

CSE 140 Homework Three

November 24, 2014

*Only **Problem Set Part B** will be graded. Turn in only **Problem Set Part B** which will be due on December 3, 2014 (Wednesday) at 4:30pm.*

1 Problem Set Part A

- Roth&Kinney, 6th Ed: 6.6
- Roth&Kinney, 6th Ed: 6.9
- Roth&Kinney, 6th Ed: 6.11
- Roth&Kinney, 6th Ed: 6.16
- Roth&Kinney, 6th Ed: 6.17
- Roth&Kinney, 6th Ed: 6.24
- Roth&Kinney, 6th Ed: 7.1
- Roth&Kinney, 6th Ed: 7.4
- Roth&Kinney, 6th Ed: 7.5
- Roth&Kinney, 6th Ed: 7.6
- Roth&Kinney, 6th Ed: 7.8
- Roth&Kinney, 6th Ed: 7.9
- Roth&Kinney, 6th Ed: 7.10
- Roth&Kinney, 6th Ed: 7.11
- Roth&Kinney, 6th Ed: 7.14
- Roth&Kinney, 6th Ed: 7.26
- Roth&Kinney, 6th Ed: 7.27
- Roth&Kinney, 6th Ed: 7.28
- Roth&Kinney, 6th Ed: 7.38
- Roth&Kinney, 6th Ed: 7.39
- Roth&Kinney, 6th Ed: 8.2

- Roth&Kinney, 6th Ed: 8.6
- Roth&Kinney, 6th Ed: 8.7
- Roth&Kinney, 6th Ed: 8.8
- Roth&Kinney, 6th Ed: 13.3(a)(b)
- Roth&Kinney, 6th Ed: 13.4(a)
- Roth&Kinney, 6th Ed: 13.8(a)
- Roth&Kinney, 6th Ed: 13.11(a)
- Roth&Kinney, 6th Ed: 14.4
- Roth&Kinney, 6th Ed: 14.12
- Roth&Kinney, 6th Ed: 14.13
- Roth&Kinney, 6th Ed: 14.17
- Roth&Kinney, 6th Ed: 14.23

2 Problem Set Part B

1 (Petrick's Method and Static Hazard)

You are given the following Karnaugh map for a Boolean function ($F(w,x,y,z)$).

wx/yz	00	01	11	10
00	X	1	1	X
01	0	0	1	0
11	1	1	1	0
10	0	1	1	0

(Part A) Please find all the prime implicants and, if there are any, identify the essential ones among these prime implicants by using the map method. Once you have identified the prime implicants and the essential prime implicants, list the minimal cover(s) for this function.

wx/yz	00	01	11	10
00	X	1	1	X
01	0	0	1	0
11	1	1	1	0
10	0	1	1	0

Prime Implicants:

Essential Prime Implicants:

Minimal Cover(s):

(Part B) Please briefly discuss whether the minimal cover(s) you identified in *Part A* constitute static hazard-free implementation(s). If not static hazard-free, list all prime implicant pairs in your minimal cover(s) that can result in a static hazard and find a static hazard-free *minimal* sum-of-product implementation by using the Karnaugh map method.

wx/yz	00	01	11	10
00	X	1	1	X
01	0	0	1	0
11	1	1	1	0
10	0	1	1	0

[illegible][illegible]

(Part E) Once you identify the implications in *Part D*, please extend *Petrick's method* to add constraints for static hazard elimination and convert the equation to the sum of product formulation to identify a minimal hazard-free implementation. Make sure to compare your result with the hazard free implementation you identified in *Part B* to confirm that you have indeed arrived at the same solution.

2 (Multi-Level Forms)

As you recall from class, while the Quine-McCluskey algorithm is capable of generating the minimal two-level form of a boolean function, it may sometimes be beneficial in terms of area to instead utilize multi-level forms. In this question, we ask you to further explore this concept.

