Lab 9: Sequential LED Tail Light

ECE 211: Digital Circuits I — Spring 2025

Abstract. In this lab, you will build a circuit that controls the turn signals on a car. Recent models of the Ford Mustang have featured sequential tail light designs that flash from the inside outward, paying homage to the Shelby Mustangs of the late 1960s. An example of this sequential tail light is shown in the images below. This video shows the tail lights in action on a recent model of the Ford Mustang: *www.youtube.com/watch?v=Wsgdh2L8t28*. If you prefer classic cars, sequential tail lights were first featured on Ford Thunderbirds in 1965: *www.youtube.com/watch?v=Qwzxn9ZPW-M*.

This tail light feature can be implemented using a finite state machine that takes in an input to control whether the turn signal is off or off. In this lab, you will design a finite state machine that implements this specification, prototype and test the circuit in SystemVerilog, and then synthesize the design on the Nexys A7 FPGA.

Objectives.

- Gain experience designing sequential circuits using finite state machines
- Test the operation of individual circuit components using testbenches
- Test the operation of the circuit using the input and output components on the Nexys A7

Materials and Equipment.

- Nexys A7-100T FPGA board with USB cable

Deliverables.

- In-person lab demonstration (per team)
- Lab report (per team)

Part 0: Set Up a Vivado Project

To begin the lab, we will set up a new Vivado project for Lab 9 and configure the Nexys A7 board. You can reference Lab 1 for a more detailed description of these steps.

1. Create a Xilinx Vivado project and import the provided constraints file and SystemVerilog modules. This lab will use both simulation and design sources. The file **`lab09_top.sv`** should be uploaded as a design source. The file **`lab09_tb.sv`** should be uploaded as a simulation source. You will use this file as a testbench to write test cases for each module.

2. In this lab, you will primarily use Vivado's simulator to verify your design at each part, and then use the FPGA board at the end to demonstrate the entire circuit. If you would like, you can connect your FPGA board and open the hardware target at this point in the lab.

Part 1: Design a Finite State Machine (FSM)

A ***finite state machine (FSM)*** is used to design and implement sequential circuits. In the first part of this lab, you will design a finite state machine that implements the behavior of the Mustang tail light. Figure 1 shows the blinking pattern of a single tail light, where `I` is the inner LED segment, `M` is the middle LED segment, and `O` is the outer LED segment. The figure illustrates the right turn signal. Your finite state machine should use a single input, dictating whether or not the turn signal is on, and have three outputs, one for each LED segment. When implementing your finite state machine as a sequential circuit, your SystemVerilog module will take other inputs, including a reset and clock. Your final sequential circuit will operate off a low-frequency clock that matches the desired flashing rate of the tail lights.

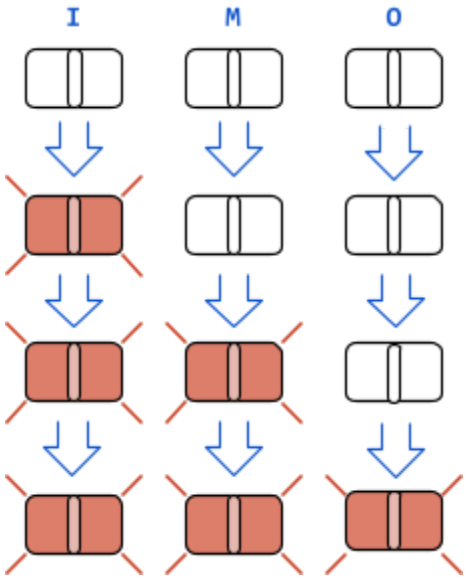

Figure 1. Tail Light Flashing Sequence (Right Turn Signal)

3. Sketch a state transition diagram that implements the tail light specification.
   a. Determine the number of unique states in the tail light problem
   b. Add these states to the state transition diagram
   c. Draw transitions between each state depending on the input signal, dictating whether or not the turn signal is on

4. Create a truth table for your state transition diagram.

5. Derive Boolean equations for each bit of *next state* and the LED *outputs* `I`, `M`, and `O`.

6. Sketch a schematic of your resulting circuit.

Record: Record your state transition diagram, truth table, Boolean equations, and circuit schematic in your Lab Report.

