Lab 1:

Project 1:: Small Design Lab

By. Melvin Evans
(ID#4692)

CSc 137
Professor Daryl Posnett
March 10, 2021

Problem Number One

1. Design a circuit that has three inputs and two outputs. Output 1 indicates when the 3 bit unsigned input is odd and output 2 indicates when the 3 bit unsigned input value is even. Provide the initial equations that describe your circuit and minimize your output equations using k-maps. Draw your circuits from the minimized equations using AND, OR, and NOT gates. Now, given that you have two outputs, two initial equations, two k-maps, and two solutions, you should have two circuits:
   A. These circuits should be minimal, but, what does this mean?
   B. If possible, draw a (third) simpler circuit that will give both outputs.

Solution:

Truth table

x	y	z	O1	O2
0	0	0	1	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	1	0
1	1	1	0	1

Original Equations + K-Maps

O1= (~x~y~z)+(~xy~z)+(x~y~z)+(xy~z)

z\xy	00	01	11	10
0				
1	1	1	1	1

O2= (~x~yz)+(~xyz)+(x~yz)+(xyz)

z\xy	00	01	11	10
0	1	1	1	1
1				

Simplified Equations

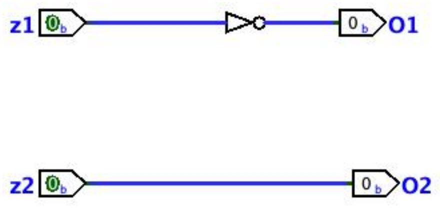
O1= z

This is due to the only value that stayed the same in the first output's grouping is that z is on.

O2= ~z

This is due to the only value that stayed the same in the second output's grouping is that z is off.

Circuit

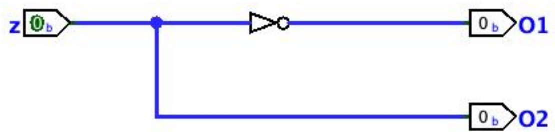


These circuits line up with what I thought they would be as when you learn counting in binary you learn that to make an odd number there has to be a 1 in the first bit and to make an even there has to be a 0.

A.

These circuits are minimal which means that they take the lowest number of gates and time in order to get the same output as the original equations. Therefore, making them optimal for use inside of computers.

B.



These can be simplified this way due to both of them relying solely on the z input with the only difference is that one requires their output to not z. So, branching the z this way helps us simplify the representation of the circuit.

Problem Number Two

2. Design a circuit that has four inputs and three outputs. Output one indicates whether the two's complement signed 4 bit input is larger than 5, output two indicates if the signed 4 bit input is smaller than -5, and output three indicates that the signed 4 bit input is equal to zero. Provide the initial equations that describe your circuit and minimize your output equations using k-maps. Draw your circuits from the minimized equations using AND, OR, and NOT gates.
   A. Redraw your equations using NAND gates only.
   B. You should have three NAND gate circuits. Think about whether or not you can reuse common segments of your NAND only circuits. If so, draw a new common circuit that is as minimal as you can achieve and describe your process for creating your circuit. If not discuss why it's not possible.

Solution:

Truth Table

	w	x	y	z	O1	O2	O3
0	0	0	0	0	0	0	1
-1	1	1	1	1	0	0	0
-2	1	1	1	0	0	0	0
-3	1	1	0	1	0	0	0
-4	1	1	0	0	0	0	0
-5	1	0	1	1	0	0	0
-6	1	0	1	0	0	1	0
-7	1	0	0	1	0	1	0
-8	1	0	0	0	0	1	0
7	0	1	1	1	1	0	0
6	0	1	1	0	1	0	0
5	0	1	0	1	0	0	0
4	0	1	0	0	0	0	0
3	0	0	1	1	0	0	0
2	0	0	1	0	0	0	0
1	0	0	0	1	0	0	0

Original Equations + K-Maps

O1= (~wxyz)+(~wxy~z)

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

O2= (w~xy~z)+(w~x~yz)+(w~x~y~z)

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

O3= (~w~x~y~z)

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

Simplified Equations

O1= (~wxy)

Simplified using K-Map above.

