Carleton University

Department of Systems and Computer Engineering

SYSC 2310 Fall 2017

Introduction to Digital Systems

Lab #5

Lab Report Due: At **6:00 am** on the day that is one week after your scheduled lab session.
(E.G. if your lab is on Tuesday, the report is due on Tuesday one week later.)

This is Version 2: Only one small change, highlighted in the next line.

Submit the report as a single PDF file. Place the report in a folder named Lab5, along with your lab work as described below. Zip the folder (Lab5.zip) and submit it to the appropriate location in the cuLearn webpage for your lab section (not the main course webpage!).

Your work must be submitted in a Windows-compatible ZIP file. Do **not** submit your work in RAR, 7z or other format. (If the TAs can't open your work using Windows' built-in zip utility then they will not mark it.)

Your work must be "SUBMITTED". "Draft" submissions will not be marked.

Note that the system will automatically stop accepting submissions at the due date/time for your lab section.

LATE submissions will NOT be accepted.

Read this document completely and carefully before deciding what to do.

Please bring questions about the lab to class for discussion at the beginning of class.

In this lab you will:

- **Design and Implement the T-Bird Taillights control circuit using ROM-Based FSMs.**
- Enter circuit designs into Logisim.
- Test and debug circuits using Logisim's simulation capability.

"[Report]" indicates content that should be addressed in the lab report.

Step 1: Read this entire document before proceeding to Step 2.

The Ford Thunderbird (https://en.wikipedia.org/wiki/Ford_Thunderbird) is a classic automobile. The Square Bird model (1958 – 1961) has the taillight configuration shown below.

The sequential taillight controls introduced to the Square Bird in 1959 have been inspiring labs in digital systems courses for years (Carleton courses have run a version of this lab since the 1970's … Prof. Pearce did the T-Bird Taillights lab as a student at Carleton "back in the day"). Unfortunately, there doesn't seem to be a readily available video of the 1959 taillights in action, or one that captures all of the control-related behaviours … but the same sequential taillight sequence for turning and braking model can be seen in this video of a 1966 model: https://www.youtube.com/watch?v=Qwzxn9ZPW-M (car buffs love that sound ☺). The second half of the video shows the taillight behaviour while the night-time running lights are on (i.e. while the headlights are on), and this behaviour is not usually included in labs since it doesn't really add anything interesting to the sequential behaviour.

This lab is based on the classic control sequence of the 1959 Square Bird model. There are 3 input controls: a 3-position turn lever on the steering column (Turn Left, Turn Right, No Turn), a 2-position brake indicator associated with the brake pedal (Brake On, Brake Off), and a 2-position 4-way emergency flasher switch on the dashboard (4-Way On, 4-Way Off).

Warning: some of the behaviours in this lab may be different than labs in other courses.

When turning left or right, the appropriate group of taillights sequence inner-to-outer as seen in the video (i.e. inner, inner + middle, all 3 on, all off, repeat). Assume that each state in the turning behaviour is one Tick in duration. The braking behaviour is also shown in the video. When braking (i.e. Brake On), all of the lights turn on simultaneously, except when a turn is also being indicated. If braking while turning, then the appropriate group of lights continue to indicate the turning sequence, while the other group of lights all turn on (i.e. braking does not override a turn indication). When the 4-way flasher is on (4-Way On), all lights cycle on for one Tick and then off for one Tick. The 4-way cycling continues until the control is turned off (4-Way Off). The 4-way flasher overrides any turn indication. If an overridden turn is still on when the 4-way flashers are turned off, then the turn indication sequence should start the

beginning of the next Tick (i.e. only the inner light on). If the brake is also applied (Brake On) while the 4-way flasher is on, then all lights should go on simultaneously (i.e. brake overrides the 4-way flasher).

Note: the above behaviour description includes important information about how controls are prioritized to override other controls.

The turning and 4-way behaviours are all synchronized to the same clock (with period = 1 Tick).

Braking is asynchronous (i.e braking behaviour is not synchronized to any clock).

The Lab5Start.circ file has been included as a starting point for the lab. Do not modify these included circuits in any way. Furthermore, do not attach wires directly to any signals in the provided circuits, i.e. use the provided tunnels only to access the circuits. The Turn Left, Turn Right and No Turn signals have been designed to mimic a 3-position control switch, and only allow one of the turn bits to be 1 at a time. A "real" implementation would only include 2 signals (Turn Left and Turn Right), but the No Turn signal has been included to simplify the use of turn signals in your design. The Brake Off and 4-Way Off signals are also redundant, but have been included for simplicity and to avoid the need to add inverters in your implementation.

The included Lab 5 Tables (Word) document contains some tables to support your FSM development and to simplify constructing your Report. Note that some tables may include more rows and columns than you need; ignore any rows and columns that you don't need. Also, be sure to label the column headers appropriately.

