CSE 100/L Logic Design
Fall 2025

Lab Assignment 5

Prelab deadline:

- There is no pre-lab assignment for this lab assignment.

Code submission deadlines:

- 100% Before the end of your section Thursday, November 6th.
- 95% after the end of your section (before 11:59pm) Thursday, November 6th.
- 80% after 11:59pm Thursday, November 6th.
- 60% after 11:59PM Sunday, November 9th.
- 40% after 11:59PM Monday, November 10th.
- 20% after 11:59PM Tuesday, November 11th.
- 0% after 11:59PM Wednesday, November 12th.

Demo deadline:

- Code may be demoed until 11:59PM Wednesday, November 12th.

Write-Up deadline:

- There is no write-up required for this lab.

Verilog operator and procedural block constraints:

There are many ways to implemement this lab. However, for CSE 100 you must follow these rules:

- All combinational logic must be implemented using only **assign** statements.
- All combinational logic must be implemented using **only** the following operators: **&**, |, ~, ^.
- Concatenation and replication are acceptable.

In all of our labs from here on, your designs must be **synchronous with the system clock** specified in the lab. This means that you must

- use only positive edge-triggered flip-flops,
- not use asynchronous clears or pre-sets of any sequential elements,
- connect only **the system clock** (provided to you) to the clock input pins of any sequential components,
- not connect **the system clock** to any input pins other than the clock input pins of sequential components, and
- only use the **assign** statement in your design.
- **not** use any procedural blocks in your design (i.e. always@, always_ff@, always_comb).

Use of any other operators or procedural blocks will be considered behavioral and will result in a **score of 0**! When in doubt, check with a TA!

Overview

In this lab you will design another state machine. This time you will be using signals that could be from two infrared light sensors to count the number of objects crossing the two sensors in both directions. We'll be using pushbuttons as stand-ins for the sensor inputs.
You will use the BASYS3 board to implement the following system. An infrared (IR) sensor is a device which converts infrared light into an electric charge. By deploying the IR emitter and IR sensor on opposite sides of a path, an object traversing the path is detected when it blocks the infrared light from reaching the sensor. Normally the IR sensor output is high, but when an object blocks the light source the output goes low.

In this lab you will use two signals from two IR sensors to detect objects crossing through a corridor. (We will use the pushbuttons to simulate the signals.) The sensors are placed next to each other on one side of the narrow path with the emitters on the opposite side of the path as shown below.

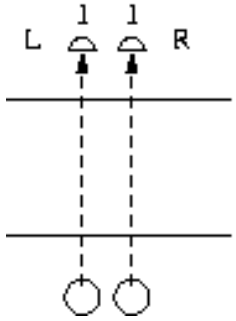

Figure 1: Double IR sensor.

The sensors are close enough so that no object of interest fits in between the two sensors without blocking at least one of them. When an object (such as a turkey) moves from left to right across the sensors the waveforms below are observed on the L and R sensor outputs.

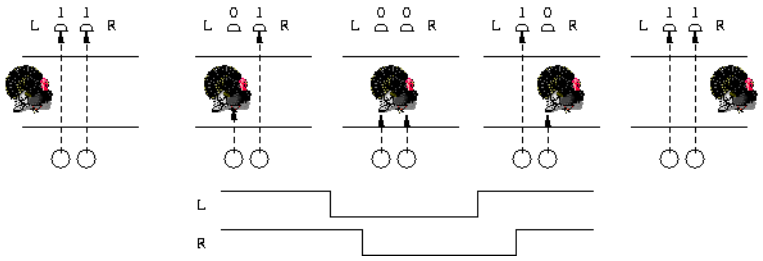

Figure 2: Turkey movement left-to-right.

If instead the turkey travelled from right to left we would observe different waveforms as shown below.

