18-341: Logic Design and Verification

Homework 5: Blinking Lights

Objective and Overview

The purpose of this assignment is for you to gain insight into generating serial waveforms, especially those that are a bit different than the standard types.

Schedule and Scoring

All homework assignments are equally weighted and will be scored out of 100%. This homework will overlap others — get it done when you can.

A Note about Collaboration

All 18-341 homework assignments are to be accomplished individually. All work must be your own.

Assigned	18 Oct 2018
Due	29 Oct 2018 at 12:30pm

Hints from others can be of great help, both to the hinter and the hintee. Thus, discussions and hints about the assignment are encouraged. However, the homework must be coded and written up individually (you may not show, nor view, any source code from other students). We may use automated tools to detect copying.

How to Turn In Your Solution

Your SystemVerilog code should be in a file named **`hw5.sv`**. The directory **`/afs/ece/class/ece341/handin/h5/`** has a subdirectory for each of you. Turn in your solution there.

Then, find a TA during office hours and demo your system.

Your Mission: Pretty Lights

We have a string of 5 LED lights that can be plugged into your DE0-CV board. These lights respond to a serial waveform that tells each light what color it should display. Develop a SystemVerilog serial transmitter that will drive these lights in response to switch and pushbutton stimuli.

You will develop some explicit-style FSMs that control the datapaths[1]. Even though you could get this same string of lights (though longer) for an Arduino, your code shouldn't look like software.[2] Google "Neopixel" to see lots of fun Arduino / LED projects, though.

---

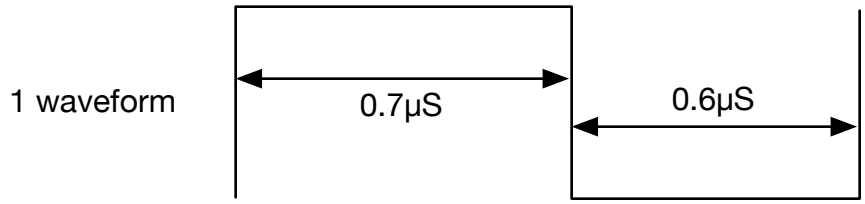
[1] The datapath elements are mostly counters. Yawn.

[2] Ask Prof Nace about his Build18 project using these guys

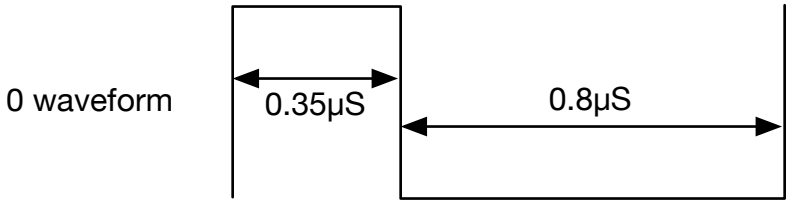
The LED Protocol

The string of LEDs responds to a serial, pulse width modulated waveform. That is, there is information in the width of the pulses as you can see below.

A 1-bit is sent with this form and timing:



A 0-bit is sent with this form and timing:



In order to tell one LED what color it should display, send it 24 contiguous bits, which we will call an *LED command*. 24 bits makes 3 bytes, each of which controls how much of the intensity should be red (R), green (G) and blue (B). So, 256 different amounts of each of R, G and B. The LED command is a concatenation of the three 8-bit values, like so:

```
logic [ 7:0] R, G, B;
logic [23:0] LED_command;
assign LED_command = {G, R, B};  // Note: Order
```

Send G[7] as the MSB of the LED command.

A *display packet* is made up of multiple LED commands, one for each LED in the string. The LED commands are sent back-to-back with no time in between. The display packet is sent serially to the first LED, which captures (removes) the first LED command (which it will obey). It then sends all remaining LED commands in the display packet to the next LED in the sequence. The second LED will capture the first LED command it receives (the second in the original display packet), obeys it, and then sends all remaining LED commands on down the line. In this way, the individual LEDs don't need to know how many LEDs there are, where they are in the string, or any of that stuff.

If you like the LEDs the way they are, you need do nothing further. If you don't send any more display packets, the LEDs will remember their state and keep sending colored photons to your eyes. However, if you wish to modify the color (or intensity) of any of the LEDs, you just send another display packet. You must wait 50 µS between display packets, as illustrated below:

This illustration shows a single LED command, consisting of 24'hF08025, followed by a space and the beginning of the next display packet.

In order to reset the LEDs, you should hold the line at logic 0 for at least 80 μS.

FPGA Interface

Pin GPIO_0[1] is a general purpose I/O pin. It is connected to the serial output that drives the LED strip. Send all your well-timed 1s and 0s out this pin. This pin is located at FPGA location B16.

You will also need to connect to CLOCK_50 (or CLOCK_50_2), so you can generate your timing.

User Interface

There are 5 LEDs in the string (i.e. 120 bits of variation). You only have 10 switches and 4 push buttons to issue commands to your LEDs. I want you to use those 10 switches / 4 push buttons to make "something creative" happen to the LEDs.

"Something creative" is a pretty open-ended requirement. Here are a few additional requirements:

- Use the buttons and switches to control your LEDs somehow. You must have some level of interactivity.
- You need to show different colors on different LEDs at the same time. Just sending all white to all 5 LEDs is boring, not creative.
- You need to show different intensities of colors somehow. Unlike the previous requirement, all the LEDs can have the same intensity at the same time, should you wish.
- You need to have something happen because time has passed. In other words, LED changes can't just be because buttons got pressed. For instance, maybe every second the colors rotate.

So, for example, you could do the following (barely, just barely, creative) project:

Each LED's color is controlled by two of the switches: 00 for Red, 01 for Blue, 10 for Green, 11 for White. In other words, SW[1:0] controls the first LED, SW[3:2] controls the second, etc. When you push a button, the intensity drops to 50%.

More creative projects might be 1-d pong, swirling lighting effects, disco-ball strips, etc.

Have some fun with this project. Don't do boring projects.

Your Code

We will grade you on a working demo, applied creativity, and appropriately written code. Strive to have your code shown off in recitation for all the right reasons.

Your code should be explicit-style FSMs that control datapath elements. Most of those datapath elements will probably be counters.

Also, the 1/0 waveforms described above have some slop. Your code should generate 1s and 0s that resemble those given. Finding a shortcut and then exclaiming, “Well, the LEDs still work” is sloppy engineering and will not result in full credit.

Availability

Check out an LED strip from the ECE Course Hub.

Below is a picture of the board with the LED strip attached. Note that it goes in the left of the two connectors. The connectors are keyed, so you shouldn’t be able to insert it backwards — please don’t test this assertion. Some strips will lean left and others right, depending on how the connector is keyed.

When removing the LED strip from the connector, please don’t pull on the ribbon cable, shrinkwrap or the LEDs themselves. They are gentle components and somewhat fragile. Please don’t make the TAs mad, as broken LED strips mean TAs will have to solder replacements.