ECEN2350 Digital Logic – Project 3

Fall, 2020

Due Wednesday, November 25th, 2020, end of day

110 points

Everyone is responsible for turning in their own project. You may work with others to complete the lab. **Read the ECEN2350 Digital Logic Project Guidelines Spring 2020 for instructions regarding project submission.**

This lab will require you to code and compile the design using iVerilog and Quartus, simulate the design using iVerilog and GTKWave, and submit a working project and a lab report. The goal of Lab 3 is to extend your design knowledge to finite state machine (FSM) design techniques, and to create a testbench that simulates this FSM design.

Upon completion of this lab, you will be familiar with FSM design coding techniques, and instantiation and usage of memory blocks contained within the MAX10 FPGA.

Note: There will be no weekly submissions required for Project 3. This is a change from the published class Syllabus. Because of the removal of additional submissions, the due date for this project is now November 25th.

Lab Description:

Lab3 consists of a finite state machine design that emulates the behavior of taillights of a 1965 Ford Thunderbird automobile.

https://www.youtube.com/watch?v=Qwzxn9ZPW-M shows how the taillights operate. This project will emulate hazard and turn signal operation including the effects of pressing the brake pedal. We not include the effects of turning the headlights on and off.

You should watch this short video multiple times to understand the correct operation of the tail lights.

You should read the remainder of this document before beginning to design.

Grading

Your DE10-Lite project will count as 80 points of the total Project 3 grade. The 80 points will be graded as follows:

1.	Successful completion and submission of base design	50 points
2.	Automation of design using MAX10 memory block	15 points
3.	Successful completion and submission of testbench	15 points

If you are unable to complete the base design, submit what you have for partial credit. Your lab report should fully explain the final state of your project.

Note: The lab report is mandatory. No Lab3 credit will be given if the lab report is not submitted. Document as much of the lab as you were able to complete. You may not submit a lab report that you did not write yourself.

Guidelines for lab report:

Follow the instructions in **ECEN2350 Digital Logic Project Guidelines.** The only addition specific to Project 3 is that you should include a state bubble diagram, with the states labelled to match your design, and a description of how the state machine operates. As a state diagram can get messy if you include all state transition information, you can augment the diagram with a paragraph that explains the operation.

Detailed Requirements for Lab3 (Read carefully, get clarifications early)

1. You must code this lab as a FSM using separate current state, next state, and output logic blocks. These blocks can be separate Verilog modules, or simply separate groups of code. Use meaningful names to reference the states in your design through the use of a separate parameter file.

2. Create a display update frequency that allows for easy viewing of the circuit operation.

3. Design the user interface as follows:
    a. Use KEY[0] as system reset.
    b. SW[0] is the hazard light switch, UP / ON turns on the hazard lights, DOWN /OFF turns off the hazard lights.
    c. SW[1] is the turn signal enable switch, UP / ON turns on the turn signals, DOWN / OFF turns off the turn signals.
    d. When SW[1] is UP, the left turn signals will operate when KEY[1] is not pressed, and the right turn signals will operate when KEY[1] is pressed .
    e. SW[2] UP will simulate pressing the brake pedal.
    f. Use any hex display (or displays, if necessary) to display the numeric value of the current state register.
    g. All unused LEDRs should be turned completely off.
    h. The hazard indication will blink LEDR[9:7] and LEDR[2:0] simultaneously and at the same flash rate as the turn signal lights.
    i. The hazard condition has higher priority than turn signals.
    j. If you have any questions, watch the video.

   Here is further clarification about the behavior of the design.
   1. The hazard switch has the highest priority, overriding both brake lights and turn signals.
   2. The brake pedal switch will turn on all 6 taillights when the turn signals are off.
   3. If a turn signal is operating, and the brake pedal switch is turned on, the selected turn signal will continue to operate normally, and the brake lights on the opposite side will turn on.

4. Complete the design described above, where you manually manipulate the switches to operate the design. Your testbench simulations will simulate the design to this point. <u>You will not be able to simulate the automated operation described below</u>.

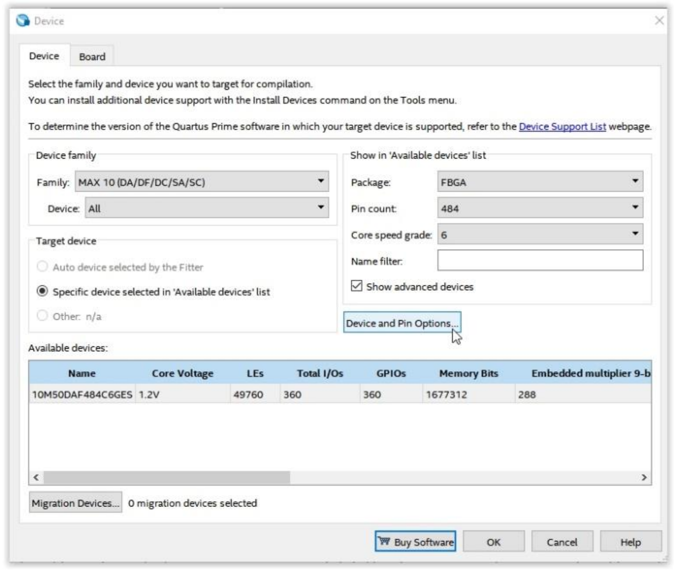
5. Once step 4 is finished and functioning properly, add a MAX10 M9K memory block to your design. Based on the state of SW[9], the design will run normally using manual manipulation of the switches (SW[9] DOWN), or will be automatically cycled through the states (SW[9] UP) by using M9K outputs to emulate the switches.
    a. When SW[9] is UP, the display will spend approximately 3 - 5 seconds in each of the following states, running in a continuous loop.
        i. All lights off
        ii. Hazard lights flashing
        iii. Left turn signal lights flashing (brake off)

iv. Right turn signal lights flashing (brake off)
v. Left turn signal flashing (brake on)
vi. Right turn signal flashing (brake on)
vii. Back to i.

