EE 4702 Homework 6 Due: 28 April 2000

Solution templates can be found in `/home/classes/ee4702/files/v` *and* `www.ee.lsu.edu/v/2000/hw06.html` *Put your solution in a file named hw06sol.v. Soon after the time the assignment is due your directory tree will be searched for files named hw06sol.v and the most-recently modified one will be copied. If no such file is found an attempt will be made to copy a file using a guessed name, but this is not something to be relied on. Give the file the correct name.*

Solutions to the problems below should be synthesized for the following technology: ASIC (type of target), Sample (manufacturer [usually]), XCL05U (technology family). Do not specify any optimization or other synthesis options. View the RTL schematic to check your solutions. (Under the Tools menu or using the toolbar button.) Leonardo is started by typing `leonardo &` *in a shell. To work around a cosmetic stdout bug start Leonardo by typing* `leonardo > /dev/null &` *. See the procedures and FAQ web pages for additional instructions on running Leonardo.*

See the FAQ page for instructions on how to write Verilog code of the synthesized module for simulation using the Leonardo GUI, a TCL script, or Emacs. The process is particularly convenient using Emacs.

The assignments will be graded under the assumption that the synthesized code was simulated using the testbench provided; a substantial number of points will be deducted for solutions that do not pass the testbench correctly.

Problem 1: A Verilog behavioral description of a module similar to the one described in the first midterm problem appears in `http://www.ee.lsu.edu/v/2000/hw06.v`. The module cannot be synthesized by Leonardo (1999.1f). Modify the module so that it can be synthesized while retaining the benefits of behavioral code. (That is, do not convert it to a structural description.) The synthesized code must pass the testbench provided with the code.

The module, `width_change`, describes a FIFO in which data is inserted in 4-bit *nibbles* (the technical term for half a byte, no kidding) and removed in 8-bit bytes. The total storage capacity is 32 bits. In addition to the 4-bit input and 8-bit output there are two 1-bit inputs, `inclk` and `outclk`. On a positive edge of `inclk` data is read; on a positive edge of `outclk` data is removed in FIFO fashion. The low nibble of the output (bits 0-3) holds data that arrived earlier than the high nibble of the output. Output `full` is one if the FIFO cannot accept another nibble, output `empty` is one if the FIFO is empty (in which case the output must be all zeros), and output `complete` is 1 if the output has eight bits of data. (Output `complete` is zero if the FIFO is empty or if it contains just 4 bits. If the FIFO contains four bits the high nibble of output should be zeros.)

Note that, unlike the test question, the sizes of the input, output, and storage capacity are digital-logic-friendly powers of two. However like the midterm exam, the input and output each have their own positive edge triggered clock. Getting this into synthesizable form will take some thought.