Code Conversion

Prerequisites

You need to have both your input and output circuitry working correctly from week 2. You need to understand the basics of code conversions, the value of don't care states, and the process of minimization.

Objectives

The objective of this laboratory assignment is to design and build a code converter that will take the POSTNET 2-out-of-5 code as input and deliver an XS3 code as output. This will enable the post office to deliver the mail using a decimal zip code.

Theory

The POSTNET code is as below. It is a 2-out-of 5 code which is weighted 74210. All of the numeric values from 0 to 9 can be accommodated, though the 0 value is an exception to the 74210 weighting. The outputs from the code conversion are to be the XS3 values from 0 to 9.

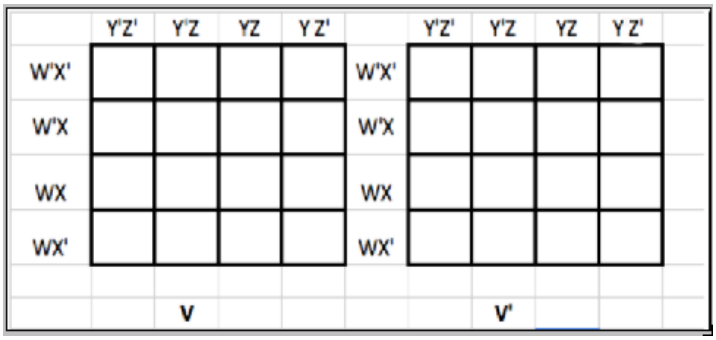
		7	4	2	1	0
		V	W	X	Y	Z
0		1	1	0	0	0
1		0	0	0	1	1
2		0	0	1	0	1
3		0	0	1	1	0
4		0	1	0	0	1
5		0	1	0	1	0
6		0	1	1	0	0
7		1	0	0	0	1
8		1	0	0	1	0
9		1	0	1	0	0

Procedure

Create a complete table showing the POSTNET input code values against the XS3 output values as partially indicated below.

	POSTNET						*XS3*			
	7	**4**	**2**	**1**	**0**		**8**	**4**	**2**	**1**
	V	**W**	**X**	**Y**	**Z**		**D**	**C**	**B**	**A**
0	**1**	**1**	**0**	**0**	**0**		**0**	**0**	**1**	**1**
1	**0**	**0**	**0**	**1**	**1**		**0**	**1**	**0**	**0**
2	**0**	**0**	**1**	**0**	**1**		**0**	**1**	**0**	**1**

I suggest that you create a 5-variable Karnaugh Map - see sample below. You need to create four 5-variable Maps for the DCBA outputs. I suggest that you first determine the values of the don't care states, plot these and then add the 1's for the required outputs. For each double Map there will be no more than five 1's.



Minimize each Map for the four outputs, document the Boolean functions, and build a circuit to implement the converter. It's suggested you do this **<u>one output bit at a time</u>**, checking that you've minimized your Map as much as possible using the corners, edges, and '2-D' layout of the two Maps. Make sure you're accounting for every don't care state, and then test it in LogicWorks. Remember that you have a limited number of gates. Check the components sheet and count the number of ANDs and ORs you're provided in total.

To walk you through the above steps and help you check your work and understanding, below is how the Karnaugh map should look like for output **D**. It is highly recommended to start with output D and create its map *on your own* first. Only after that, you could compare your work with the provided map for checking purposes.

D=VW'+WX+WY

V	Y'Z'	Y'Z	YZ	YZ'
W'X'	X	1	X	1
W'X	1	X	X	X
WX	X	X	X	X
WX'	0	X	X	X

V'	Y'Z'	Y'Z	YZ	YZ'
W'X'	X	X	0	X
W'X	X	0	X	0
WX	1	X	X	X
WX'	X	0	X	1

Note how we took advantage of the don't care states to group as many cells as we can in order to minimize the circuit. Now simulate the circuit for output **D** using *LogicWorks*. Note the equation for output **D** above.

When you've accomplished the minimization and LogicWorks simulation for

every output bit, we recommend drafting out every chip you'll use. Google the part number to see the pin arrangement and label the inputs and outputs (the dot/semicircle is on the same edge as pin 1!). Lay out your connections on paper before going through the effort of wiring. If you have incorrect outputs after wiring, use your probe or DMM to find where the problem begins.

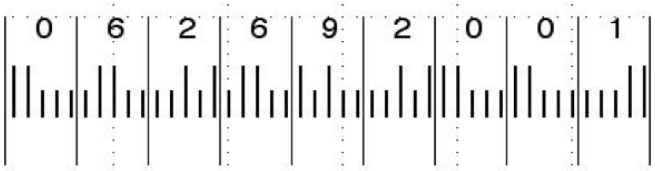
Testing

You should test your circuit exhaustively for the 10 possible valid inputs. I suggest that you decide on two non-valid input states, calculate and document the outputs by plotting them on the Map and determine that you are correct by validating them in your circuit design.

Deliverables

Your circuit must work correctly and your documentation should agree with your circuit outputs. Your TA must check your design and implementation. A full report is required for this lab.

Your report should contain the POSNET symbology code for your home area, that is, 5-digit code plus 4. An example is shown below for a UConn department.



Also include a documentary note of the two invalid inputs for which you tested.

References

http://www.barcodeisland.com/postnet.phtml

Questions

How many non-valid input codes are there?

Would your design be simpler if there was no zero input? Does your design conflict with your answer?