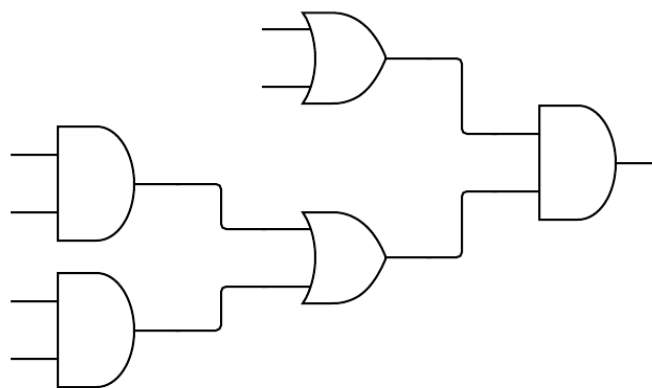
(Part A) For the following function, while one can implement the sum-of-products form using four 3-input gates, it is also possible to utilize only 2-input gates by adding a third level consisting of a single AND-gate (effectively making it an AND-OR-AND decomposition, as described in your textbook). As such, the function can be decomposed into the AND of two separate functions. Please illustrate your understanding of this concept by filling in the Karnaugh maps with the two separate functions that, when combined with an AND gate, form the desired function. The function should map directly to the incomplete schematic shown below. Then, please fill in the inputs to the incomplete schematic.

$wx \backslash yz$	00	01	11	10
00	0	0	1	1
01	0	0	0	1
11	1	0	0	0
10	1	1	0	0

$wx \backslash yz$	00	01	11	10
00				
01				
11				
10				

.

$wx \backslash yz$	00	01	11	10
00				
01				
11				
10				



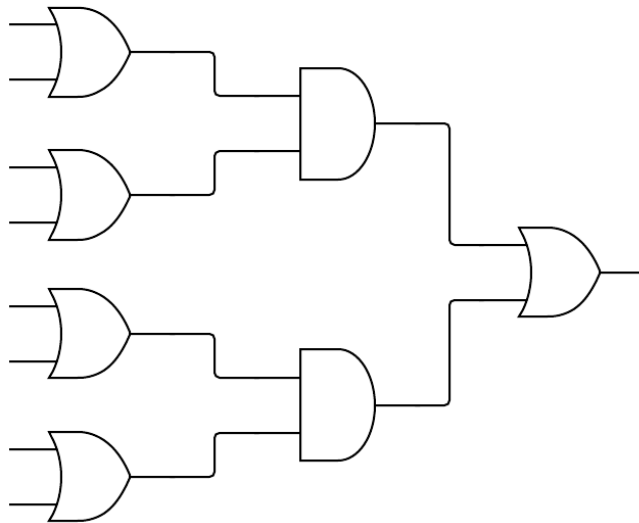
(Part B) Please decompose the following function, but this time into an OR-AND-OR decomposition. Again, use the incomplete schematic as a guide, and then please fill in the inputs for this function.

$wx \backslash yz$	00	01	11	10
00	1	1	1	1
01	1	1	0	0
11	1	1	1	1
10	0	0	1	1

$wx \backslash yz$	00	01	11	10
00				
01				
11				
10				

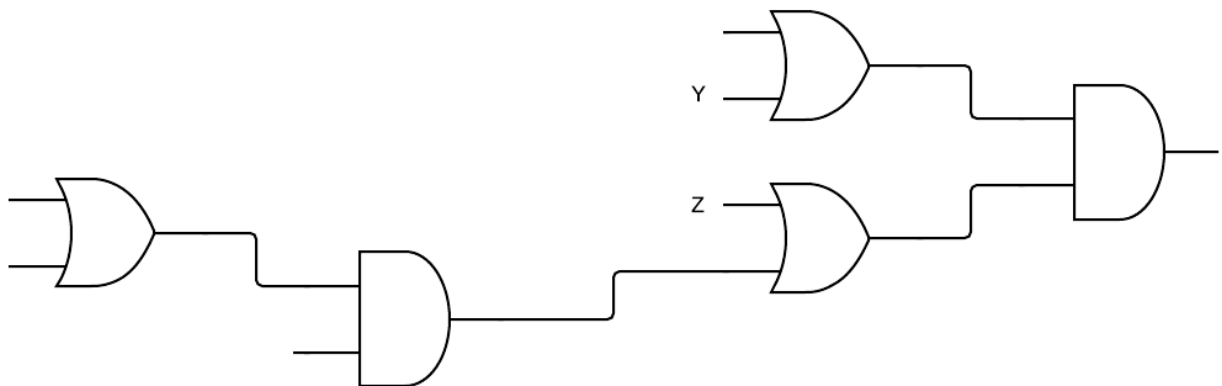
+

$wx \backslash yz$	00	01	11	10
00				
01				
11				
10				



(Part C) In this part, we ask that you now derive a four-level form of the function below. To help guide you, we have provided an incomplete schematic, but this time some of the inputs are already specified. Please fill in the remaining inputs.

