Intel® FPGA Hands-On Lab:

Light Rider IDE version

Table of Contents

Background 3
Light Rider Pattern 4
Summary 4
Light Rider SystemVerilog Code 6
Opening the LabsLand IDE 6
Creating "light_rider.sv" 7
Synthesis 8
Debugging Code 9
Upload to FPGA 10
More Debugging 11
Even More Debugging! 11
Correcting the Double Blink Problem 12
Bonus!!! 13

Background

A field-programmable gate array, or FPGA, is a digital semiconductor device that can be used to build a wide variety of electronic functions. These include data center accelerators, wireless base stations, and industrial motor controllers to name a few common applications. FPGAs can be infinitely reconfigured to perform different digital hardware functions, which also makes them an excellent learning platform.

To configure an FPGA, first you describe your digital electronics with either a Hardware Description Language (HDL), such as Verilog or VHDL, or a schematic. Then you assign the "pins" of your FPGA based on how the Printed Circuit Board (PCB) connects to various peripheral components. Some examples of peripherals are switches, LEDs, memory devices and various connectors. Finally, you "compile" your design and program the FPGA to perform the function you have specified.

This demonstration assumes you have some working knowledge of how computers and digital electronics work, but by no means do you need an electrical engineering degree to attend this lab.

Light Rider Pattern

Summary

In this lab we will create an interesting lighting pattern that you may have seen before, a horizontal bar of lights that sequences one at a time from left to right and back again at the rate of about 1/10th of a second per light. This lab will teach you about sequential logic, timers, and flip-flops.

Let's quickly review how flip-flops work.

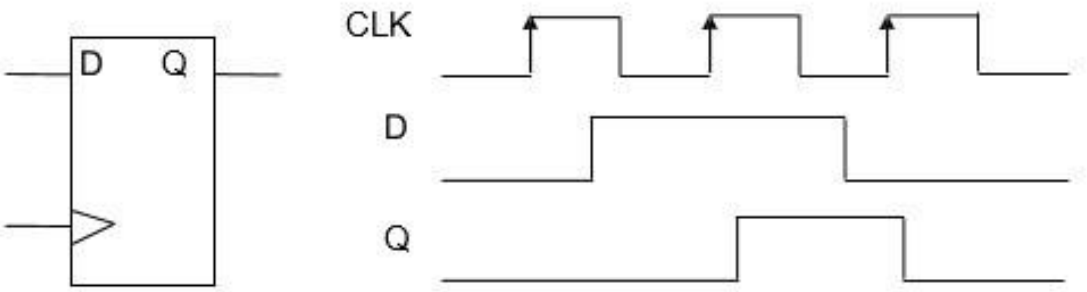

Figure 1: Flip-flop diagram

Flip-flops are basic storage elements in digital electronics. In their simplest form, they have 3 pins: D, Q, and Clock. The diagram of voltage versus time (often referred to as a waveform) for a flip-flop is shown in Figure 1. Flip-flops capture the value of the "D" pin when the clock pin (the one with the triangle at its input) transitions from low to high. This value of D then shows up at the Q output of the flip-flop a very short time later.

