LAB Assignment #5 for ECE 522

Assigned: Mon., Oct 21, 2015
Due: Mon, Nov. 4, 2015

Description: Write a new VHDL module called EvalFUOutputs that generalizes what you have done in lab4. Simulate as necessary and prepare a hardware demo.

1) LCTest_Driver.vhd should be simplified to only start/stop/synchronize three state machines, *PhaseAdjust*, *LaunchCapture* and the new module *EvalFUOutputs*. Every time the *LCTest_Driver* is started with a new pair of LaunchVals, you should carry out two tasks: First *LCTest_Driver* should start the *PhaseAdjust* engine to run the phase out to 150. It should then start *EvalFUOutputs* to carry out initialization tasks prior to the testing of the FU's outputs (see below). Once initialization finishes, you should enter a loop in *LCTest_Driver* that carries out a series of launch/capture tests, once for each phase shift value. Here, you start the *LC* engine, followed by starting *EvalFUOuts* again but in launch/capture mode (see below) and then, if necessary start the *PhaseAdjust* again if there are still paths that need to be timed (see below). Note that initialization is only performed ONCE, while the launch/capture will be repeated over and over again until all FU's outputs are timed. Note that you need to 'parameterize' your VHDL to handle any number of outputs (by changing a constant in the provided *DataTypes_pkg.vhd*) and either type of transition, i.e., the tested path transitons from '0' to '1' (as it does in lab4) or from '1' to '0'.

2) *EvalFUOutputs* can be started in one of two modes:
a) Mode 1 (Initialization): During initialization, it should carry out special launch/capture test that allows a full clock cycle for the second vector values to propagate to the FU outputs (an example of this is already provided in lab4). During this special test, you should first record the FU's initial values under the first vector and then determine which of the FU outputs change under the second vector (after waiting a full clock cycle). Create a set of single bit registers (width corresponding to the number of outputs from the FU -- see *DataTypes_pkg.vhd* for *OutRowType* typedef), called *init_FU_vals*, that stores the initial values of the path and a second set of single bit registers call *has_trans* that stores a '0' if the corresponding path does NOT have a transition and a '1' if it does. You should also initialize the '*row_timings'* 2D register array to all '0's (see *RowResultsType* provided *DataTypes_pkg.vhd* -- first dimension refers to the FU path output, second dimensioin refers to the timing value (phase shift value)). Last, initialize a second set of single bit registers (width also corrsponding to the number of outputs from the FU), called '*path_is_timed'* and initialize it to all '0's.

3) Mode 2 (Launch/Capture): During launch/capture mode, inspect the capture FF values (Capture_vals) for each path (one at a time) to determine if the path succeeded in propagating its transition to the capture FF in the time allotted by the current value of the phase shift. This can be determined by xor'ing the Capture_vals(x) with the corresponding *init_FU_vals(x)* that you initialized as described above. If the values are different, you should update *path_is_timed(x)* with a '1' and store the current phase shift value in *row_timings(x)*. The *path_is_timed* status registers should be used to prevent further updates to *row_timings* for those paths that are been timed. Also, you should skip the inspection and update to *row_timings(x)* of paths that have no transitions (*has_trans(x)* stores a '0'), i.e., leave their *row_timings(x)* values at 0. Also, optionally, you can

keep updating the timing values for paths that have NOT yet propagated their transition with the current phase value. Once the transition arrives, you would stop updating the timing value (as described above) because the *path_is_timed(x)* bit would be set. This allows you to handle overflow properly, i.e., paths that are too long and therefore, the transition never makes it to the capture FF even with the largest phase shift value (1100 in our case). Also, optionally, you can create a set of single bit registers called *glitches* that records whether a path has more than one transition. This information can be used later to filter paths that are 'unstable', i.e., those with more than one transition. Last thing to consider is the 'done' condition. LCTest_Driver.vhd has a signal called *all_timings_done*, which should be asserted in *EvaluFUOutputs* when all paths with transtions have been timed.

Laboratory Report Requirements:

Grading:
The grading from this lab will be based entirely on your in-class demo. Bonus points will be given to any implementation feature that goes above and beyond the requirements. Please print out and turn in a copy of your *EvalFUOutputs* code.