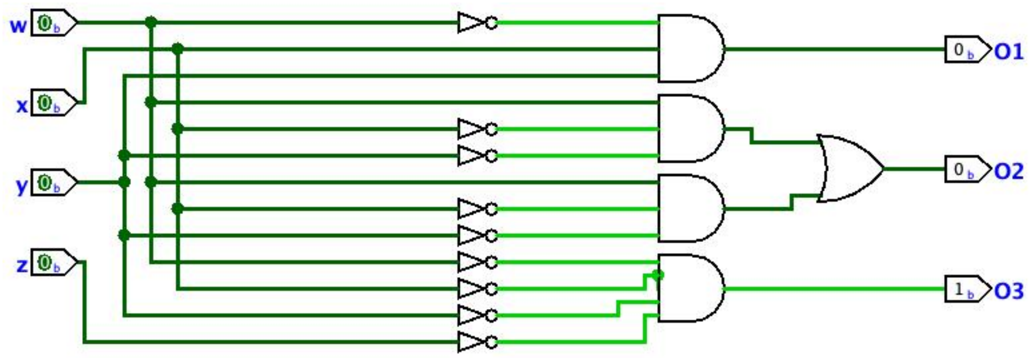
O2= (w~x~y)+(w~x~z)

Simplified using K-Map above.

O3= (~w~x~y~z)

This output was already in its simplest form from the beginning.

Circuit



Works as long as the inputs are in two's compliment form.

A.

Key: (NOT, AND, OR gates to NAND)

