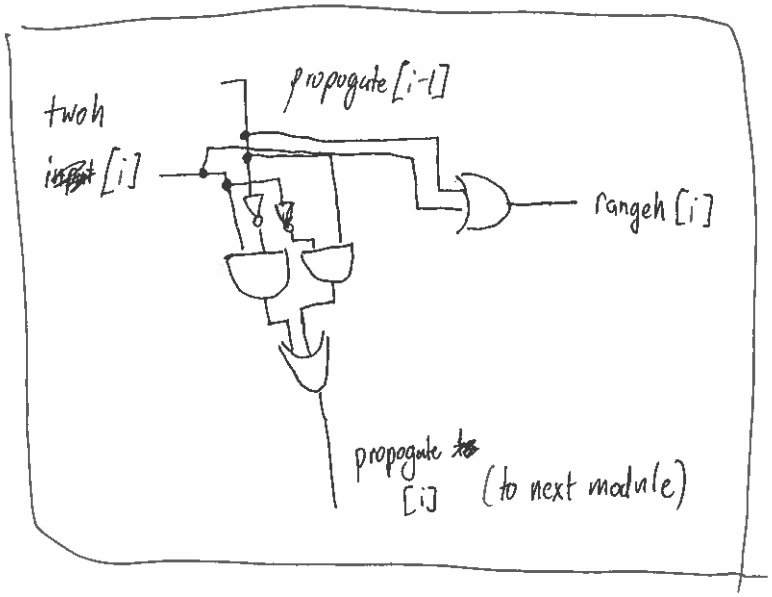
if previous is 1
and current is 1, don't propogate

if previous is 1
and current is 0, propagate

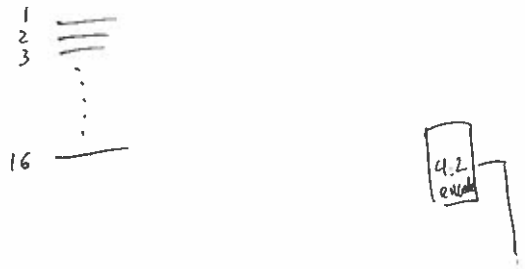
previous	current	Propogate	output
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1

$prop = \bar{p}C \vee p\bar{C}$

output



(c) Finally, encode inverse the function of (a) by accepting the *twoh[15:0]* signal as input and output 2 binary values *bo1[3:0]* and *bo2[3:0]* to indicate the binary position of the two hot bits. You may use any building block we have covered in lecture as well as any number of AND, OR, or INV gates. Again you can describe connections to avoid drawing every signal and line.



Here, you need to have encoders and probably a Mux to