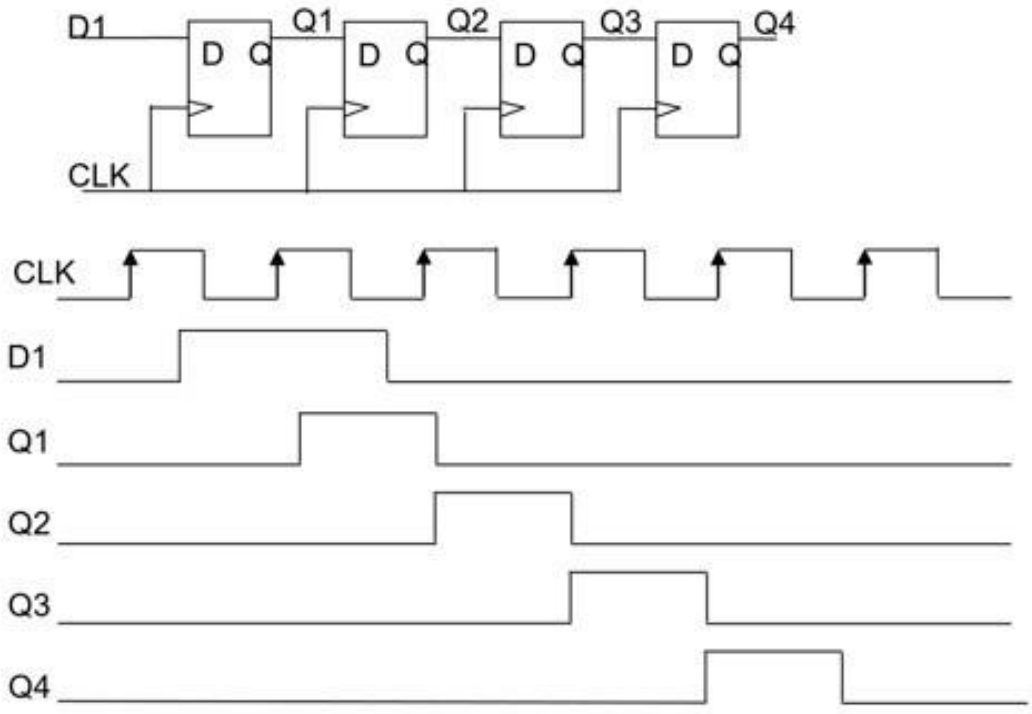

Figure 2: Shift Register diagram

When you connect several flip-flops together serially you get what is known as a shift register. That circuit serves as the basis for the Light Rider LED controller that we will study in this lab. Notice how we clock in a 1 for a single cycle and that bit "shifts" through the circuit. If that "1" is driving an LED each successive LED will light up for 1/10 of a second.

Light Rider SystemVerilog Code

Review the light_rider.sv code. The file can be opened as a text document.

The code given **intentionally** has errors. Spend a few seconds and see if you can find them.

If you do find them, don't fix them yet. We need these to show off the message console in the IDE to see how errors are presented.

Opening the LabsLand IDE

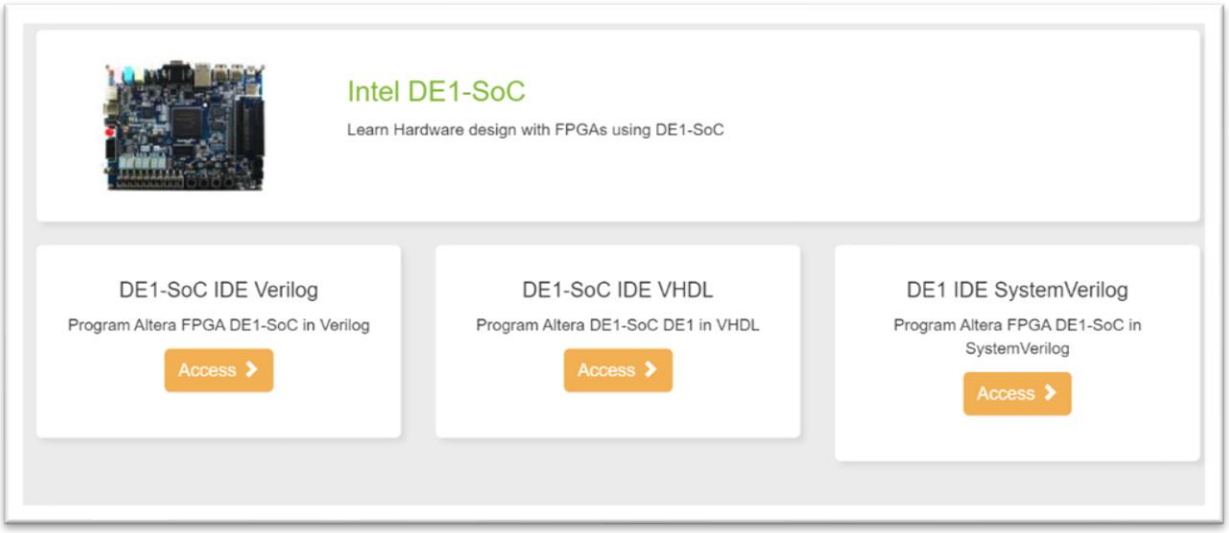

Figure 3: IDE Language selection menu

Select the **Intel DE1-SoC** tile next to the Digital Trainer tile.