NOT = NAND

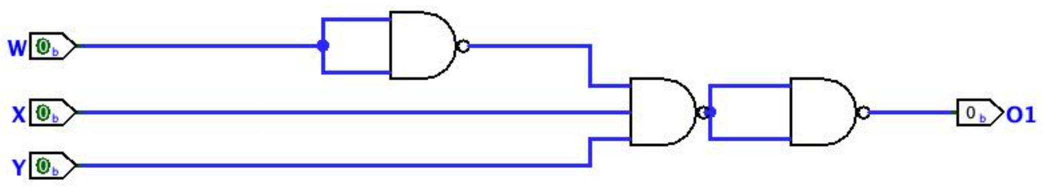
AND = two NANDS in succession

OR = NAND on the inputs and then both inputs ran through another NAND gate

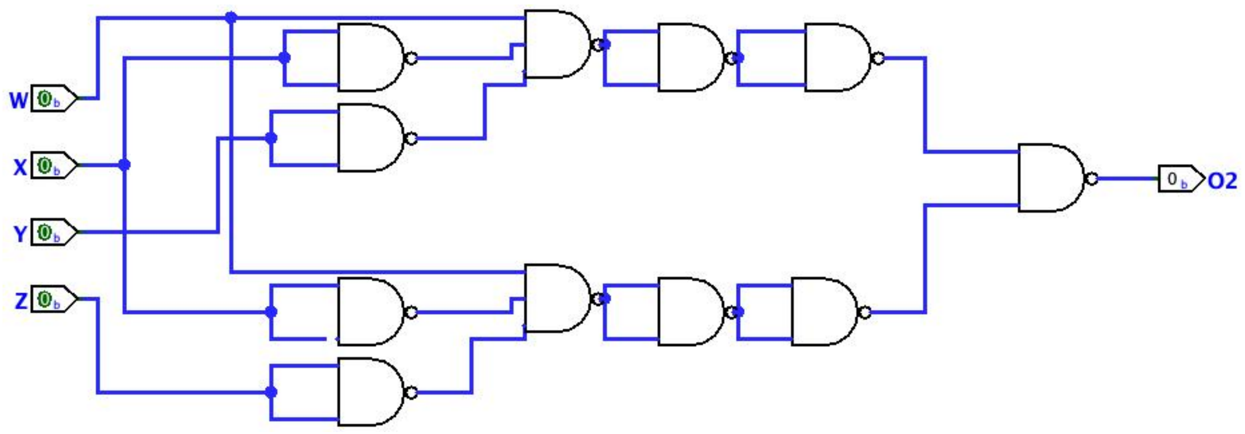
'.' = NAND

Simplified Equations + Circuits

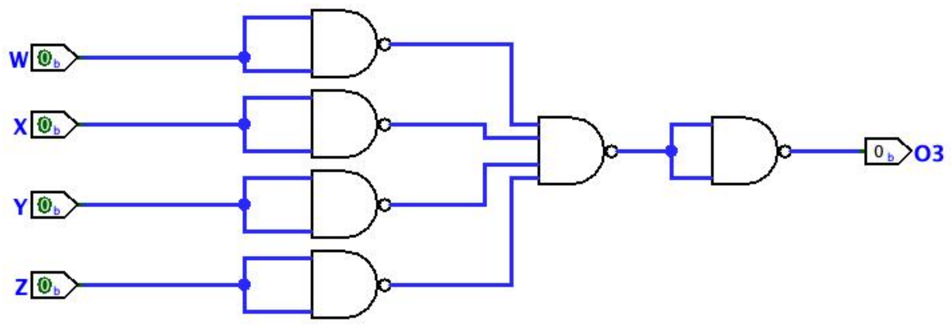
O1= ((w.).x.y).



O2= (w.(x.).(y.).)…(w.(x.).(z.))

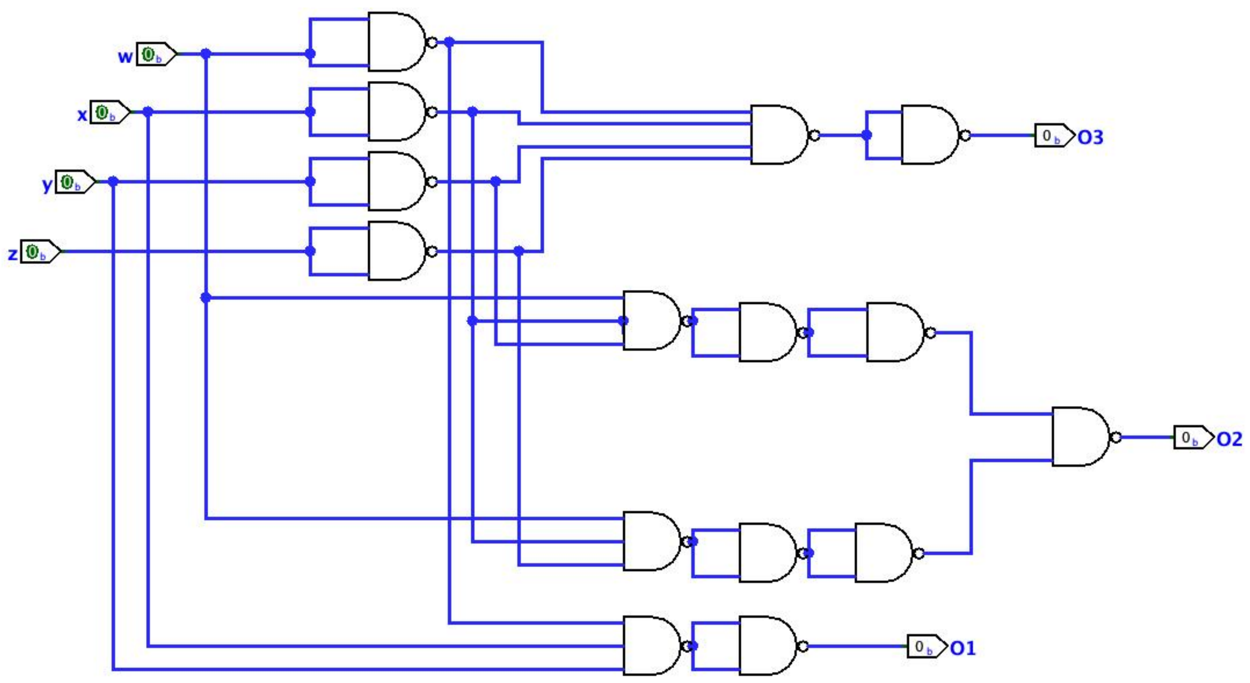


O3= ((w.).(x.).(y.).(z.)).



