Lab Assignment 4: Stop-Watch Design

ECE/CS 3700

Fall 2024

Due date:
Checkoff in your respective lab sessions during the week of Oct 30.
Lab report is due Sunday Nov 10.

I. Laboratory Objectives

In this lab, you will design and implement a stop-watch on the DE10-Lite board.

So far, we have used Verilog to design only combinational logic using device instantiations (structural code), assign statements (functional code), or the always construct (behavioral code). Now that you have the basics down, it's time to take on a more challenging problem. Designing sequential circuits that have memory.

You have the freedom to design the circuit however you wish meaning all the important design decisions are up to you. You must figure out how to simulate a sequential circuit and generate a clock within its testbench.

As you read through the specifications, think about the following challenges you'll need to overcome:

1. You need a method of counting numbers.
2. Your circuit needs a clock. You also need to observe it in real time. However, the clock generated by your board toggles at 50 MHz (way to fast to be observed in real time).

II. Design Specifications

Your design will be a single (decimal) digit stop-watch with inputs for reset, start/resume, and stop/pause.
The functions for each of the inputs are as follows:

- Start/Resume: This button is used to trigger the counting on the stop-watch. When this input is activated from an initial state, the watch advances every second counting from 0 to 9 and then loops back to 0. The counting process should just keep on running until and unless any other inputs (stop or reset) are activated. If start/resume is activated from a previously steady display on the watch, then the watch resumes counting every second from that particular display value.

- Stop/pause: This button is used to stop the stopwatch counting and to hold the display. When this button is pressed the stopwatch stops counting at the very instance and displays the time (in seconds) continuously, until any other input (resume/reset) is pressed.
- Reset: This is used to reset the stopwatch display to 0. When reset is active, irrespective of the state of your counter, your stop watch is reset to zero (the display should read 0) and should hold this value zero until the *start/resume* button is activated.

A. Input-Output Interface

Use the push buttons (Key0 and Key 1; Refer to the DE10-Lite user manual) available on the FPGA board for the start/resume and stop inputs. Use a slide switch for the Reset input, and the 7-segment display LED on the board to display the output. You are to program the FPGA on the DE10-Lite board with the necessary code to achieve the above stopwatch design.

How to Track Time? Use the internal clock on the DE10-Lite board as the reference for computing time. *Hint: If I have a clock running at 60 MHZ, how many positive edge triggers will I observe in 1 second?* The intellectual challenge here is to do the necessary "clock division" within your Verilog code to increment the counter *every second*. The reason we want to "divide" our clock is because the DE10-Lite board uses a clock generator circuit to output a very stable 50MHz clock (square wave, 50% duty cycle) which is a very high frequency for our application. We need to divide this clock to generate a 1Hz signal that we can use in our application. Refer to Sec 3.2 of the DE10-Lite User manual.

III. The Assignment

- Design the above stopwatch.
- Write behavioural Verilog descriptions for the stopwatch.
- Write a testbench to simulate your design "exhaustively".
- Synthesize the design and observe the synthesis, map and place-and-route reports. What new information do you find in these reports vs combinational synthesis (from previous lab assignments)?
- To get checked off, download your implementation onto a DE10-Lite board and demonstrate its correct operation to the TA.

Report: As usual, your report should have a brief write-up of your objective, computations (if any), your approach, the Verilog Code, observations, results, a block diagram on how you designed the hierarchy, and conclusions. Attach appropriate code, test benches, waveforms, or whatever you deem necessary. It is a good idea to break the design into individual components and test them before integrating into the final design.

Tips for completing the Stopwatch lab.

You have two weeks to complete this lab. I recommend keeping up with (or ahead of) the following schedule:

1) *Week 1:*

- Implement the BCD-to-7 segment display. It is given in the book and is also used in the Verilog installation tutorial on Canvas. It would be a good idea to test your implementation by flashing it onto your board.

- Design and implement a “CLK DIVIDER” circuit in Verilog.

- Think about how you want to design your stopwatch. Including how you are going to count and remember numbers. Here are some ideas:
  1. You could design a '+1 adder' and instantiate a DFF to remember the current number as shown in figure 1.
  2. You could contain the problem into two concerns. One for counting, and one for controlling if the counter is stopped or running.
  3. You could write a Finite State Machine. Either have two states (counting and idle) and a 'count' variable to drive the outputs or you could have ten states (one for each digit) and only concern yourself with responding to changes in the input signals.

My advice is that method 2 should be easiest for you but if you're willing to study a little extra, going with method 3 and learning Finite State Machines now would be beneficial. Your lecture will cover them in a few weeks and lab 5 also uses one so it makes little difference which method you choose here.

The choice is up to you. You are the designer in lab 4. Have fun!

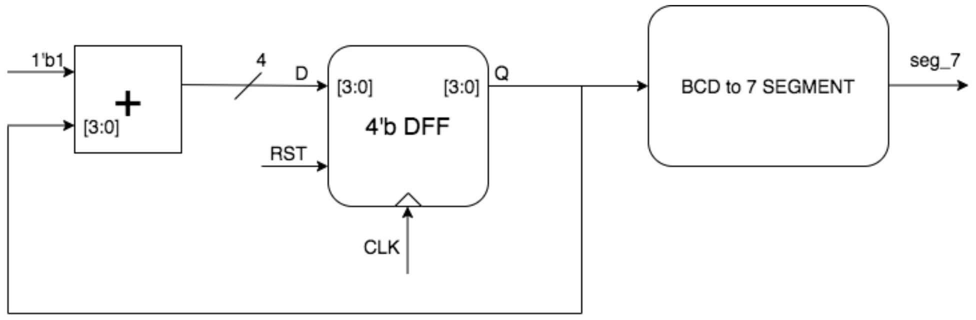

Figure 1: DFF and '+1 adder' approach

2) *Week 2:*

- This lab can be challenging so it is better if you already have a strategy at the end of the first week. Use the second week to implement your chosen strategy. Also don't forget to WRITE A TESTBENCH THAT VALIDATES SOMETHING NON-TRIVIAL ABOUT YOUR CIRCUIT!!!! It is up to you to decide what to test. You are all experts in Verilog by now so it shouldn't be a problem for you.
- Download the working stopwatch onto your board and get checked-off with the TAs.