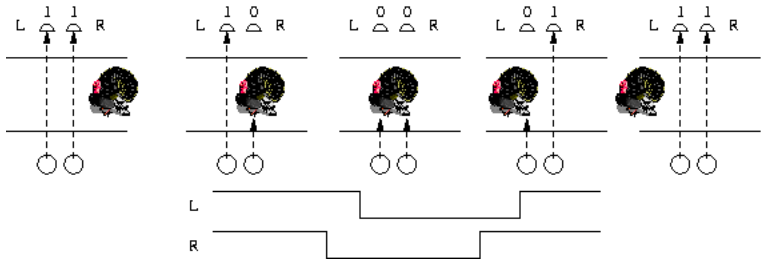

Figure 3: Turkey movement right-to-left.

The length of the low pulses depend on how fast the turkey is traveling. One difficulty is that some objects (particularly turkeys) don't always know which way to go. So the waveforms observed might be more complicated if the object changes direction, perhaps multiple times, before completely crossing both sensors.

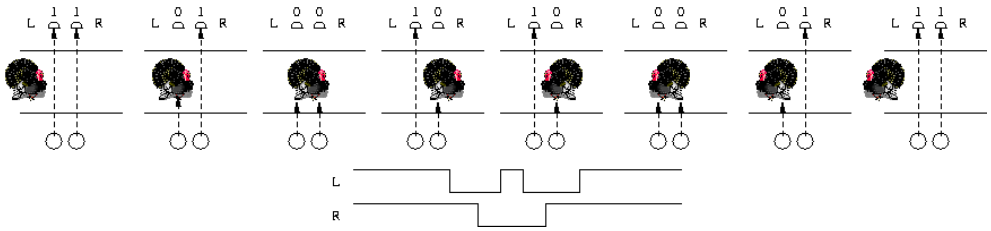

Figure 4: Indecisive turkey movement.

There are many possibilities to consider. We assume that there is no limit on how many times a turkey can change direction. But all of the turkeys are too wide to fit in between the sensors. That is, no object can cross the sensors without both sensors being simultaneously low at some point. All objects appear as convex polygons (no holes) from the side, so they will interrupt the sensor continuously while they are crossing it. You may assume that there is sufficient space between turkeys so that only one turkey can be in the region bounded by the two sensors. Finally, you may assume turkey's cannot fly. That is, there is no way for a turkey to block both IR sensors, then suddenly leave the region bounded by the two sensors.

Your assignment is to implement a digital system on the BASYS3 board that monitors the signals from the two IR sensors, and counts the number of objects that cross from right to left as well as the number of objects that cross from left to right. You will keep track of the difference using an up-down counter and display the difference on the 2 rightmost digits of the seven segment display. There are at most 127 turkeys so 7 bits will be enough to hold the highest positive value (127). But there could be more left-to-right crossings than right-to-left and this will be a negative number (as low as -127). So your counter will range from -127(-7F in hex) to 127(7F in hex).

It is extremely convenient that no change in the binary counter is required to hold 2's complement integers. But when the value held by the counter is negative, it should be converted to its positive magnitude and displayed with a minus sign. For example, in the image below the number displayed is -2 since internally the counter has FE. So, when the number is negative you will need to turn on the led CG (seg[6]) of AN2 and convert the value to its positive magnitude. (Hmmmm, perhaps a component from a past lab would be useful here .....)

LED15 (led[15]) displays the signal from the left IR sensor and LED8 (led[8]) displays the signal from the right IR sensor.

One last detail . . . we need to keep track of the direction of the last crossing. The 8 rightmost leds, led[7:0], should turn on in sequence repeatedly from left to right if the last crossing was left-to-right, and turn on repeatedly from right to left if that crossing was right-to-left. They should turn on at the rate of one per 0.25 seconds. All of these eight leds should be off before the first crossing and also should be off while a turkey is attempting a crossing. Note that if the turkey does not succeed in crossing (both sensors become unblocked again without a crossing), then the leds will resume turning on based on the last crossing.

You will be provided with a signal **qsec** that is high for one clock cycle every quarter of a second (same as Lab 4) that you can be used for the leds. Do not use **qsec** as a clock.

