Assignment # 5

Due Friday February 15th

Read Chapter 5 Sections 1, 2, and 3.

1. An encoder is a digital function that performs the inverse operation of a decoder (or demultiplexer or selector). That is, an encoder has 2^n inputs and n outputs. The outputs generate the binary code corresponding to the input value. In order to take care of the case where two inputs would be on simultaneously, one can design a *priority encoder* such that if 2 or more inputs are true at the same time, the input having the highest priority takes precedence.
   Design a 4-input (I_3, I_2, I_1, I_0), 3-output (A_1, A_0, V) priority encoder where the input indices correspond to their priorities and where V = 0 if all inputs are 0 and V = 1 otherwise. In other words, if inputs I_i and I_j are on at the same time, and $i > j$, then I_i has priority. For example the input (0,1,1,1) should generate the output (1,0,1) (note the input (0,1,0,0) would also generate the output (1,0,1)).

2. Chapter 4 Exercise 4.15. In (b), you may use an OR gate that has less than 16 inputs. In (c), draw a block diagram of the ROM, label the address lines, and explain what bit values you would store at each memory location. In (d), don't forget to minimize the logic before implementation.

3. You have a collection of 16 CDs. To keep track of your favorites, you barcoded each CD. Unfortunately, the barcoding system only outputs a four-bit number, with no information as to whether the CD is one of your favorites. Create a system that will output a 1 if the barcoding system detects a favorite CD. Your favorite CDs are numbers 0, 5, 6, 9, 10, 12, and 15 . which doesn't simplify nicely via a K-map. Implement the function using a 4:16 decoder and 3-input OR gates.

4. Design a 2-bit full adder using two 4:1 multiplexers.

5. In Assignment 3, you created a DesignWorks project for a binary full-adder. You drew the adder schematic, created a block symbol for the adder, verified that your adder functioned correctly, and cascaded four full adders to make a ripple-carry adder similar to the shown on pg. 249 of Katz. Now you will construct a 4-bit carry-lookahead adder using the same procedure you used in assignment 3, and compare your two adders.

   - Construct a 4-bit carry-lookahead adder (CLA). Use pg. 255-256 of Katz as a reference, but implement your carry logic more like pg 256 than page 255, that is rather than computing P and G as a block of 4, construct a two-level tree of lookahead blocks to provide P and G to each bit of your adder. Turn in DesignWorks schematics of your 4-bit CLA. Verify that your adder operates correctly.

- Simulate the timing of both the ripple-carry and carry-lookahead adders by setting the inputs to "1111" and "0000", and then changing the "0000" to "0001". Set all gate delays to be 10 units. How long does each circuit take to complete the calculation? Generate (and turn in) timing waveforms that show how long each circuit takes.