B.

Yes, there are ways in order to minimize the number of gates being used. This is done by using the NAND gated versions of each of the inputs taken from O3's circuit and use those as replacements for all of the not values in other circuits. Therefore, cutting out 5 gates overall from the whole circuit.


w
x
y
z
0b
O1
O2
O3

Problem Number Three

3. Consider an apartment that has two bedrooms and a shared hallway to a common area. There is a single hall light and a switch in each room to turn on the hall light. No matter what position each switch is in, flipping either switch turns the hall light on if off, or off if on. Your goal is to design a logic circuit to represent this problem. Define the inputs and outputs and then provide the initial equations that describe your circuit and minimize your output equations using k-maps. Draw your circuits from the minimized equations using AND, OR, and NOT gates.
   A. Now that you've designed this circuit can you reduce the number of gates? You may use any kind of gates that we have discussed in class.
   B. Your landlord is thinking of adding a third bedroom. Can your design be easily modified to accommodate this? Why or why not?

Solution

Truth Table*

x	Y	O1
0	0	0
0	1	1
1	0	1
1	1	0

*This table is running off of the assumption that the initial state of the switches is both off when the rooms are created. This assumption is made to allow representation of the effects of turning on and off the lights through the use of flipping switches in each of the rooms x and y.

Original Equation + K-Map

O1= (~xy)+(x~y)

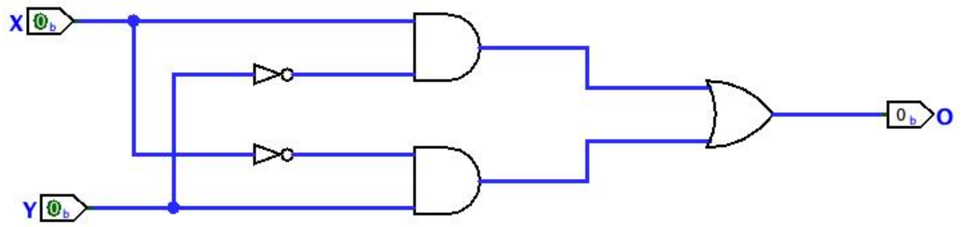
x\y	0	1
0		1
1	1	

Due to the diagonal grouping and our truth table, we can say that O1 represents the XOR of x y.

Simplified Equation + Circuit

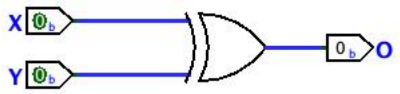
O1= (~xy)+(x~y)

This is due to the original equation already being simplified in terms of making a XOR representation when you are not using an XOR gate. Therefore, this is the current representation of the circuit.



A.

Since there is the aim of using as little gates as possible, we should now implement the XOR gate which will lower the amount of gates by two in our drawing. Resulting in the below circuit.



B.

With the way that the above solution is written it does allow easy modification to be made in order to accommodate a third room being added on. To show this here are the new truth table, K-Map, and circuit for three bedrooms with the prior specifications

Truth table

x	y	z	O1
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0

1	1	0	0
1	1	1	1

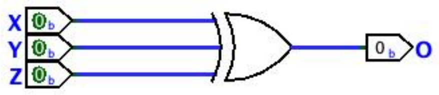
This graph makes the same assumption as the truth table in Part 3.

K-Map

z\xy	00	01	11	10
0		1		1
1	1		1	

Through the same logic as discussed in Part 3 due to the groupings only being able to be formed through diagonals this table represents an XOR gate. Due to this we can presume that the relation can be represented as x XOR y XOR z. Which is logically equivalent to running all three of these inputs through a single XOR gate and taking the input as tested through Logism. Therefore, the change from two rooms to three would only require the minor change of adding another input to the already set up system. As depicted in the following circuit.

