Laboratory # 5: Multiplexer Design (pt 2)

Aaron Goldsmith
Thomas King
ELEN 21 -- 35219
May 8th, 2016

Introduction:

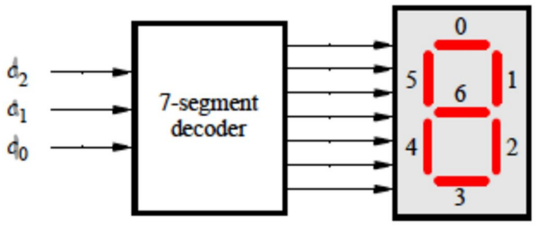
For this lab we had to display 4 different characters on five, 7-segment displays. By first drawing our K-maps for each character, we could then coming up with each equation to display the correct characters. Since we could treat each letter as a function to be called for specific inputs. Our Mux should have its inputs connected to the character functions, so that changing the selection of the mux, will change the order which the characters are displayed. The decoder should be able to take in 3 inputs, and output a single hex value containing the segments that should illuminate on each display.

By building a hierarchical design, we could use the multiplexers we built from part one and implement them in part 2 to rotate the characters on the displays.

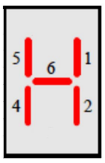
Pre lab work:

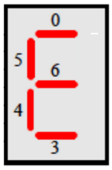
Aaron Goldsmith
Prelab 5
May 1, 2016
Grade 10/10
Elen 21 - Monday lab section

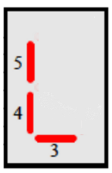
The following letters (H,E,L,O) were determined from the minterms in the picture below:

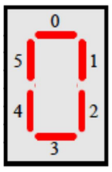


Select 000 ⇒ H = ∑m(1,2,4,5,6)
Select 001 ⇒ E = ∑m(0,3,4,5,6)
Select 010 ⇒ L = ∑m(3,4,5)
Select 011 ⇒ O = ∑m(0,1,2,3,4,5)





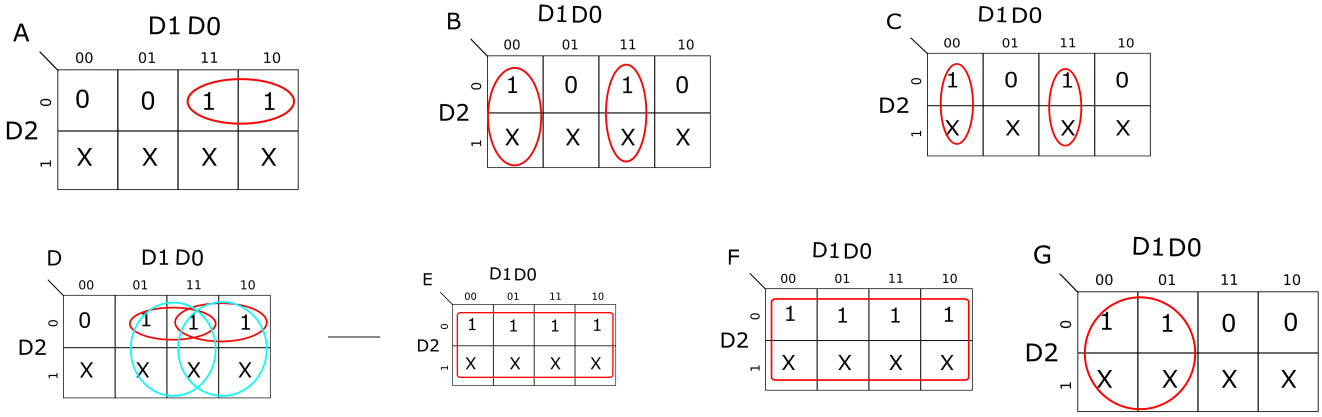




D2	D1	D0	A = 0	B = 1	C = 2	D = 3	E = 4	F = 5	G = 6
0	0	0	0	**1**	**1**	0	**1**	**1**	**1**
0	0	1	**1**	0	0	**1**	**1**	**1**	**1**
0	1	0	0	0	0	**1**	**1**	**1**	0
0	1	1	**1**	**1**	**1**	**1**	**1**	**1**	0
1	0	0	*x*	*x*	*x*	*x*	*x*	*x*	*x*
1	0	1	*x*	*x*	*x*	*x*	*x*	*x*	*x*
1	1	0	*x*	*x*	*x*	*x*	*x*	*x*	*x*
1	1	1	*x*	*x*	*x*	*x*	*x*	*x*	*x*