Then, select **DE1 IDE SystemVerilog**, shown in Figure 3.

Creating “light_rider.sv”

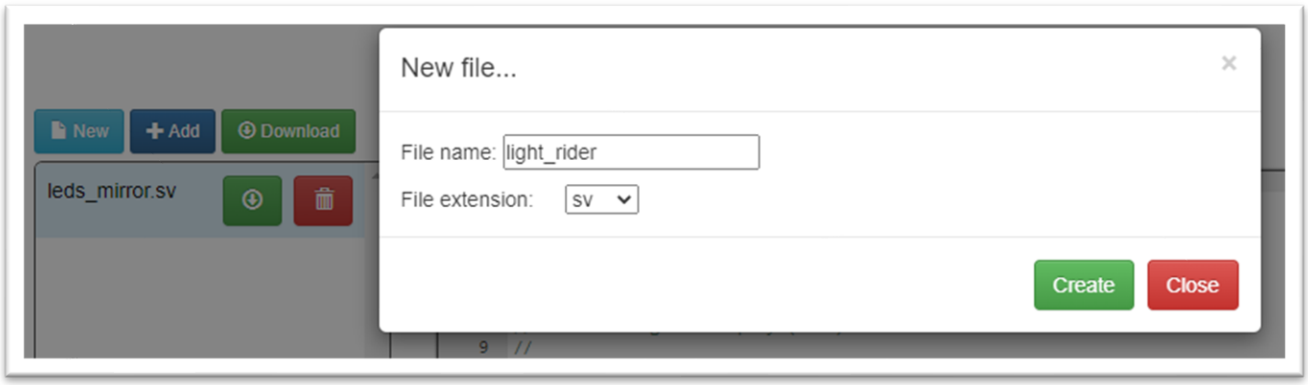

Figure 4: New file... dialogue box

To create the light_rider.sv file do either of the following:

1. Click **Add** and follow the dialogue boxes to submit the downloaded file, or
2. Create a **New** file named light_rider (figure 4) and paste the contents.

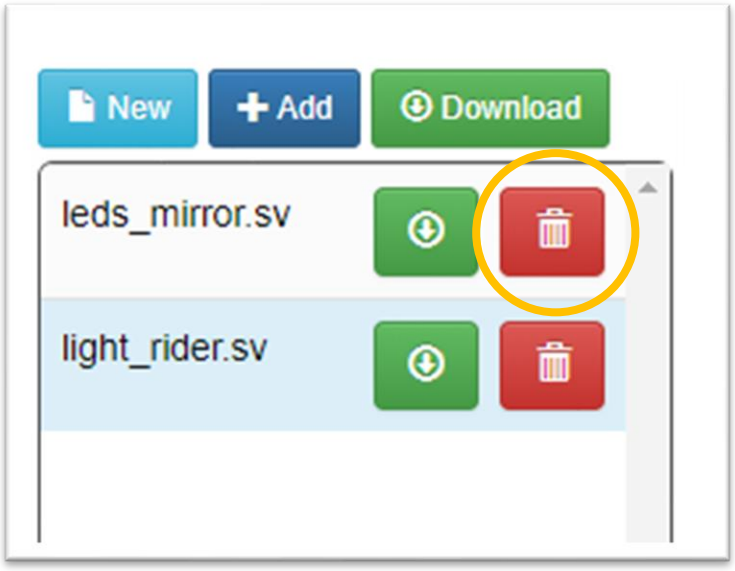

Figure 5: Delete button

Delete the **leds_mirror.sv** file (figure 5) that was already in the interface.