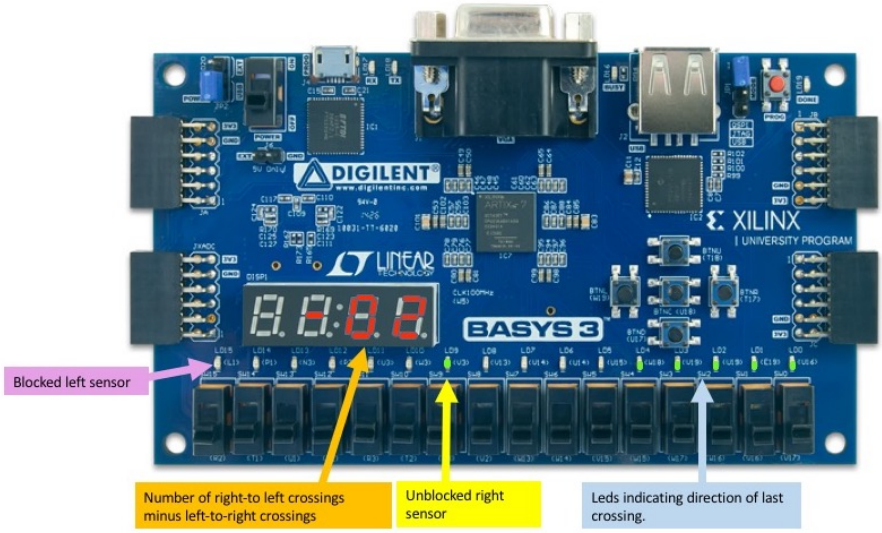

Figure 5: Implementation on Basys3.

System

You will need to build a state machine which takes the sensor inputs and sends outputs to control a Turkey Counter and the LED Shifter. The output of the Turkey Counter will be displayed on the seven-segment displays. Note that negative numbers are represented with a "-" sign.

Turkey Counter

You will need an 8-bit counter that can count up or down. Your Lab 4 counters should be handy.

LED Shifter

You will also need an 8-bit shifter for controlling the 8 rightmost LEDs. You'll want to be able to shift in either from the right or left at the rate of 0.25 seconds. The signal **qsec** provided by **qsec_clks** should be used. You'll also want to be able to reset the shifter to turn off all of the LEDs.

Sensor Inputs

The current value of the sensor inputs should be displayed on LED15 and LED8. Since the sensor inputs are normally high (when unblocked) while the pushbuttons are normally low you will need to invert these signals from the pushbuttons before sending them to the LEDs. You may choose to use the complement or uncomplemented versions of the inputs in your state machine.

Since the pushbuttons are asynchronous inputs you will want to synchronize them before connecting them to you state machine. Be sure to initialize these synchronizers to reflect the default value of the sensors.

Procedure

1. Simulate your state machine separately. Try interesting cases where the object changes direction several times in different places.
2. Download this verilog file and save it as **qsec_clks.v** in your project directory.
3. In the Vivado Project Manager, add it to your project. Make sure you select the option to copy it into your project.
4. Add an instance of the module **qsec_clks** to your top level as follows:

```
qsec_clks slowit (
  .clkin(clkin),
  .greset(btnU),
  .clk(clk),
  .digsel(digsel),
  .qsec(qsec)
 );
```

5. The signal **clk** is your system clock. The signal **digsel** should be used to advance the Ring Counter for the 7-segment displays; it should not be used as a clock!!!
6. The signal **qsec** is high for one clock cycle each 1/4 second (4 times per second) and should be used to advance the shifter; it should not be used as a clock input. **Note that btnU is being used for global reset signal in Lab 5.**
7. Simulate your entire design. Though your parts may seem correct when you simulate them separately, they may not work together.
8. Implement your design and download it. Test your design by pressing btnL and btnR to simulate an object crossing the sensors. Try several tests including some where the object changes directions several times. Before demonstrating your design, confirm that the crossing difference and direction display correctly for all cases.
9. Once testing is successful, ***demonstrate your simulation and implementation to a TA.***

    - The TA will ask you to describe the tests you performed in simulation.
    - The TA will confirm that your code is structural.
    - The TA will ask you to demonstrate left-right, right-left, and indicisive movements.

10. Once your working design has been demonstrated, ***submit your project***.
11. Remember to archive your project. Files left on the PC are not protected. Should it become necessary to re-image that PC, its disks will be wiped clean.
12. **Important** Please remember to turn off the power to BASYS3 board when you are done.