CS/EE 5830/6830 -- CAD Assignment #2

Due Tuesday, February 8th , 11:59pm
Use electronic handin to the CAD2 assignment

For this assignment you will turn in your entire Cadence project directory as a tar file using the CADE lab handin program. There are instructions on how to use both handin and tar on the class web site. Note that this assignment name is CAD2 so to handin a file you would use the following command on a CADE machine:

handin cs6830 CAD2 <filename>

1. Start with your full-adder bit from CAD1. Create a test schematic in which to simulate a single full adder bit with the Spectre analog simulator as described in Chapter 7 of the CAD book. Assuming that you used the UofU_Digital_v1_2 cells, they will have the appropriate transistor level (cmos_sch) views for this analog simulation. Remember to use a vdc between vdd and gnd with a DC voltage of 5 v in your test schematic. Drive the inputs to the full adder from vpulse, vpwl, or vpwlf voltage sources in such a way that you get an exhaustive test of the full adder circuit. In order to make a slightly realistic load for your adder cell, use an INVX4 connected to each output (S and Cout) of your full adder, but plot the actual outputs of your adder not the output from the inverter. Look at Figure 7.4 for an example, but use an INVX4 as a load instead of the capacitor used in that figure. Plot the input and output waveforms. Note in the README file that you submit what the name of the plot file is.

   Note that in V6 of the Cadence tools that we're using this semester, you fire up the analog simulator with Launch->ADE L *from the schematic tool's menu.*

2. Now take your 4-bit ripple adder from CAD1. Simulate your four-bit adder using NC_Verilog using the following timing techniques (see Chapters 4 and 7 for details). Test the four-bit adder for functionality either by testing exhaustively or by considering a set of regular and corner cases. In the second case be sure to justify why your corner cases are sufficient to test the adder. You can use the same testbench for all simulations.

   a. Behavioral unit-gate-delay timing. In this mode you should netlist to the **behavioral** views of the UofU_Digital_v1_2 cells where each gate has a single unit of delay (through the **specify** blocks that are in the behavioral views).
   b. Switch-level unit-transistor-delay timing. In this mode you should netlist to the **functional** views of the UofU_Digital_v1_2 cells where you netlist all the way to transistors. The transistors have 0.1 units of delay each. Look for the **stopping view** comments in the CIW to make sure you're netlisting to the right level.
   c. Mixed-mode analog-digital simulation. That is, use Spectre to simulate the four-bit adder at the analog level, but use Verilog to provide the testbench as described in Section 7.4 of the CAD book. Make a new schematic with pairs of inverters on all inputs and outputs of an instance of your four-bit adder (see Fig 7.16 in the CAD book for an example). Also include a vdc voltage source from

the NCSU_Analog_Parts library with the DC voltage set to 5 v. Use the same Verilog test fixture as in the previous tests to simulate the four-bit adder with Spectre, but apply the inputs digitally through the Verilog test bench. You will need to create a **config** view of this test schematic. To make the inputs to your four-bit adder more realistic, make the inverter in the pair that's closest to the four-bit adder analog (view **cmos_sch** and inherited view list **spectre**), and the inverter closest to the input/output pins digital (view **behavioral**, inherited view list **$default**).

Where the book says to use Tools->Mixed-Signal Ops*, you should use* Launch->MixedSignalOptions -> Verimix.

Make sure to change the Verimix -> Interface Elements -> Default Options *to have* a Default IE Library Name *of* NCSU_Analog_Parts.

Launch the Analog Design Environment with Launch -> ADE L. *Remember to change the simulator to be* spectreVerilog *in the* Setup -> Simulator/Directory/Host *option.*

Important note about mixed-mode analog-digital simulation with SpectreVerilog: this currently doesn't have a 64-bit binary, so you need to force it to use the 32-bit binary. In the Analog Design Environment window, choose the Setup->Environment *option. In the* Environment Options *window, find the* userCmdLineOption *field and put* -32 *in that field. This will pass the -32 switch to the tool and force it to use the 32-bit binary.*

You'll need to follow along carefully with section 7.4 in the CAD book. The dialog boxes look a little different in the V6 tools, but they're basically the same in terms of function. Note that the new config view dialog box has both a Table View and a Tree View tab which allows you to quickly switch the views. The Tree view is where you'll make the changes in the View to Use and Inherited View List.

3. Compute the worst-case and best-case timing of your four-bit ripple adder from all three methods: behavioral, switch-level, and analog (using the mixed mode simulator to apply patterns). You should be measuring the four-bit adder itself (i.e. without the extra inverters). Put this information in a table and make sure that it's handed in with the project library. You can look at the timings from the previous simulations, or you can simulate the specific cases that you think will be the best and worst cases for timing.

4. Turn in the following through handin:
    a. A tar file of your CAD2 project library
    b. A tar file of your CAD1 project directory if you used instances from that library in your CAD2 schematics.
    c. A waveform of the analog simulation of the adder bit
    d. Testbench code in Verilog for testing the four-bit adder

e. A table of measured worst-case timings for all three timing techniques
f. A README file that tells us what file names to look for

5. From the book, do the following problems:
    a. 1.6, 1.10, 1.17, 1.22 for left shift (not right shift), 1.23 but compute x= [(-a + 4b) + 2c – 4d] (extend the number of bits in the arguments as needed), 1.25 (two's comp only, and multiply 19 by –12 instead), and 1.29 (except divide 13 by 5). Turn this using handin, or in the slot in the homework box outside the CS office.
        i. Clarification of 1.17b: Part 1 - Assume you have a 2's complement adder/subtractor. Will that hardware work for unsigned add and sub? Which of the flags will be computed correctly for the unsigned operations and which will be computed incorrectly? Part2 – Assume you have a 1's complement add/sub. Will that hardware work for unsigned add/sub? Which of the flags will be computed correctly for the unsigned operations and which will be computed incorrectly? For each part, for each operation/flag that you say will be incorrect for the unsigned case, give an example of the error.
        ii. Clarification of 1.17c: Answer the following questions for part c – 1: assuming you have a 2's complement add/subtractor that you're using for the comparison operation (i.e. the subtraction) How would you compute greater, equal, and less-than (i.e. combinations of flags) for 2's complement arguments and for unsigned arguments? 2: same question but for a 1's complement add/sub unit and the same flags.