Part 2: Implement and Test your Sequential Circuit

In this part of the lab, you will implement your sequential tail light FSM as a SystemVerilog module. Your module will take in a clock signal, a reset, and a control switch to dictate whether the tail light is on (blinking) or off. The current state of your FSM will be stored using D flip-flops. In SystemVerilog, this functionality is implemented with the **`always_ff`** block. The next state of your FSM can be computed using combinational logic. In SystemVerilog, this functionality is implemented with the **`always_comb`** block. An example of a finite state machine is shown in Figure 2.

```
module example_fsm(  input  logic rst, clk
                     output logic [1:0] current_state);

   logic [1:0] next_state;

   // Update state on rising clock edge
   always_ff @(posedge clk)
   begin
     if(rst)
         current_state <= 2'd0;
     else
         current_state <= next_state;
   end


   // Compute next_state with combinational logic
   always_comb
   begin
     if(current_state == 2'd0)
         next_state = 2'd1;
     else if(current_state == 2'd1)
         next_state = 2'd2;
     else
         next_state = 2'd0;
   end

endmodule
```

Figure 2. Example FSM in SystemVerilog

7. Implement a SystemVerilog module for your tail light sequencer.
   a. Add a new source file to your project for `taillight.sv`
   b. Your module should take a scalar reset signal and a scalar clock signal, as well as an input that turns the tail light signal on and off
   c. Your module should produce three 1-bit scalar outputs, corresponding to the outputs **`I`**, **`M`**, and **`O`**
   d. Implement your finite state machine design. You may choose to use either the Boolean equations from Part 1 or if-/case-statements to implement your state transition logic

e. Note that your implementation should use the **`always_ff`** block and a non-blocking assignment (`<=`) to implement the sequential logic. Refer to Lab 8 for more details on the syntax of these expressions.

8. Test your tail light signal module using a testbench.
   a. In `lab09_tb.sv`, instantiate a tail light module
   b. Generate a clock signal for your module and initialize the module using the reset signal in the first clock period. Refer to Lab 8 for additional information about generating clocks in simulation to test sequential circuits
   c. Configure your simulation so that the tail light module will cycle through all of its states

9. Simulate your circuit and verify your circuit's operation using the Wave view or by printing messages to the Tcl Console

10. Verify that your tail light module implements your finite state machine properly before moving to the next part of this lab assignment.

Stop and Check: Using the Vivado Simulator, ensure that your module works correctly and is triggered on the rising edge of the clock pulse.

Record: Screenshot the waveform of your test to include in your Lab Report.

Part 3: Connect your Sequential Tail Light to the Nexys A7 Board

In this part of this lab assignment, you will connect your sequential tail light module to the left and right LEDs on the Nexys A7 board, as shown in Figure 3. Switches **`SW[1]`** and **`SW[0]`** will be used to turn each turn signal on and off. Since your board will have two tail lights, you will need to instantiate two sequential tail light modules.

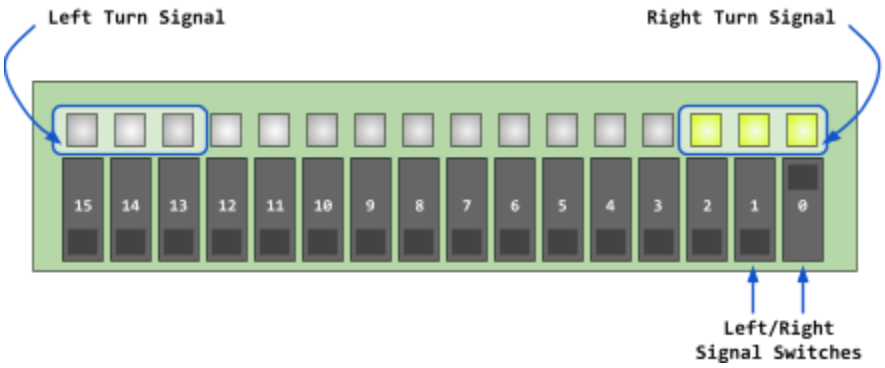

Figure 3. Input/Output Combination for Lab 9

The clock period for your sequential circuit should be configured to the desired frequency of your blinking tail light. A few cycles per second should suffice (*e.g.,* 2 Hz). To reduce the frequency of the Nexys A7's 100 MHz clock signal, you will use a clock divider, as done in Lab 8. This circuit creates a slower clock at the frequency you specify. The clock divider module is provided in the starter code. The following code example demonstrates instantiating the clock divider to produce a 100 Hz clock signal. You will need to choose a slower clock period to make the blinking tail light visually appealing.

```
clkdiv #(.DIVFREQ(100))  D0(.clk(clk100MHz), .reset(1'b0), .sclk(clk));
```

