TCES 330, Spring 2020

Project A: Thunderbird Tail Light

General Comments
This is a one week team project on which discussion started in class. You will work in teams of maximum 3 members; individual submissions will not be accepted. Ideally a team has both EE student(s) and CES student(s), but this is not mandatory. The submission for this project will consist of a Quartus project, your ModelSim files, and a written report (more details are given later). Your written report should follow the required format.

Make sure that all team members share proper amount of workload for this project as equally as possible. Each member should describe your contribution in the report; each member should read the report before its final submission and sign the cover page to show your consent on the content. The work you turn in must come from your group only. Do not share your work with other groups.

Problem & Requirements

- In short, the project implements the thunderbird tail light design. You will describe the design using SystemVerilog, and test your FSM design using ModelSim. Finally you implement the design using DE2-115 board.

- Starting by sketching a state transition diagram on paper, followed by developing a module named `TaillightsFSM.sv` using SystemVerilog which fitting the following problem description: An older model Ford Thunderbird has three left and three right taillights which flash in unique patterns to indicate left and right turns. You will design a **Moore machine** to control these lights. The circuit has three inputs L (for Left), R (for Right), and H (for Hazard). L and R come from the driver's turn signal switch and cannot be 1 at the same time.

  - When L=1 the lights La, Lb, Lc flash in the following pattern: La on; La and Lb on; La, Lb, and Lc on; all off; and then the sequence repeats.
  - When R=1 the lights Ra, Rb, Rc flash in the following pattern: Ra on; Ra and Rb on; Ra, Rb, and Rc on; all off; and then the sequence repeats.
  - Position of tail lights: Lc   Lb   La   Ra Rb Rc
  - If a switch from L to R (or vice versa) occurs in the middle of a flashing sequence, the circuit should immediately go to the IDLE state (all lights off) and then start the new sequence.
  - H comes from the Hazard switch, and when H=1 all six lights flash on and off in unison. H takes precedence if L or R is also on. If H switches from 0 to 1 while in any state other than IDLE, the circuit should switch to IDLE first.
  - The desired flashing rate is 1 second.

- Verify your design by checking FSM viewer under Quartus. Take a screenshot, and name the picture `taillightsQuartus.XXX` (any picture format should be fine).

- Develop a testbench named `TaillightsFSM_tb`. Verify your design by ModelSim and take a screenshot of the FSM under ModelSim FSM view. Name the picture `taillightsModelSim.XXX` (any picture format should be fine).

- Reserve the name runlab.do or runrtl.do to run the testbench of your FSM module, with a line that calls the appropriate **wave.do. Using both waveform and data viewing in the transcript window to verify your design. Missing of relevant macro .do files in the submission will get penalized.

- Your Quartus project should be in a folder called **`ProjectA`** and your top-level module should also be called **`ProjectA.sv`**.

    - Your top-level module will instantiate your FSM module (`TaillightsFSM.sv`) and other submodules (by your own design); the top-level module also interfaces to the DEII board.
    - All the modules need to be properly commented!
    - Taking screenshots of the RTL viewer after project being compiled to show the hierachchy of your project construction.
    - Properly dealing with those warning message during the project compilation process. A screenshot of the final warning messages is required.
    - Describe your test mechanism in the report (e.g. using switches 1-3 for left light etc).

- A demonstration of your project on DEII board is required. You can either make a video record and provide the link, or ask for a Zoom demonstration. No video clips inside your zipped folder!

When you are done with your project.

Zip the contents of the project folder and your written report into a zip file (the exact name of the zip file should contain the name of at least one of your team members, as an example, **`A_B_C_ProjectA.zip`**). When your zip file is opened, we'd like to see a top-level folder called **`ProjectA`**, plus the report. Make sure that you turn in all files necessary for the project and that these files are in the proper locations for compilation. It's a good idea to test this by unzipping your files to a new location.

Check the rubrics when your group divide the work and prepare the submission. Make sure your package has included all the files that satisfying the specifications/requirements.

Submit the complete package on Canvas by the due time.