

CSE 140 Homework Two

August 19, 2014

*Only **Problem Set Part B** will be graded. Turn in only **Problem Set Part B** which will be due on August 27, 2014 (Wednesday) at 4:30pm.*

1 Problem Set Part A

- Roth&Kinney, 6th Ed: 4.5
- Roth&Kinney, 6th Ed: 4.6
- Roth&Kinney, 6th Ed: 4.8
- Roth&Kinney, 6th Ed: 4.13
- Roth&Kinney, 6th Ed: 4.17
- Roth&Kinney, 6th Ed: 4.18
- Roth&Kinney, 6th Ed: 4.25
- Roth&Kinney, 6th Ed: 4.32
- Roth&Kinney, 6th Ed: 4.42
- Roth&Kinney, 6th Ed: 5.10
- Roth&Kinney, 6th Ed: 5.11
- Roth&Kinney, 6th Ed: 5.12
- Roth&Kinney, 6th Ed: 5.21
- Roth&Kinney, 6th Ed: 5.28
- Roth&Kinney, 6th Ed: 5.29
- Roth&Kinney, 6th Ed: 5.32
- Roth&Kinney, 6th Ed: 5.33
- Roth&Kinney, 6th Ed: 5.35
- Roth&Kinney, 6th Ed: 5.42
- Roth&Kinney, 6th Ed: 6.4
- Roth&Kinney, 6th Ed: 6.5

- Roth&Kinney, 6th Ed: 6.11
- Roth&Kinney, 6th Ed: 6.17
- 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: 7.15 (a)
- 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: 13.10(a)

2 Problem Set Part B

1 (Minterm & Maxterm Properties)

This question concerns a set of transformations on a 4-input function $F(a, b, c, d)$ that is implemented in a 2-level sum-of-products form. You are provided with three transformations, defined as follows:

1. Complement all the inputs to F , and get a function G_1 .
2. Replace the AND gates in F with the OR gates and vice versa, and get a function G_2 .
3. Complement all the inputs and exchange the AND and OR gates in F , and get a function G_3 .

(Part A) Given the number of **minterms** in F , is it possible to determine the number of **maxterms** in G_1 , G_2 , or G_3 ? For any of the three functions, if you think it is possible to determine the number of its **maxterms**, please complete the corresponding column in the following table. If you think it is impossible to determine the number of its **maxterms**, please provide a brief reasoning.

Minterm count of F	Maxterm count of		
	G_1	G_2	G_3
k ($0 \leq k \leq 16$)			

(Part B) Is there any particular function F that would not change under the first transformation, i.e., $F = G_1$? If you think there is, please give an example of F . Otherwise, if you think $F = G_1$ is impossible, please provide a brief reasoning.

(Part C) Is there any particular function F that would not change under the second transformation, i.e., $F = G_2$? If you think there is, please give an example of F . Otherwise, if you think $F = G_2$ is impossible, please provide a brief reasoning.

(Part D) Is there any particular function F that would not change under the third transformation, i.e., $F = G_3$? If you think there is, please give an example of F . Otherwise, if you think $F = G_3$ is impossible, please provide a brief reasoning.

2 (Karnaugh Map Equivalence)

In this problem we explore the equivalence of K-Maps under some basic transformations. In particular we see what happens when the variables are re-positioned and when the variable values are shifted. To be a useful K-Map, Hamming adjacency of adjacent squares must be maintained; hence, transforms must preserve **Hamming adjacency** for variable value assignment.

For clarification, here is a variable re-positioning:

$x \backslash y$	0	1
0	1	1
1		

$y \backslash x$	0	1
0	1	
1	1	

And here is a variable value shifting:

$x \backslash y$	0	1
0	1	
1		1

$x \backslash y$	1	0
0		1
1	1	

(Part A) Given the following K-Maps, complete the partially filled in transformed K-Map.

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

=

$xw \backslash zy$	00	01	11	10
00				
01	1			1
11				
10				1

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

=

$wy \backslash xz$	11	01	00	10
10	1			1
11		1		
01				
00	1			1

(Part B-D) For the next parts we give you the original and the fully transformed K-Map; however, we leave out the complete definition of the transformation. Please fill in the name and/or value assignments which make the two K-Maps equal. If no such name/value assignment exists, cross off the transformed K-Map and write “INVALID” next to it.

(Part B) Here we just leave out the *variable name* definitions.