Displaying the information in a truth table allows us to debug easily. We can see what each combination of inputs will output. Want A•B•C•D•E•F•G

K-Maps

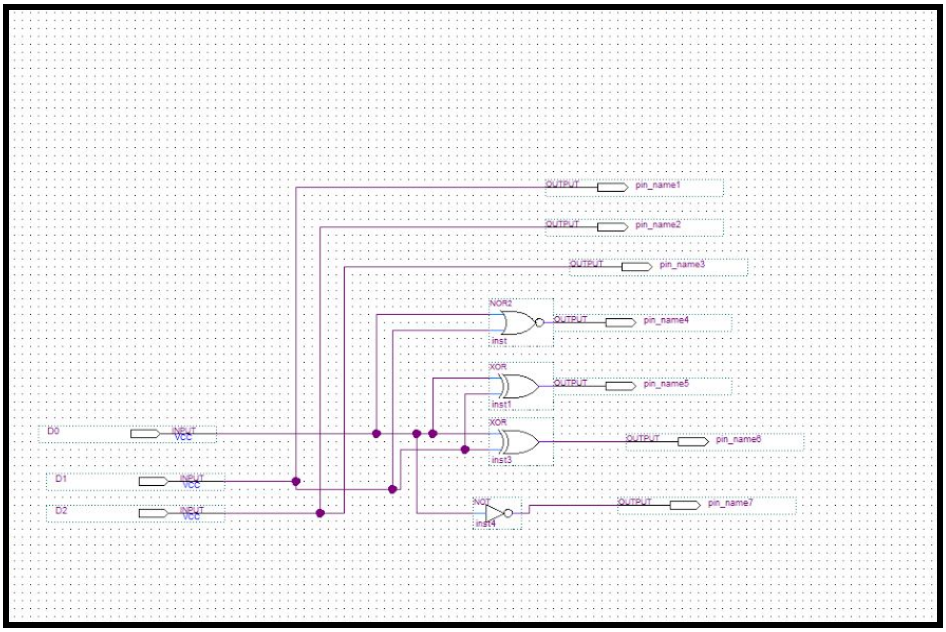


A = (D2)'• D1 **B** = (D0)'•(D1)' + D0•D1 **C** = (D0)'•(D1)' + D0•D1
D = D1 + D0 **E** =(D2)' **F** = (D2)' **G** = (D1)'

Schematic for 3:7 (this didn't work, so we had to find the negation of the function)

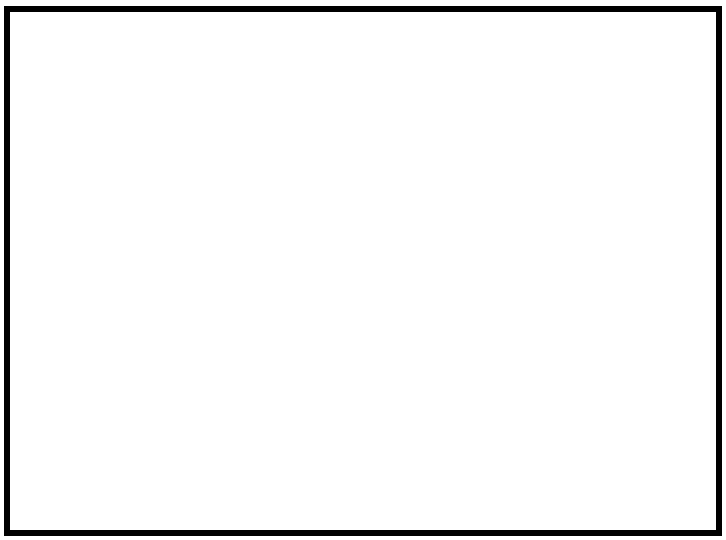
Lab Data + Analysis:

