The George Washington University
Depart of Electrical and Computer Engineering

ECE 6213
Fall 2023
Dr. Matthew LaRue

Project 2: Mustang Tail Light Sequencer

Step 1: Create a project 2 directory with the same directory structure as project 1. Copy over all the scripts from the project 1 directory.
Hint: navigate to the `~/ece6213` directory, and then use the command "`cp -rv project1 project2`" to copy over the directories and scripts all at once

Step 2: Create a Verilog module for a Mustang tail light sequencer using the Verilog coding style presented in lecture 2.

Example tail light sequence video:
https://www.youtube.com/watch?v=7SyWdmnsMfQ
Note that the video has a slightly different light sequence, but I am providing it to show the concept, see functional requirements for actual sequences needed

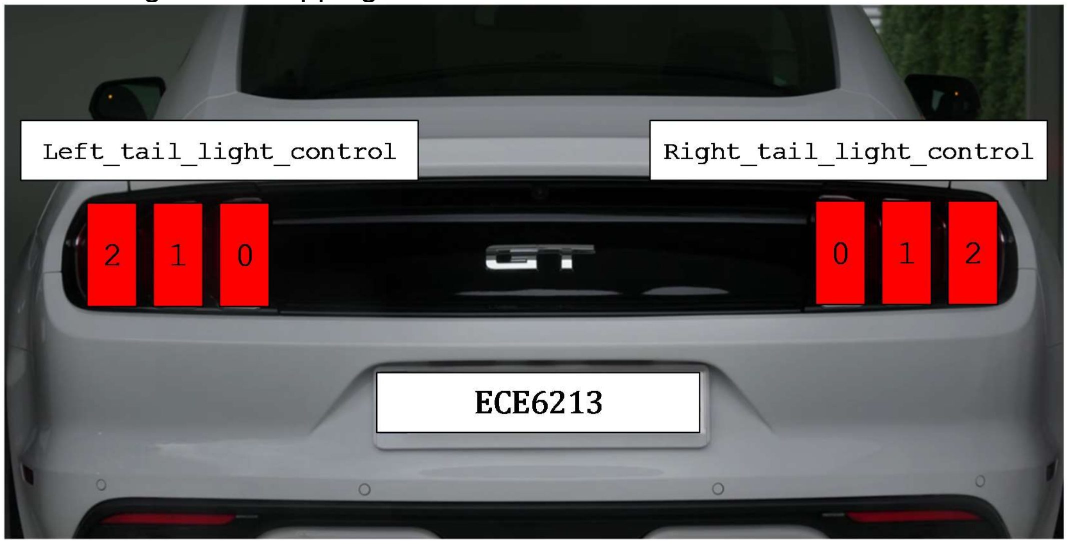
Module Inputs:

clk	100 MHz clock signal
rst_n	active-low asynchronous reset
brake	signal from brake pedal
turn_right	signal from right turn indicator
turn_left	signal from left turn indicator

Module Outputs:

right_tail_light_control[2:0]	signal to turn on right taillight bulbs
left_tail_light_controll[2:0]	signal to turn on left taillight bulbs

Output control signal bit mapping:



<u>Functional requirements:</u>

1. When only turn_right is active, right_tail_light_control[2:0] must illuminate in a repeating {001} {011} {111} {000} sequence, with each state lasting five clock cycles
2. When only turn_left is active, left_tail_light_control[2:0] must illuminate in a repeating {001} {011} {111} {000} sequence, with each state lasting five clock cycles
3. When only brake is active, all tail light must illuminate
4. When turn_right and brake is active, all left_tail_light_control lights must illuminate and right_tail_light_control[2:0] must illuminate in a repeating {111} {110} {100} {000} sequence, with each state lasting five clock cycles
5. When turn_left and brake is active, all right_tail_light_control lights must illuminate and left_tail_light_control[2:0] must illuminate in a repeating {111} {110} {100} {000} sequence, with each state lasting five clock cycles

Step 3: Create a Verilog testbench and simulate your design to verify that it meets all of these requirements

Step 4: Create and submit a datasheet for the controller, an example datasheet is provided on blackboard. At a minimum, be sure to include:

1. Input/output signal names, description, polarity (active high/low)
2. Functional description of controller
3. Simulation waveforms demonstrating all functional requirements are met
    a. Make sure the waveforms are legible
4. Controller Verilog code
5. Controller testbench Verilog code