Synthesis

After creating your new light_rider.sv file in LabsLand, click **Synthesize** to prepare the new design to be uploaded to the FPGA board.

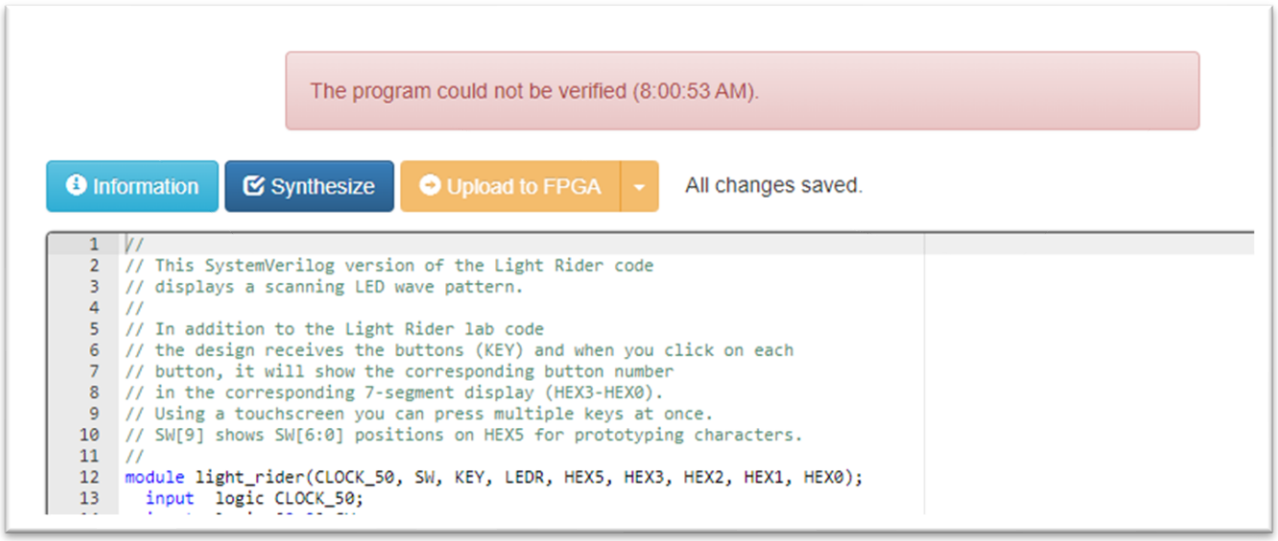

Figure 6: Synthesis failure message

You should see, “The program could not be verified.” It looks like there was an error with our design.