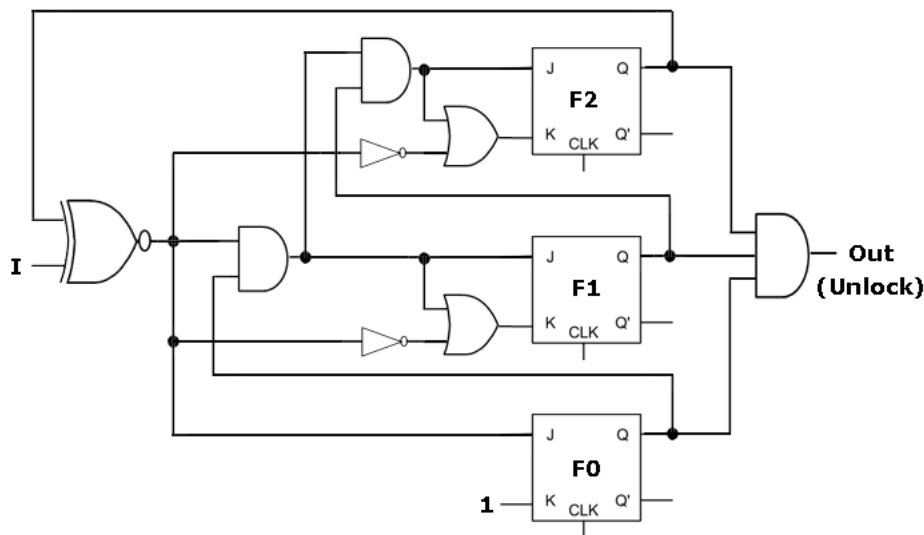
$wx \backslash yz$	00	01	11	10
00	0	0	1	0
01	0	0	1	1
11	0	1	1	1
10	0	1	1	1



3 (Lock Logic)

After graduating from UCSD, you find yourself employed at LockgicWorks, a company that manufactures hardware for security applications. Your first assignment is to design an electronic lock which unlocks a door when a user correctly inputs a certain binary sequence.

(Part A) You start by picking up where a previously fired intern left off. Apparently, your manager disapproved of her designs, saying that they could be easily brute-forced, and that the logic she used to implement them was far from efficient. **First, fill in the state table for this FSM. Then, identify the sequences that cause the circuit to output a “1”, unlocking the lock.** Assume that when the lock opens, the state is reset to $F_2F_1F_0 = 000$ asynchronously. (Hint: What does the FSM do if the XNOR output is fixed at 0? At 1? Which flip flops depend on the value of other flip flops?)



Current State $F_2F_1F_0$	Next State	
	$I = 0$ $F_2F_1F_0$	$I = 1$ $F_2F_1F_0$
0 0 0		
0 0 1		
0 1 0		
0 1 1		
1 0 0		
1 0 1		
1 1 0		
1 1 1		

Unlock Sequence:

(Part B-D) After analyzing this particular sequence detection logic, you realize that, using a few types of simple modifications, this FSM can be efficiently modified to recognize any particular binary sequence. Perhaps by fleshing out this generic utility of the circuit, you can restore your predecessor's reputation, and establish a reputation of your own while you're at it. In each of the following parts, you will explore a different type of modification you can make to increase the flexibility of the circuit.

(Part B) For each of the following sequences, describe the modifications that must be made to the lock FSM in (Part A) that will enable it to recognize the new sequence. *These modifications should only require modifying wiring or adding gates — all original gates should remain in the design.* Then, describe the kinds of sequences that such modifications enable the circuit to recognize.

i) 0011001

ii) 1100000

iii) 1011011

(Part C) Now that you have practice with the modifications in (Part B), **describe the set of sequences that these type of modifications can enable the circuit to recognize.**

(Part D) Each of the following sequences bear some relationship to the sequences in (Part B). For each sub-part i), ii), and iii) below, explain how you could modify the circuits recognizing (Part B)'s i), ii), and iii) sequences respectively to recognize the given sequence. Your modifications only require rewiring – no gates or inputs should be added or removed.

i) 001100

ii) 1100

iii) 10110

(Part E) As you certainly know, the fact that we use only 3 flip-flops limits the size of the sequence we could recognize. On the schematic below, show how you can efficiently increase this capacity by adding another JK flip-flop to your circuit, as well as two logic gates and an inverter. (Do not worry about the specific sequence that your result can detect – as you have learned in the previous parts, you can customize this.)

