LSU EE 4755 **Homework 4** **Due: 12 October 2015**

Problem 0: Follow the instructions for account setup and homework workflow on the course procedures page, `http://www.ece.lsu.edu/koppel/v/proc.html`. Run the testbench on the unmodified file. There should be errors on the `shift_lt_seq_d_sol` module, but the others should run correctly. Run the Note: There are no points for this problem.

Problem 1: The homework Verilog file, `hw04.v`, contains a module `shift_lt_seq_d_sol` which is based on `shift_lt_seq_d`. It contains an `always_ff` block that assigns the same variables that are assigned in `shift_lt_seq_d`, however it assigns them from variables of the same name with `next_` prefixed:

```
always_ff @( posedge clk ) begin
   ready = next_ready;
   shifted = next_shifted;
   shift = next_shift;
   cnt = next_cnt;
end
```

Add code so that these `next_` objects will be assigned values from combinational logic, and so that the resulting module describes the same hardware as `shift_lt_seq_d`. A hand-drawn diagram of synthesized hardware should be identical, though it's possible that there will be small differences in the actual output of a synthesis program.

The added code can be implicit structural or behavioral, but it must synthesize to combinational logic.

Problem 2: Module `shift_lt_seq_d_live` takes one more cycle to produce a result than module `shift_lt_seq_d`. Module `shift_lt_seq_d_p2` initially is identical to `shift_lt_seq_d_live`.

(*a*) Modify `shift_lt_seq_d_p2` so that it uses one less cycle to produce a result without changing the number of shifters per stage. There are two possible ways of doing this, performing some work in the same cycle that the `start` signal arrives, or doing work in the cycle when `ready` is set to 1. Either method is fine.

(*b*) Run `syn.tcl` and compare the cost and performance of your design and `shift_lt_seq_d_live`. Comment on the differences. An answer might start *"The cost was about the same because the same hardware was used..."*.