ECE 3544: Digital Design I

Project 4 – Design and Synthesis of a Synchronous Finite State Machine

*Read this specification in its entirety before you begin working on the project!*

*Read the Assignment page on Scholar where you downloaded the project materials!*

**Honor Code Requirements**

You must do this assignment individually. The following represent unauthorized aids, as the term is used to define cheating in the Virginia Tech Undergraduate Honor System Constitution:

* Discussing *any aspect or result* of your design with *any person* other than your instructor and the ECE 3544 GTAs. This includes but is not limited to the implementation of Verilog code, as well as the supporting details for that code – state diagrams, state tables, block diagrams, *etc.*
* Using *any design element* – including Verilog code – from *any printed or electronic source* other than your course textbook and those sources posted on the course Scholar site.

*You may re-use any of your own original Verilog code. You may also use the* buttonpressed *module that was provided to you in Project 3B.*

All code submitted is subject to plagiarism checking by MOSS (theory.stanford.edu/~aiken/moss/). Any copying flagged by MOSS will be treated as Honor Code violations and submitted to the Virginia Tech Undergraduate Honor System.

**Project Objective**

In this project, you will implement a synchronous finite state machine that operates as a stopwatch with split and lap time functions. You will gain proficiency in the following:

* Using techniques to derive state machine models from natural language specifications.
* Using state machine models to produce synthesizable Verilog modules.
* Designing and implementing synchronous counters having various control and output functions.
* Designing interacting synchronous finite state machines.
* Implementing larger-scale synchronous systems via synthesis from smaller finite state machines

You will implement your top-level module on the Altera DE1-SoC board, so you will gain greater practice with assigning FPGA pins to the module’s input and output ports, and with using Quartus to synthesize hardware from Verilog models.

**Requirements**

*The DE1-SoC board IS REQUIRED for this project.*

You must have the current version of ModelSim ALTERA STARTER EDITION and Quartus II Web Edition 14.1 installed on your computer. The instructions are done using these versions of ModelSim and Quartus. While the directions may be consistent with other versions of ModelSim and Quartus, you must use the versions indicated above.

**Project Description**

In this assignment you will design and implement a stopwatch with split and lap time functions so that it works on the DE1-SoC board. The block diagram below is a *rudimentary* representation of the top-level module.

HEX4 (mode indicator)

HEX3 (1 m)

HEX2 (10 s)

HEX1 (1 s)

HEX0 (0.1 s)

**Stopwatch**

**System**

KEY[3] (reset)

KEY[2] (mode)

KEY[1] (start/split/lap)

KEY[0] (stop/clear))

CLOCK\_50 (clock)

*Submit a detailed block diagram representing your implementation of the system as part of your report.*

*In Quartus, choose* **Tools > Netlist Viewers > RTL Viewers** *to view the RTL schematic that represents your top-level Verilog module. You may submit a screenshot of this schematic if you believe that it is sufficiently detailed, simple to follow, and descriptive of the structure of your implementation. It might interest you to view the structure of the instances in your top-level module; clicking the boxed plus sign on any instance allows you to see the structure of that instance.*

For this project, the basic module that you will receive only represents the inputs and outputs of the DE1-SoC board. The system will not require all of the inputs and outputs that the top-level module declaration includes on its port list or that are listed in the port declaration. You may use the unused inputs and outputs for any testing or debugging purpose that you decide is appropriate. You may also remove them from the port list of the module declaration and from the port declaration.

You will be responsible for writing all of the Verilog that implements the design elements that comprise your stopwatch system. *You may re-use the* buttonpressed *modules from Project 3B. You may also re-use any of your own original Verilog code.*

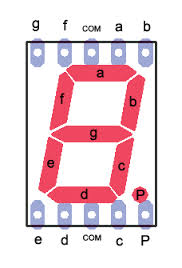
**System Interface**

The system has the following inputs:

* Clock: CLOCK\_50 – The DE1-SoC board has four 50 MHz clock signals. Use CLOCK\_50 as the clock for the sequential elements of your design. Do not use the switches or pushbuttons as clock signals.
* Pushbuttons: The system uses four pushbuttons as follows.
  + Reset: KEY[3] – The Reset button is an active-low system reset. Pressing and releasing the Reset button should return the system to its initial state no matter what its current state.
  + Mode: KEY[2] – Pressing and releasing the Mode button toggles the mode choice between Split and Lap time functions. The mode determines the behavior of the output in response to the Start/Split/Lap button. The mode should be able to be changed at any point in the operation of the stopwatch, e.g., while stopped or while counting.
  + Start/Split/Lap: KEY[1] – Depending on the mode of operation and whether the stopwatch is currently running, pressing and releasing the Start/Split/Lap button starts the stopwatch counting on the hex displays, or displays the current Split or Lap time for five seconds. If the stopwatch is not running, this button starts the count from its current value. If the stopwatch is currently running, the current Split or Lap time is displayed. While the Split or Lap time is displayed, the stopwatch should continue running in the background before being displayed again after five seconds.
  + Stop/Clear: KEY[0] – When the stopwatch is running, pressing and releasing the Stop/Clear button stops the stopwatch count. When the stopwatch is not running, pressing and releasing Stop/Clear clears the time to 0:00.0.

The system has the following outputs:

* Hex displays: The system uses five hex displays as follows:
  + Mode Indicator / HEX4 –The mode indicator identifies the mode of operation of the system. When the system is in Split time mode, the mode indicator should show an S (segments a, c, d, f, and g lit on the figure below). When the system is in Lap time mode, the indicator should show an upper-case L (segments d, e, and f lit on the figure below). The mode indicator should be on steadily whenever the stopwatch is stopped and should flash on and off at 2 Hz (that is, it should be on for one second and off for one second) whenever the stopwatch is running.



* + Stopwatch timer value / HEX3, HEX2, HEX1, HEX0 – The timer value shows the current value of the stopwatch or either the split or lap time if one of those times have been selected. The appearance of the displays also provide a general indicator of the level of operation of the system. Respectively, the four displays show units of minutes, tens of seconds, seconds, and tenths of one second. The timer value should appear in a manner that is consistent with a digital watch: The value should be displayed in decimal, with one minute (1:00.0) occurring immediately after 59.9 seconds. The maximum time displayed should be 9:59.9, which should roll over to 0:00.0.

**Split and lap times**

For many types of races, in addition to the overall time to traverse the entire race course beginning to end, participants keep track of the time from the beginning to a given point on course (the split time) and the time between two points on the course (the lap time). These times are useful for training and for maintaining a desired pace.

As an example, consider someone who is running on a 1km track at a constant pace of 4 minutes per kilometer. At the first kilometer, both the split time and lap time would be 4 minutes. At the second kilometer, their split time would be 8 minutes and their lap time would be 4 minutes. At the third kilometer, their split time would be 12 minutes and their lap time would be 4 minutes.

As a second example, consider someone running the same track but slowing down over time such that their time for the first kilometer was 4 minutes, the second kilometer was 5 minutes, and the third kilometer was 6 minutes. Then at the first kilometer, the split time and lap time would be 4 minutes; at the second kilometer the split time would be 9 minutes and the lap time would be 5 minutes; and at the third kilometer the split time would be 15 minutes and the lap time would be 6 minutes.

While it is convenient to think of laps as being of fixed length, the concept of lap times is also used for races that consist of several different stages, such as a triathlon, which involves running, swimming, and bicycling. For races with different types or lengths of stages, the lap time is the time to complete a particular stage.

**Method of Operation**

This diagram shows the basic manner in which the pushbuttons are used to control the operation of the stopwatch. It is not necessarily meant to represent a complete state diagram, but you might consider using it to help make the state diagram that you must make to control your system. The diagram is not exhaustive; additional constraints on the interactions of buttons and modes are also provided.