Circuit



Problem Number Four

4. Design a circuit that has four inputs and three outputs. The four inputs are considered to be two 2-bit inputs. One output views the two inputs to be binary numbers and indicates when the two input number are not equivalent. The other output considers the two inputs to be stone-age binary inputs and indicates when the two binary inputs are equivalent (see the concept of stone age unary on page 62 of Digital McLogic Design. In stone age binary, 0=finger down, 1=finger up, note that the fingers can be held up in any order). The third output indicates when the previously described outputs are both in an "on" state. For this problem, implement the first two outputs using POS forms; implement the third output in any way you deem appropriate, but minimize your *use of gates* in the implementation.

Solution

Stone-age binary: has three states 0 (represented by 00), 1 (represented by 01 or 10), and many (represented as 11)

Truth table

w	x	y	z	O1	O2	O3
0	0	0	0	0	1	1
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	1	1	1
0	1	1	1	1	0	0
1	0	0	0	1	0	0
1	0	0	1	1	1	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

Due to POS we look at the 0's not the 1's like previously done. POS is only used for O1 and O2. SOP is used for O3.

Original equations + K-Maps

O1= (~w+~x+~y+~z)(~w+x+~y+z)(w+~x+y+~z)(w+x+y+z)

yz\wx	00	01	11	10
00	0			
01		0		
11			0	
10				0

O2= (~w+~x+~y+z) (~w+~x+y+~z) (~w+~x+y+z) (~w+x+~y+~z) (~w+x+y+z) (w+~x+~y+~z) (w+~x+y+z) (w+x+~y+~z) (w+x+~y+z) (w+x+y+~z)

yz\wx	00	01	11	10
00		0	0	0
01	0		0	
11	0	0		0
10	0		0	

O3= (~wxy~z)+(w~x~yz)
This again was done in SOP.

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

Simplified Equations

O1= A XOR B

(A= wx; B=yz)
The use of A and B is done due to the fact that you are checking on if the pairs are even or not. Allowing for the use of pairing like this.

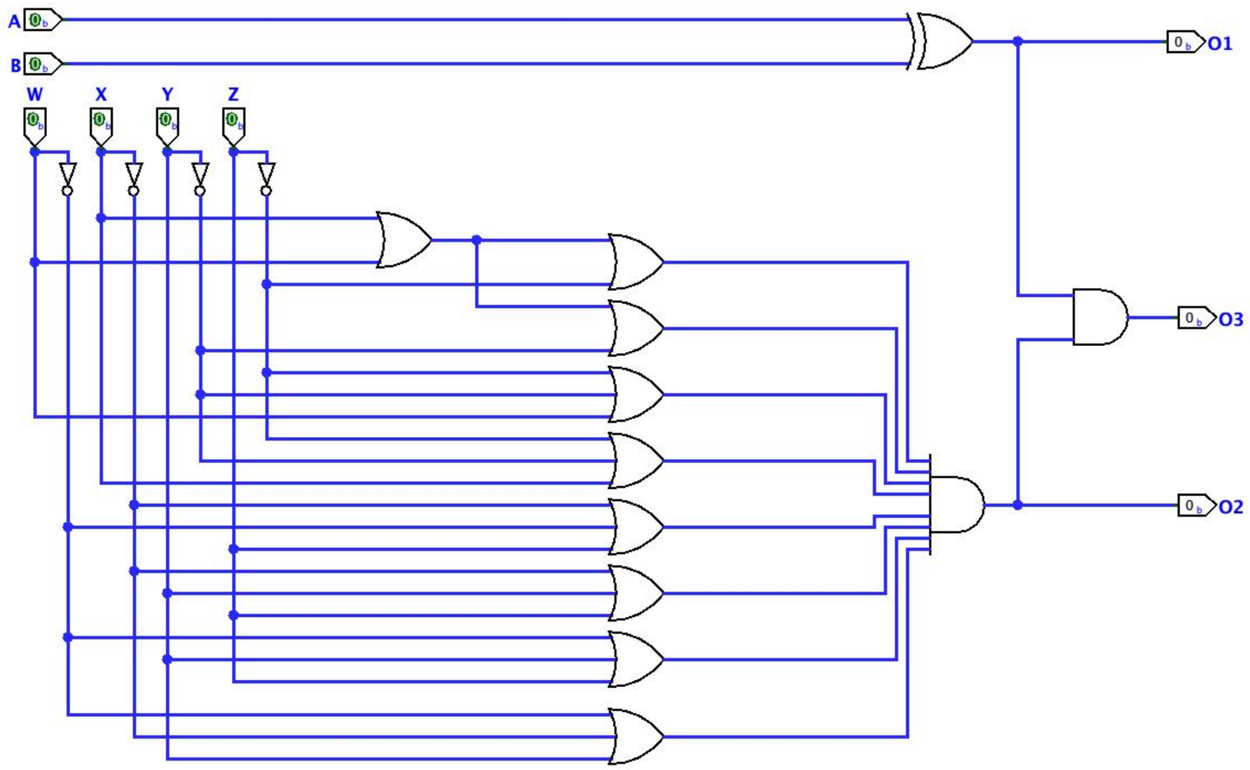
O2=
(w+x+~z)(w+x+~y)(w+~y+~z)(x+~y+~z)(~w+~x+z)(~x+y+z)(~w+y+z)(~w+~x+y)