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

=

$\backslash$	00	10	11	01
01				
11	1	1	1	1
10			1	
00				

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

=

$\backslash$	01	00	10	11
11		1	1	
10			1	
00		1		
01	1	1	1	1

(Part C) Here we just leave out the *value* definitions. Remember that **hamming adjacency must be preserved** for the variable value assignment.

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

=

$zw \backslash xy$				
	1	1		
			1	1
	1	1	1	1
	1			1

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

=

$wx \backslash yz$				
		1	1	1
	1			
	1		1	
	1			

(Part D) Now both the *variable name* and most of the *value* definitions are hidden. Remember that **Hamming adjacency must be preserved** for the variable value assignment.

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

=

$\backslash$	00			
	1	1		
				1
00	1	1		

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

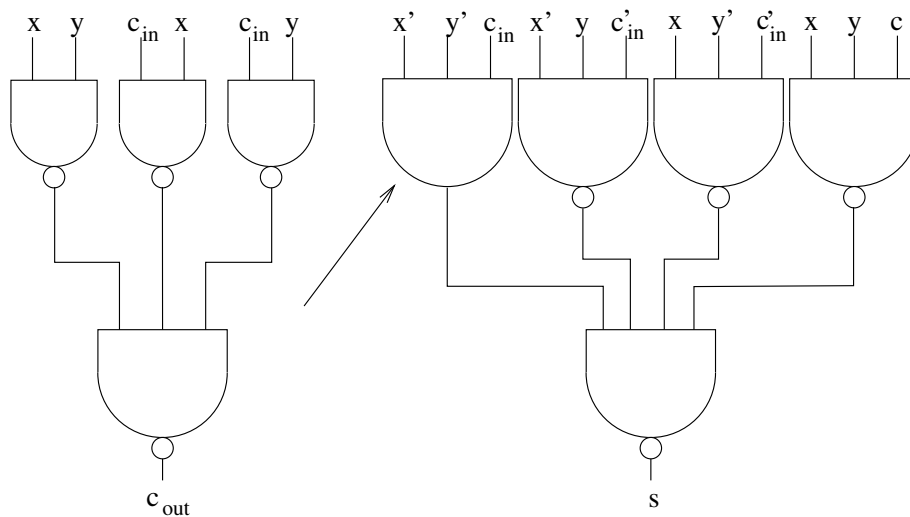
=

$\backslash$			10	
01	1		1	
				1
	1	1		1
	1	1		

3 (Adder Implementation)

You have just signed on with a start-up that plans on producing the next big pocket calculator, and instead of outsourcing the job of building circuits to a company that knows how to do it, your miserly CEO has declared that all production will happen in-house by your engineers. Unfortunately none of them graduated from UCSD and as such they have all done a terrible job, and as the only competent engineer among them, you must fix their mistakes! Thank God, at least their mistakes all seem to be centered on incorrect gate utilizations; the interconnects and the carry to the least significant bit all seem to be safe and sound.

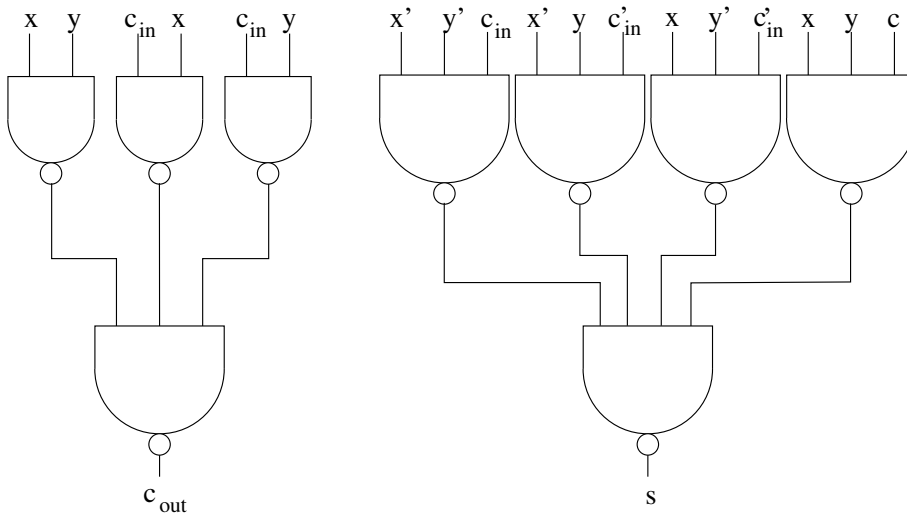
(Part A) The first engineer has attempted to create a NAND-only implementation of a full adder, but has **inadvertently used an AND gate where he should have used a NAND**, as outlined in the diagram below. Though your miserly CEO is resigned to his fate of having a malfunctioning adder, he would like you to help him salvage the situation. Given the known error, **please fill in the result of the addition listed below** that one would acquire by attempting to use this broken adder. Now that you have correctly computed the result of this incorrect 5-bit addition, you realize that the single-bit sum logic, s , turns out to be equivalent to a single gate! Please show this simplified boolean algebra representation for the incorrect single bit sum logic, s , and how you actually managed to map this 5-gate boolean circuit to a single gate.



