**CSC 325 Computer Architecture**

**Class 8 Homework**

**September 12, 2013**

Complete this work by Monday, September 16, before class time. *It has been suggested that you use a USB drive for saving with no crashing. Please report circumstances of LW crashes.*

**Exercise 1 Deriving an expression from a combinational circuit**

Derive expressions for D3, D2, D1, and D0 in the circuit below in terms of E, A, B (this variable ordering may help you think about the circuit).

D3 = ABE

D2 = AB’E

D1 = A’BE

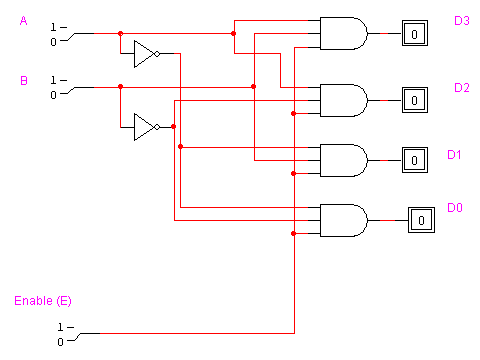
D0 = A’B’E

What is the role of E?

E is asserted high and if it is NOT 1 then all of the “and” gates fail.

Summarize the pattern you see for the A B terms in the expressions for D3..D0.

The pattern for A B is that they count from 0 to 3 in binary.



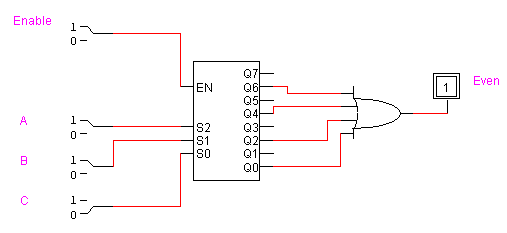
The above circuit implements a *decoder*, which is used to generate minterms. Notice that one and only one of the outputs is ever asserted, specified by the select signals A and B.

A decoder integrated circuit (IC) converts ***n*** binary inputs to a maximum 2n outputs. Its purpose is to generate the 2n binary combinations of the n input variables.

Decoders can also be used to implement canonical sum of minterms expressions. Since a product term is a minterm, and a decoder generates minterms (or product terms), we can use a decoder and an OR gate to implement a function as shown below.

F(A,B,C) = ∑m(0,2,4,6)

The circuit below uses a 3-to-8 decoder to generate the minterms (product terms) and generates the sum with the OR gate. Build the circuit in LW, selecting the Decoder-8 non-inv device from the Library. This circuit implements the even function (1 = input number is even).



**Exercise 2 Using a decoder to implement a canonical equation**

Use a 3 to 8 decoder to implement a function M that outputs a 1 when a majority of the three inputs are set. First, determine the sum of minterms expression for the function M. (Use a truth table.) Then, implement M using a decoder. (Note: no Kmap simplification is needed when using a decoder implementation of the sum of minterms expression.)

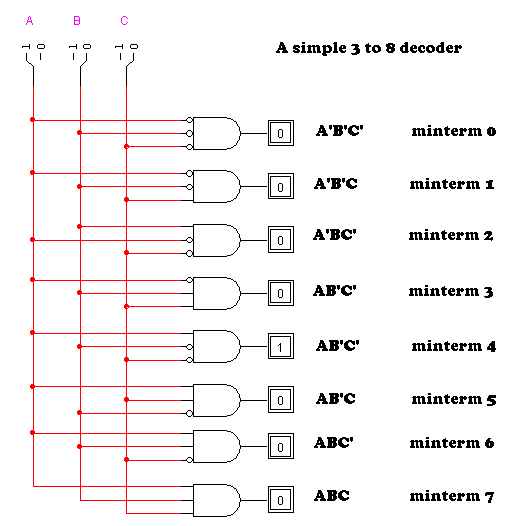
M(a,b,c) = ∑m(3, 5, 6, 7)

Test your completed majority function circuit.

**Exercise 3 Scalability of design**

Study the following circuit. What do you need to add to the circuit below to make it equivalent to the built in LW device depicted above?

You could add an enabler to the circuit for the output



LogicWorks does not have a built in device that supports a simple 4-to-16 decoder. You are to implement such a decoder with an enable. You need four inputs A, B, C, D plus an enable input, and need to generate 16 minterms. Design this circuit, build it, and test it.

**Creating a custom made device (\*\* See last page for Mac instructions\*\*)**

You can create your own library of devices. These devices will be available for you to use in your designs. The devices that you create are not to be shared with others in the course now or in the future.

To create a subcircuit for the 4-to-16 decoder:

First, save your circuit file in your H: drive or on a flash drive or in the cloud.

Replace the input switches and the output probes with port in and port out connectors, which are found in the Connect.clf library.

Label the ports (touch the pencil (click on the A) directly onto the port symbol) with the corresponding literal. Save the modified circuit file with a descriptive name. *Make certain your name(s) and the circuit title document the circuit file.*

Leaving the window with the subcircuit open in LogicWorks, click on File/New/Device Symbol OK. This opens a new window where you will draw the symbol for the decoder.

Click on the Options/Subcircuit and part type

Select the radio button for “Create a subcircuit symbol and select an open circuit to attach to it” and then select the appropriate file name in the pop up window. Click OK and then click Done. You should see the input and output literals listed in the left hand window (pin window).

Next, click on Options/Auto Create Symbol. Click on Extract Pin List. Change the part name to 4-to-16 Decoder. Then click on Generate Symbol. In this window, you can change where the pins are placed and their order. For example, all the inputs show up in the left side. You can move these as desired.

To save your device, click on File/Save As/New Lib. Then you can name your personal library. (Don’t put the device into the general LogicWorks libraries. Keep your designs in your personal library only.) Save your personal library on your USB or in your personal account. You may now close the device window. Within the libraries listing, you will see your personal library included.

Now, select the new device and test it out. If it doesn’t work, then you must start the process over.

**Exercise 4 Applied combinational design**

Using your new decoder, implement a circuit that determines if a number is less than, greater than, or equal to another number (three separate functions). Each number is two bits wide, and you may assume the numbers are unsigned.

Build your circuit and test it out. Submit your completed circuit to Canvas. Remember to include the title, etc. information. Documentation should be directly on the circuit page. (Only exercise 4 is to be submitted.)

**Just for Fun**:

Create a full adder device using your full adder circuit from Lab 2. Create a 4-bit ripple adder from the FA device and test it out.

*Advice for future work: Do not nest devices. It is tempting and LW permits it; however, each device is actually a separate procedure/module/object/etc. in the program code that LW generates for the simulation. If you build a 4 bit ripple adder out of FA devices, and then created a device from the 4 bit ripple adder in order to create an 8 bit ripple adder, and so on, LW will incur a lot of overhead. This has created some unacceptable delays in the large project implementation. So, my advice is to have no more than one level of devices.*

**Creating a device in LogicWorks on the Mac**

0.  File, Libraries, new lib or open lib (Do not save in the generic LW library)

1.  Create circuit and save file

2.  Zap the I/O devices and replace inputs with portin and outputs with portout   (found in the makePLD.clf)

3.  Label the ports by touching the tip of the pencil ("A" tool) to the inside of the port symbol

4. Click on DeviceEditor menu selection, click on sub circuit and part type, click on internal circuit, select the open subcircuit (the one with the ports)  from the menu list, click on internal, click done

5.  Click auto create symbol, click extract pin list,  type in the part name that you would like (such as TS 2x1 Mux), click generate.

6.  Click menu option File, click save part as, then select YOUR library and save the part for later use.