Schematic: *7 segment decoder*



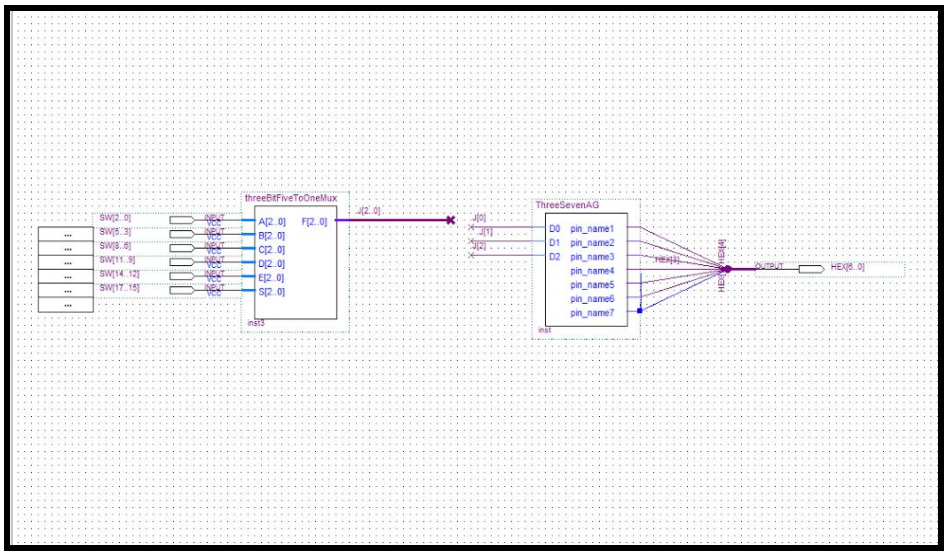
This decoder functions by taking in three inputs (or a three bit number) and pushing them through a set of logical gates. The output will be the corresponding 'on' and 'off' states of the light segments *a* – *g* . (*a* is pin_name7, and they follow to *g* is pin_name1). Because the decoder is active low, we used the inverse function gained from our prelab truth table, to describe the outputs. This decoder allows us to set our values to out the different letters: H, E, L, and O. For example, to select the character "H" all three inputs are set to zero. This means output pins 1, 2, 3, 5, and 6 are set to 0

Schematic: *Three bit 5 -to- 1 MUX*



This 3bit 5:1 will compute a hex value containing the selections from the MUX. By using the electrical bus, we were able reduce the number of connections between inputs, MUX, and outputs.

Schematic: Five- to-1 ((3-bit) to (7-bit)) MUX



This MUX should take the bit inputs from our selection mux, and encode them into each 7 segment controller value to correctly display which segments of the controller should be on and off.

In the example described above, this would activate HEX[1], HEX[2], and HEX[4-6], forming the character H.

Schematic: Rotating H E L L O decoder

The way this decoder works is that once you set an initial state for each controller (000 = H, 001 = E, 010 = L, 011=O). Each set of three bits for the 5 7-segment-controllers hold the “starting” letter, and then switches 15,16,17 could be used to select how many times rotated left.

Conclusion:

We ran into a few issues this lab. We had to recreate the three bit 5 to 1 MUX because the ones we created in our last lab weren’t easy to work with. Another issue we had was with file organization, we were a bit confused because we didn’t have a clean structure set up before we started. When dividing the 3 bit outputs from the 5:1 MUX into their respective bits we made a couple errors which we had to fix. Also, our design for the decoder wasn’t functioning properly, so we had to redesign that. After fixing all that we managed to finish the lab, however, and I think we learned a lot in solving all those problems.