Debugging Code

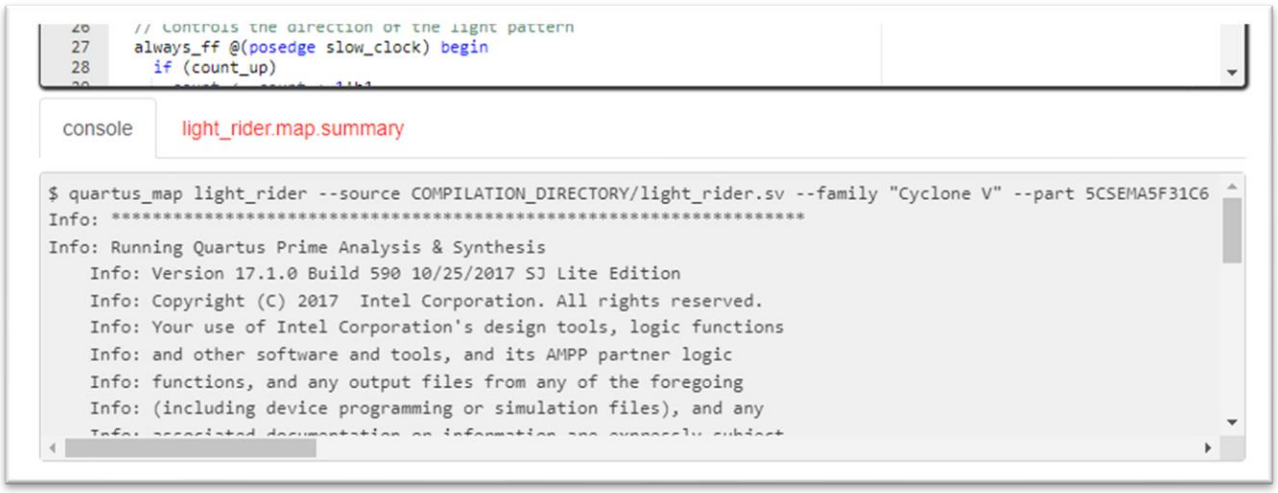

Figure 7: Console tab

Below the text editor there is a tab called “console.” Scroll down past the numerous Info lines until you see the first Error message.

Carefully look at the source code and fix the error and then click **Synthesize** again.

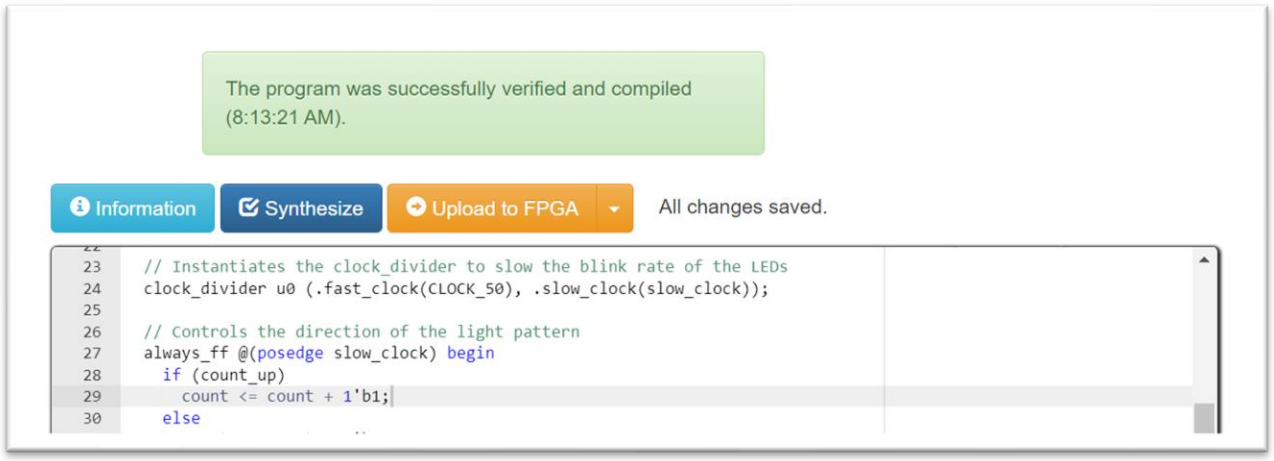

Figure 8: Synthesis success message

Wait until the synthesis process runs to completion as indicated by the dialogue box in Figure 8.

Upload to FPGA

Click **Upload to FPGA**.

After a waiting screen the board will appear and immediately begin its operations. Do you see the Light Rider pattern?

When working properly, you should see something like the following:

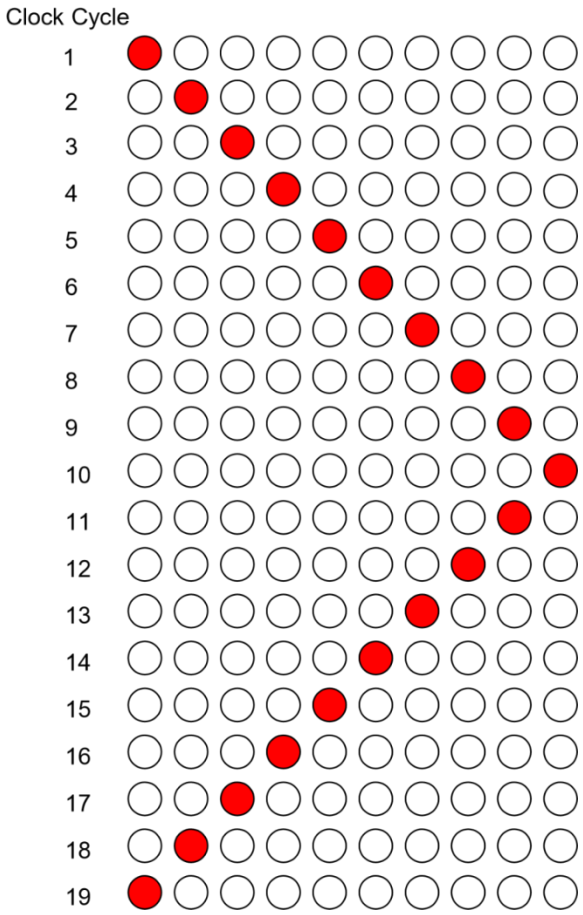

Figure 9: Example of the Light Rider sequence

What do you see? If it does not work move on to the next section.

More Debugging

Why don't the lights sequence? The selected clock frequency is 50 MHz. That means the clock changes 50 million times per second. If you change the LEDs at that rate, you cannot view them with the unaided eye.

Click **Leave now**, to return to the IDE.

When you go through the code you will see a module called **clock_divider**. We need the output clock to toggle at around 10 Hz (10x per second). This clock_divider module takes the 50 MHz clock and divides the clock to a slower frequency. Your lab instructor goofed and did not calculate the right ratio to slow the 50 MHz clock down to 10 Hz.

Calculate the proper size of the counter that divides 50 MHz to roughly 10 Hz. Think about a ratio that is 2^N where N is the width of the counter.

Solve N for the following equation: $10=50{,}000{,}000/2^N$ (HINT: use the log function). Round N up to the nearest integer to obtain the proper COUNTER_SIZE parameter setting.

Adjust the parameter COUNTER_SIZE to the appropriate value then **Synthesize** and **Upload to FPGA**.

Even More Debugging!

Is the Light Rider sequence working properly? Does each LED stay on for about 1/10 second? If not, redo your math to find the right parameter. Look at the sequencing carefully. Does each LED illuminate once and proceed to its neighboring LED? As you will observe LED[0] and LED[9] will blink twice.

That lab instructor made another error in the design! Look in the light_rider.sv code and see if you can find the error.

Note: The COUNTER_SIZE looks good when set to 23 or 24.

Correcting the Double Blink Problem

An example simulation shown below illustrates the problem. Note how count increases past 9 to 10 and causes the erroneous double blink problem.