Engineer's incorrect implementation

x	01101
y	+ 01011
$x + y =$	
$s =$	

(Part B) The second engineer was also aiming for a NAND-only implementation, but has fallen victim to faulty hardware. While she does not know which gate is problematic, she does know that **exactly one of her NAND gates is broken and always outputs a 1**. To figure out which gate has gone wrong, the engineer has run an addition through her adder, and the results are outlined below. Using this information, **please indicate on the diagram below which gate is malfunctioning**. Once you have determined the error, please fill in the subsequent 7-bit addition with what the incorrect adder would produce.



x		0010
y		+ 1111
s		1001

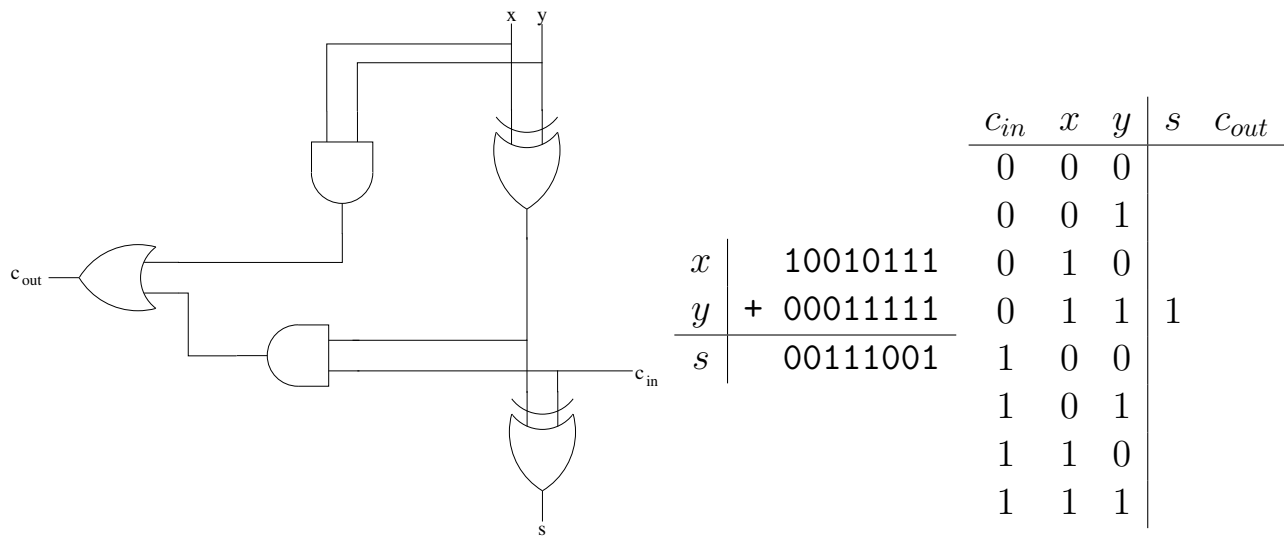
x		0101110
y		+ 0010100
s		

Correct NAND implementation (please circle broken gate)

(Part C) The third engineer has chosen the minimal-gates implementation of a full adder, but was sloppy in specifying which gates they were! In this third and final implementation for you to debug, **a single gate has been replaced with some incorrect alternative gate**. Once again, using the available output from an addition run through the malfunctioning adder, **please determine which gate has been swapped by circling it, and clearly write what it has been changed to**.

As we realize this is a somewhat challenging task, let us help guide you through at least the beginning of the necessary reasoning chain. One thing to realize by looking at the schematic is that certain gates will affect only c_{out} , others only s , and a special gate may modify both. Another important fact to keep in mind is that an incorrect gate will **not** necessarily cause a problem in all cases; typically only certain input combinations will lead to a faulty output, and the provided example should illuminate these.

To aid you, a truth table has been provided for you to record the sum/carry outputs of various input combinations. **This will not be graded**, but you may find it useful.



Correct minimal implementation (please circle broken gate)