b. We will discuss the steps to create and instantiate the MAX10 memory block in class.
c. You will create a new top level module to add the automation, it will not be necessary to modify any of your existing files.

You must make this settings change to Quartus to get the automatic mode to compile properly:

Go to Assignments > Device, and click on the Device and Pin Options button.



Select the family and device you want to target for compilation.
You can install additional device support with the Install Devices command on the Tools menu.

To determine the version of the Quartus Prime software in which your target device is supported, refer to the Device Support List webpage.

Device family
Family: MAX 10 (DA/DF/DC/SA/SC)
Device: All

Show in 'Available devices' list
Package: FBGA
Pin count: 484
Core speed grade: 6
Name filter:
☑ Show advanced devices

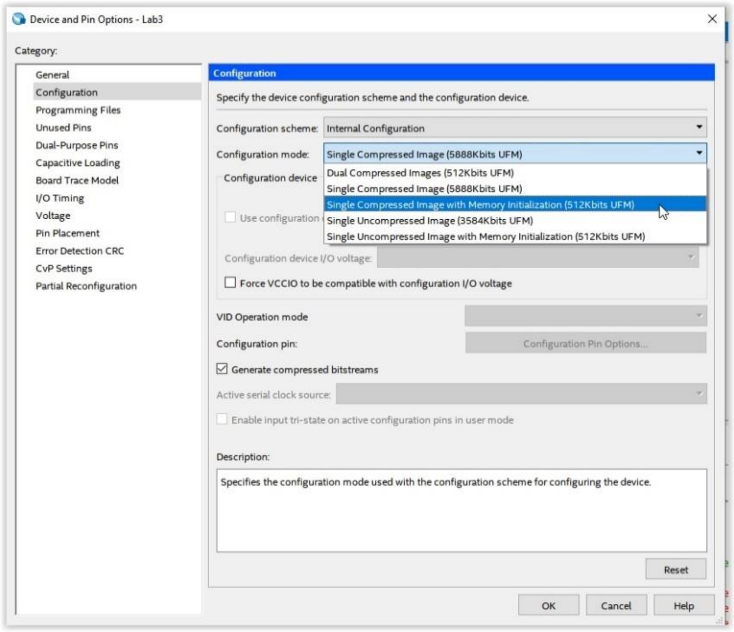
Target device
○ Auto device selected by the Fitter
● Specific device selected in 'Available devices' list
○ Other: n/a

Available devices:

Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier 9-b
10M50DAF484C6GES	1.2V	49760	360	360	1677312	288

Migration Devices... 0 migration devices selected

Select the Configuration Category, then using the Configuration mode drop down, select 3rd item “Single Compressed Image with Memory Initialization..”.


Device and Pin Options - Lab3
Category:
General
Configuration
Programming Files
Unused Pins
Dual-Purpose Pins
Capacitive Loading
Board Trace Model
I/O Timing
Voltage
Pin Placement
Error Detection CRC
CvP Settings
Partial Reconfiguration
Configuration
Dual Compressed Images (512Kbits UFM)
Single Compressed Image (5888Kbits UFM)
Single Compressed Image with Memory Initialization (512Kbits UFM)
Single Uncompressed Image (3584Kbits UFM)
Single Uncompressed Image with Memory Initialization (512Kbits UFM)
Configuration device
Use configuration
Configuration device I/O voltage:
VID Operation mode
Configuration Pin Options...
Active serial clock source:
Enable input tri-state on active configuration pins in user mode
Reset
OK
Cancel
Help

Specify the device configuration scheme and the configuration device.

Configuration scheme: Internal Configuration

Configuration mode: Single Compressed Image (5888Kbits UFM)

Force VCCIO to be compatible with configuration I/O voltage

Configuration pin:

[✓] Generate compressed bitstreams

Description:

Specifies the configuration mode used with the configuration scheme for configuring the device.

Click OK. You only have to make this settings change one time.

Testbench Requirements

1. **Simulate the design before adding the additional logic that automates the design using the M9K memory block. There is no expectation that the automated design is simulated.**

2. You must create a testbench that demonstrates aspects of the design. At a minimum:
   a. Simulate the behavior of the state machine by changing inputs, showing the state values.
   b. Simulate the behavior when in hazard mode, left and right turn signal mode, and when the brake pedal is both pressed and not pressed.

3. Your simulation output must include waveform captures. You may include simulation text output if you want. Include the simulation images in your project report with appropriate descriptive text.

4. Label your states in GTKWave. The video GTKWave Tricks for more readable simulation output describes how to use unique labels in GTKWave to make the waveform images more understandable.

5. You may have one or multiple testbench files. It is likely you will need multiple screen captures to demonstrate operation of all possible behaviors.