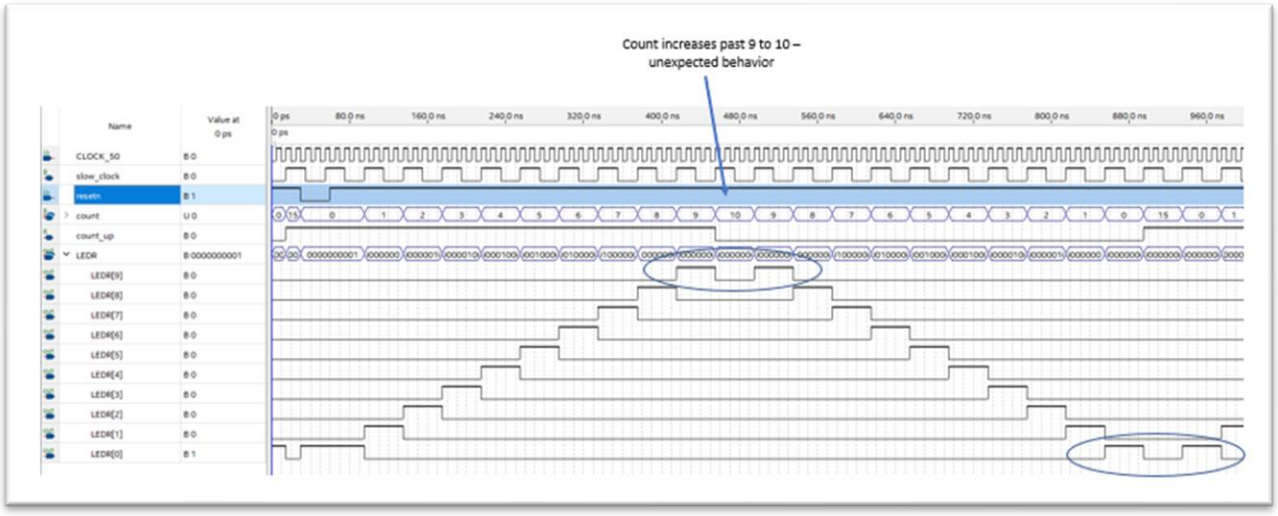

*Figure **10**: Incorrect sequencing waveform*

To fix this over-counting issue, change the cutoffs from between (count == 9) and (count == 0) to between (count == 8) and (count == 1) to correct the problem.

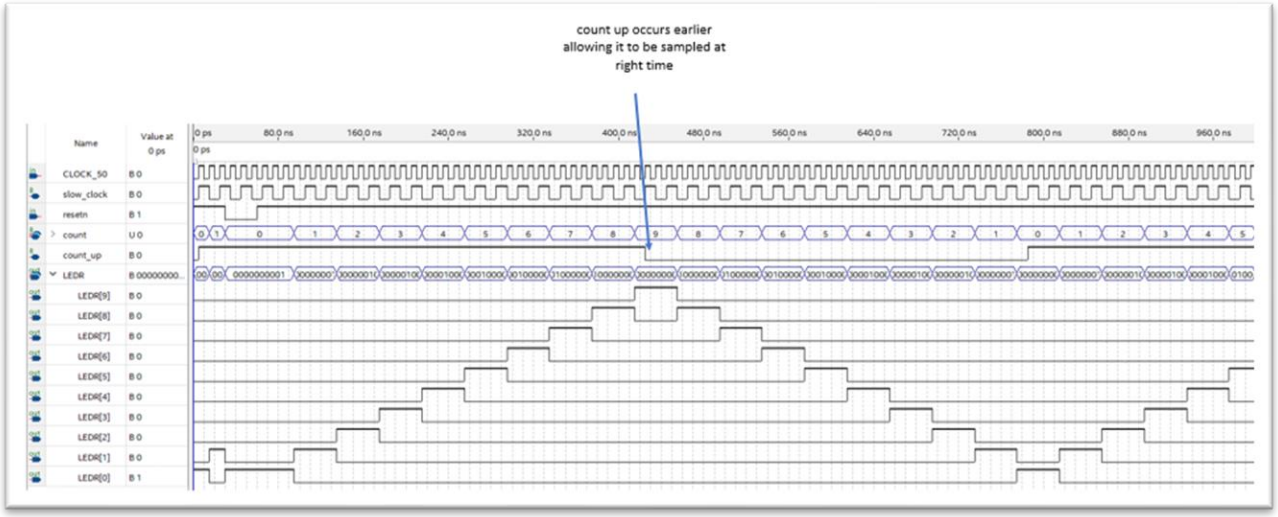

*Figure **11**: Corrected sequencing waveform*

Bonus!!!

Now that you have completed the Light Rider lab, click-and-hold the blue buttons labeled KEY3 through KEY0. If you have a touchscreen, try holding down multiple buttons at once.

Note: If LabsLand times out, you can synthesize and upload as many times as needed.

Can you modify the code to show **A**, **b**, **C**, and **d** instead of **0**, **1**, **2**, and **3**?

Place SW9 in the on position and the far left 7-segment display will illuminate. Prototype your character designs using SW6 through SW0 to control which segments are illuminated before you code them into the light_rider.sv source file in the IDE.

Thanks for taking the time and learning how to develop Intel FPGA products. We hope you found this lab interesting.