CSE370 Fall '99
Assignment 2
Distributed: 10/4/99
Due: 10/11/99

Reading:

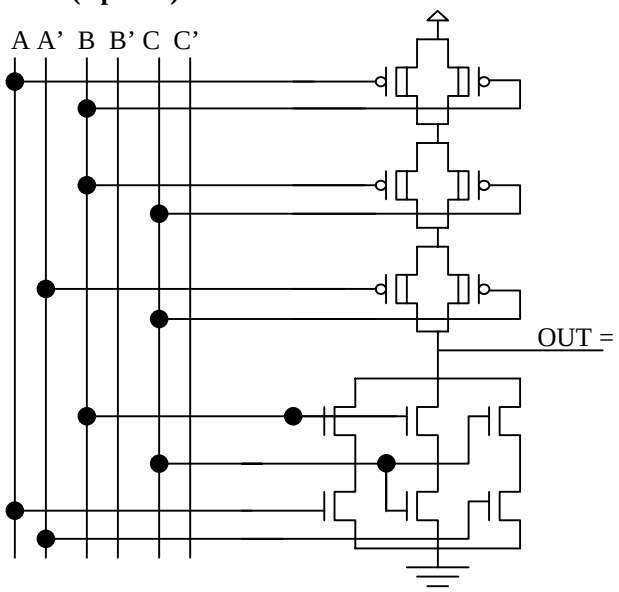
Katz, Chapter 2 (pp. 40-83)
Refer to courses/370/98au/admin/Tools/DesignWorks/DesignWorks.html for basics on using DesignWorks.

Exercises:

1. **Consider the complex CMOS gate below. (4 points)**
   a) Express OUT as a function of A, B, C by converting directly from the schematic without optimizations.

   b) Using the theorems of Boolean algebra, reduce this expression to one that contains at most 4 literals.

   c) Draw a new complex CMOS gate that directly implements the minimized expression. If each transistor requires $2\mu m^2$, how much total area have you saved?



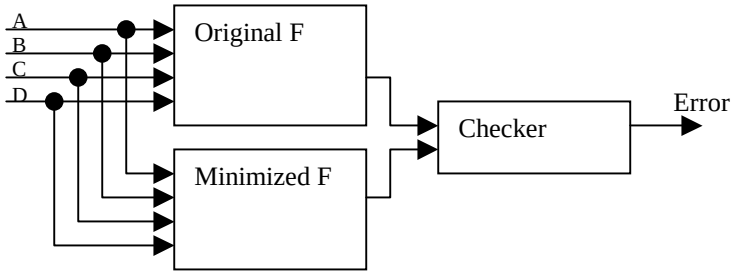
2. **Completeness (3 points)**
   a) A set of logic gates is "complete" if they can be used to implement any logic function. For example, the set of gates consisting of OR, AND, and NOT is complete. Using the axioms and theorems of Boolean algebra, prove that the NAND gate alone forms a complete alternative set. One way to do this is to show that NAND gates can be used to implement all of the functions of an already known complete set.

   b) Using the results from part a, derive an expression for the function **F=A'B+B'C** that uses only NAND operations. Note that [(WX)'(YZ)']' is read (W nand X) nand (Y nand Z). Draw the resulting NAND gate only schematic and show where there are opportunities for minimization.

3. **Simulation, Minimization, and Verification (5 points)**
   a) Using only AND and OR gates, and inverters. Create a schematic in DesignWorks (DW) that directly implements the following function: **F(A,B,C,D) = (AD+A'C)[B'(C+BD')]**. In DW, do the following:
   - Set the delay for each gate to 10 time units. See *courses/370/98au/admin/Tools/DesignWorks/Tips.html* for tips on how to set gate delay. Tip number 6 on this web page is useful for setting any delay, not just zero.
   - Create a complete set of input test stimuli for your circuit using the DW Test Vectors spreadsheet tool (refer to DW webpage under "Generating useful output"). Your input vectors should change every 80 time units.
   - Simulate your test vectors and turn in the schematic and timing diagram (waveforms) showing **ABCD** and **F**. Save your design for the next step.

b) Using Boolean algebra, find a minimized **two level** expression for F, and enter the circuit into DW. Repeat the steps above for this implementation of F.

c) Verify that the two designs are equivalent using the "expected output" feature of the DesignWorks test vector spreadsheet tool. One way to do this is to enter into the spreadsheet the expected output for every input. However, that could be a lot of work for a large design. It might be easier to design a third combinational circuit, called **CHECKER**, that simply compares the output of the two designs and generates an error signal if they are different, as shown in the figure above. The expected value for **Error** should always be "0". Using CHECKER, you should be able to verify your optimized design without even knowing the expected output for all inputs.