* The initial “state” of the system is the stopwatch clear (HEX3-HEX0 showing 0:00.0), the time is not counting, and the stopwatch is in split time mode (HEX4 showing S).
  + Pressing and releasing reset in *any* “state” causes the system to enter this “state.”
  + Pressing and releasing mode in this “state” causes the system to change to lap time mode (HEX4 showing L).
* When the stopwatch is stopped, if start/split/lap is pressed and released, then the stopwatch should start counting from its current value and in its current split or lap time mode (not shown in the figure).
* It should be possible to switch between split and lap time functions at any time by pressing and releasing mode (only two examples are shown in the figure). The mode determines the value displayed on HEX4 and what happens when start/split/lap is pressed and released. However, switching mode has no effect on the current stopwatch time value displayed on HEX3-HEX0. For example, if a split time is being displayed and mode is pressed, the split time will continue to be displayed for the remainder of its 5 seconds, but the next time start/split/lap is pressed and released, the lap time will be displayed.
* It should be possible to measure as many split or lap times as desired.
* Lap times should be measured since the last time that a lap time was displayed no matter how many mode changes or starts/stops of the stopwatch count have occurred. If it is the first lap time to be measured since the count was cleared, then it should be the time elapsed since 0:00.0.
* Pressing and releasing stop/clear should stop the stopwatch count at its current elapsed time no matter what time is displayed on HEX3-HEX0, i.e. it should be possible to stop the stopwatch while a split or lap time is displayed.

Stop/clear button

RESET

Start/split/lap button

Start/split/lap button

Mode button

*Repeat as often as desired*

Stop/clear button

Stop/clear button

Mode button

Start/split/lap button

Start/split/lap button

In general, for any other situation not described in this specification, your stopwatch should behave in a manner that would seem useful and reasonable to someone who wanted to use it to measure their performance in a race. This listing of “states” is not exclusive and need not represent the entire set of states of your actual state machine. Add states as you determine is appropriate to implement correct behavior of the system, but do not change any element of the system behavior described here.

**Design Tips**

* Design smaller aspects of this system individually. Implement and test one before moving on the next one. *Don’t try to implement the whole system before testing any of it*.
* Make a state diagram to represent the manner in which button presses change the mode and level of operation. *Include the state diagram for your “system controller” in your report.*
* Decompose the design into communicating finite state machines, e.g., a set of counters that receive direction from your system controller and might also provide information to the system controller, or counters that supply outputs that control the inputs of other counters.
* Make a block diagram of the system before you write any of the module. This will help you settle the manner in which the various state machines should interact. *Include a detailed block diagram representing your implementation of the system as part of your report. You need not break-out the structure of every block in your diagram, but a good guideline is to include a block for every Verilog module you write and instantiate in some other module.*
* Use the buttonpressed module from Project 3B to generate a one-clock-cycle enable signal when a pushbutton is pressed and released.

**Project Submission**

Write a report describing your design and implementation process.

* Discuss the decisions you made about implementing the elements of your design.
* Include a detailed block diagram of the interacting units in your system.
* Include a state diagram for the system controller, and the state diagrams of other units that might require them.
* Include waveforms showing the correct behavior of your design.
* Devise a simple test (presumably involving repeated trials) to test the accuracy of your timer. Include the results of your test and comment on the accuracy of your timer.
* Document any additions you make for debugging purposes, and describe your motivation for each debug operation.

The project assignment includes the validation sheet that the GTAs will use to test your design after the submission deadline. You do not have to go to the CEL to have your project validated. Instead, you should use the information in the validation sheet as one basis for testing your design before you submit your project.

Your project submission on Scholar should include the following items:

1. Project report in Word or PDF. When you create your report document, use the included cover sheet as the first page, and the included validation sheet as the second page. Include your Virginia Tech PID in the file name of your submission – for example, *project4report\_thmartin.pdf*.
2. A Quartus Archive containing the source files for your top-level module, any modules that the top-level module requires to function, and your test benches for the top-level module.

To create the Quartus Archive, choose **Project > Archive Project** after you complete your implementation. When prompted for a name for your archive, the default archive name will be the same as the original archive. *Append your Virginia Tech PID to the end of the filename*.

Your top-level module may be tested with our own secret test bench, so it is important that you use the module declaration provided with the archived project. You must also include the source files for every module required for your top-level module. Failure to do so will result in a grade of 0 for that portion of the project.

Grading for your submission will be as described on the cover sheet included with this description.