This is drawn from the POS K-Map from above.

O3= O1*O2

We can decide this through looking at the truth table which has O3 at 1 when both O1 and O2 are also equal to 1. Everywhere else where O1 and O2 is 1 O3 is 0. Therefore, we can use and AND gate on O1 and O2 and come up with O3.

Circuit



All three are represented in this circuit diagram due to using O1 and O2 as input for O3.

Problem Number Five

5. A given circuit has four inputs. Two of the inputs are considered the fractional portion of a binary number while the other two inputs are considered the integral portion of the binary number. The outputs of this circuit should represent a 2-bit binary number associated with the 4-bit input but with rounding up and down. In other words, if the input is greater or equal to 0.5, the output should represent the input rounded up. Otherwise, it output should represent the input rounded down to the nearest integer. Note: For this exercise you may assume that the input will never be larger than can be represented by the output, in other words, it will not *overflow.* You may treat any overflow input conditions as don't cares. Provide a truth table, a reduced Boolean equation and circuit diagram for your solution using AND, OR, and NOT gates.

Solution

Truth table

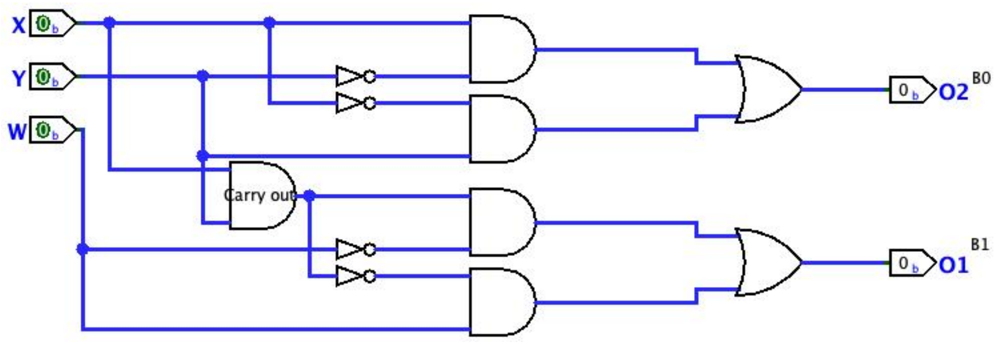
w	x	y	z	Ru
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

K-Map for Ru

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

The simplified equation for this map is simply equal to y. Meaning that only y decides when we should round or not Which is helpful in our circuit implementation.

Circuit



I used the setup of two half adders due to the problem requiring us to round up only if decimal is greater or equal to .5. So, the simplest way is to use an already crafted circuit and use the carry out from the first half adder to use as one of the two inputs for the second. Due to us not worrying about the overflow on this problem we will not need to have the carry out for the second half adder. Combined these forms the circuit needed to do this computation with O2 representing the index 0 bit and O1 representing the bit in index 1 when the indexes start from right to left.