11. In the top-level module (lab09_top), instantiate two sequential tail light modules and a clock divider circuit. Configure the clock divider module and tail lights to match the above specification. Use switches **`SW[1]`** and **`SW[0]`** to control the tail lights.

12. Verify that the circuit schematic matches your proposed design. You can view the circuit schematic under "RTL Analysis/Open Elaborated Design/Schematic" in the Flow Navigator.

13. Generate a bitstream and program the FPGA. Test that your circuit functions correctly and demonstrate it to an instructor.

Stop and Check: Observe your blinking tail lights execute sequentially when the corresponding switch is on. Ensure that your clock period is appropriate.

Record: Screenshot a schematic for each module for inclusion in your lab report.

OPTIONAL: Add a Custom Tail Light Pattern

You can extend this lab assignment by designing a finite state machine and corresponding SystemVerilog module for a *custom* tail light display. Design a sequential LED display with at least three unique states. Use **`SW[2]`** to control whether your custom light pattern is on or off. If **`SW[2]`** is on, you can use a multiplexor to send the output from your custom module to the board's LEDs. When **`SW[2]`** is off, you can use the same multiplexor to select the signals from the sequential tail light modules from Part 3. This task is an optional (but fun) extension to the lab assignment.

Part 4: Wrap-Up

Before leaving the lab, make sure to complete the following steps!

14. ***Demonstrate the operation of your circuit to the instructor.*** A portion of your lab grade will come from the lab report in addition to this demonstration.

15. Turn the Nexys board OFF and disconnect it from the computer. Make sure to save your Vivado project and sign out of your machine!

Lab Report

Lab Report. In collaboration with your lab partner, submit one lab report on Moodle that describes the lab activity and your team's circuit design. Your written report is worth 60% of your grade for each lab assignment. Your lab report should include the following sections:

- **Title Page:** Your lab report should begin with a title block that includes the course number, the title of the lab assignment, the names of each team member, and the date of the lab assignment. The title page should also include a "Statement of Collaboration" (detailed below) and an estimate of the total amount of time spent on the lab assignment. The inclusion of these components are graded, but the content of each statement is not used in determining your grade.

- **Statement of Collaboration:** Describe the contributions of each team member to the lab assignment, including writing of the lab report. What is specified in this section will not affect your grade, however it is expected that you periodically rotate roles within your lab group if you do divide work, including writing the report.

- **Introduction:** Include a brief paragraph summarizing the objectives of the lab and what your team achieved. This section should be written in your own words; it is not acceptable to copy text for this from the lab handout.

- **Design:** Include a concise yet descriptive explanation of the circuit design, following all steps in the process—from the problem specification and initial truth table, to equations and circuit schematics. Tables and equations should be typeset and clearly labeled. Technical information should be accompanied by some text narrating each step or aspect of the design.

- **Testing & Results:** Describe how you tested your circuit, including rationale for why you chose specific test cases if the circuit is not tested exhaustively. If you tested your circuit at multiple points (*i.e.,* using a testbench and then on the Nexys board), this section should include all test cases at each point. Test cases can be included in the form of tables or simulation waveforms. Results should be neatly formatted and labeled, indicating what step is being tested. This section should also include any recorded measurements, tests, diagrams, or images specified by the lab document.

- **Discussion/Conclusion:** Your discussion/conclusion section should comment on any observations about the technique you employed to get your results. Briefly describe any difficulties that you encountered and how you resolved them. Then, summarize what you concluded from your work. (It may also be helpful to think negatively, *i.e.* what your results did not show. Remember that a negative result is often as valuable as a positive result!)

  A number of technical questions may be posed in the assignment sheet. If questions are given, include your answers to them in this section.

Your lab report should be typed and submitted as a .pdf document. Only one member per team needs to submit the lab report.