Turn in the gate level schematic for the entire circuit shown in the figure along with a printout of the test vector spreadsheet from DesignWorks showing that the two implementations of **F** are equivalent, along with the timing diagram for the simulation. **Is *Error* always zero in the simulation? Explain**. Note: you can build the verification schematic by cutting and pasting the original design into the new design.

4. **Hierarchical Design in DesignWorks (5)**
   a) Create your own your own DesignWorks library to contain the symbols for the circuits that you will be creating this term (right click over the "parts" window and select "new"). More info on this in courses/370/98au/admin/Tools/DesignWorks/DesignWorks.html
   b) Using the Device Editor (DevEdit) tool, create a symbol with three input pins **A**, **B,** and **Cin**, and two outputs pins **Sum** and **Carry**. The name of your new device should be **FullAdder**. Save this device in the library you just created and close the device editor. Make sure your pins have the proper directions and names. At this point, **FullAdder** is just an empty box with pins. It doesn't matter how big you make the box for the symbol, smaller is generally better. Put text in the symbol that indicates what the device is.
   c) To define the internal structure of **FullAdder** follow this procedure (more info on this in DW Help->Tech Notes->Technical Notes->DesignNote 4):
      - Create a new design (file-> new design) of type **Flat - General.**
      - Place the **FullAdder** from your library onto the new schematic page
      - Push into the part by double clicking on it. A dialog will appear indicating that the part has no internal circuit and prompting you to create one, answer OK.
      - A new circuit window will appear with all the pins in place. You can move the pins around to make room for your circuit. Now create your internal circuit with gates and wires using the Katz figure 5.7 as your model.
      - When the internal circuit is complete, close the window by selecting "Pop-up" from the right-click menu or click the close box on the top left corner of the circuit window.
      - From the "Part Type" sub menu in the "Options" menu, select "Save to Lib...".
      - Select the same library and part for **FullAdder**. Say yes to overwrite. You now have a complete **FullAdder** library part that you can use for the next step.
   d) Create a schematic for the 4-bit adder design of Katz figure 5.6 using your **FullAdder** device. Note that the Cin on the first **FullAdder** is not used, so tie it to ground.
   e) Use two "Hex Keyboard" devices from the *primio* library for the A and B inputs to your 4-bit adder, and one "Hex Display" device for the 4-bits Sum output. Finally, hook C_4, the highest

carry out, to a "Binary Probe" device. Turn in a copy of your schematic showing the results of $A_{16} + 7_{16}$.

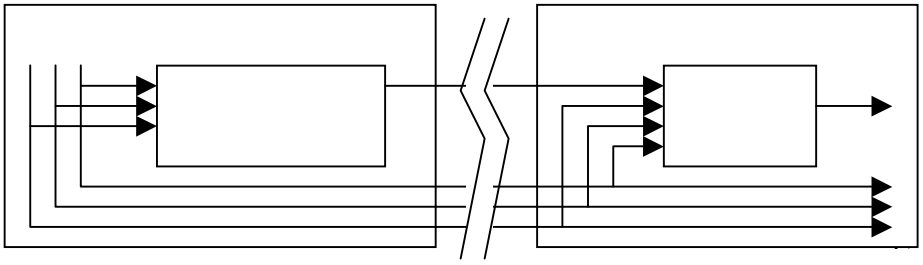
5. XOR Gates (3 pts)

a) Write a sum-of-products expression for the parity function, **P**, on four inputs, which is defined as follows:

P(A,B,C) = 1 if and only if an odd number of inputs are 1

b) Implement **P** using only 2-input XOR gates. Explain how you arrived at your design.

c) Suppose you are going to transmit signals **A,B,C** over a long noisy set of wires that are prone to transmission errors. If you also transmit **P** over a 4th wire, then the receiver can detect an error on one of the four bits. Using a truth table, or other means, define the error function **E** so that **E(A,B,C,P) = 1** if a single bit error has occurred, and **E=0** if no errors have occurred.



d) Show how this function can be implemented using only 2 input XOR gates. Draw your schematic.