Step 2) Left Turn: Follow the FSM Design-then-Implement development guidelines presented in the course slides to develop an FSM to control the left turn behaviour. Develop your in the following order:

1. Design: Complete a Turn (Abstract) Sequence Table. In the table, replace <direction> with "Left". In the Light Pattern show the state of each of the three lights in the left turn group, and use "x" to indicate a light is off and "o" to indicate a light is on.
2. Implementation: Complete an Encoding Table for the abstract States in your Abstract Sequence Table, and another Encoding Table for the Light Patterns. In the Encoding Tables, be sure to label the headers appropriately (replace "<info>" with the name of the abstract info that is encoded), and enter 0 or 1 in the appropriate binary columns (recall: ignore any rows and columns that you do not use).
3. Implementation: Complete an Implementation Next State Table. All of the completed entries should contain binary numbers, except for the rightmost column which should contain hex numbers.
4. Design the FSM circuit.

Your FSM implementation must include only one Register and one ROM (no additional gates). The numbers of bits needed in the Attributes of these components depends on the data in your Implementation Next State Table. There will be marks associated with using only the minimum number of bits required for a solution. Use the Turn Left input signal to control when the FSM is activated. Be sure the FSM is in an appropriate state when not turning left.

Note: you will have to Enable Ticks in order to get the Clock to work.

Be sure to save your work periodically.

[Report] Include the tables completed during the left turn FSM development.

Step 3) Right Turn: Extend your Step 2 design. Follow the FSM Design-then-Implement development guidelines presented in the course slides to develop another FSM to control the right turn behaviour (i.e. your solution will have 2 separate FSMs, one for turning left and one for turning right). Develop the right turn FSM by following the same development process listed in Step 2, but use table headings appropriate to this step.

Your FSM implementation must include only one Register and one ROM (no additional gates). The numbers of bits needed in the Attributes of these components depends on the data in your Implementation Next State Table. There will be marks associated with using only the minimum number of bits required for a solution. Use the Turn Right input signal to control when the FSM is activated. Be sure the FSM is in an appropriate state when not turning right.

[Report] Include the tables completed during the right turn FSM development.

Step 4) 4-Way Flasher: Extend your Step 3 design. Follow the FSM Design-then-Implement development guidelines presented in the course slides to develop another FSM to control the 4-way emergency flasher behaviour (i.e. your solution will have 3 separate FSMs, two for turning and one for the 4-way flasher). Develop the 4-way flasher FSM by following the same development process listed in Step 2, but use table headings appropriate to this step.

Your FSM implementation must include only one Register and one ROM (no additional gates). The numbers of bits needed in the Attributes of these components depends on the data in your Implementation Next State Table. There will be marks associated with using only the minimum number of bits required for a solution. Use the 4-Way On input signal to control when the FSM is activated. Be sure the FSM is in an appropriate state when not activated.

In order to override the turn signals appropriately, two output logic circuits will be needed (one for the Left Group output logic and one for the Right Group output). Each output logic circuit must consist of only one MUX.

Also, care must be taken to ensure that an overridden turn signal indication always starts from the innermost light when the 4-way flasher is turned off. This may require one additional gate to be added to the control inputs to each of the turning FSM circuits (from Steps 2 and 3).

[Report] Include the tables completed during the 4-way flasher FSM development.

Step 5) Asynchronous Braking: Extend the design of (only) the output logic in Step 4 to include braking behaviour. Braking is an asynchronous behaviour, so must not involve the Tick clock signal. Be sure to consider the override conditions carefully. The logic is a bit more complex, but not much.

Full marks will be awarded for designs that use a total of two MUXes + 5 or less gates (with no inverters or inverted inputs), i.e. the total is for both the Left Group and Right Group output logic circuits combined. Part marks will be awarded for designs that use extra MUXes and/or gates (inverters or inverted inputs count as extra).

[Report] State whether your circuit works successfully relative to the required behaviour. If you have some bugs, describe only the behaviour that is working correctly (do not describe the bugs). The TAs will test your circuit looking for the required behaviour that you have claimed works. Marks will be awarded for both how much you have completed and how accurately you have claimed the degree to which the circuit works.

Save your final circuit as Lab5Final.circ.

Bonus) Output Logic LUT:

In each of the output logic circuits, replace any gates used with a LUT.

Save your bonus circuit as Lab5Bonus.circ.

[Report] State whether you have included a working bonus circuit.

The lab is done! ☺ If you could not complete the lab during the scheduled lab time … that's OK … finish up the lab on your own time.

Before you leave the lab (even if you could not complete the lab), show the TA what you have completed. [Attendance marks are associated with being present and working on the lab through the entire lab session.]

Before the due date for your lab section (i.e. does not have to be done during the scheduled lab): Prepare your report and place the single pdf in the unzipped Lab5 folder containing your lab circuits. Name the report Lab5Report.PDF (the folder should retain the name Lab5). Zip and submit the Lab5.ZIP folder to the appropriate place in your lab section's cuLearn page.