Georgia Institute of Technology

School of Computer Science

CS3220: Fall, 2015

Project 1: Timer

Version 1.1

Due: Friday, October 2nd, 2015 at 10:00pm

Project description:

In this project you will design and synthesize an egg timer for your FPGA development board. The timer will count down from a user-defined value to zero.

You will use the following components of the board:

(1) SW[7:0] (switches).
(2) KEY[2:0] (keys).
(3) LEDR[9:0] (red lights).
(4) HEX3, HEX2, HEX1, HEX0 (four 7-segment displays).
(5) CLOCK_50, the clock you will use for your design (50Mhz).

With the following user interface:

(1) KEY[0] is the reset button of the timer.
(2) KEY[1] is used to set the initial value of the timer.
(3) KEY[2] is used for stop/start.
(4) HEX3 and HEX2 will display the minutes, whereas HEX1 and HEX0 will display the seconds as the timer counts down.
(5) Flashing LEDR[9:0] will indicate the count down has reached zero.
(6) SW[3:0] will set the least significant BCD digit, and SW[7:4] will set the most significant BCD digit of the timer. Refer to step 2.

You need to design the timer such that it provides the following functionality:

Step 1: Reset
Whenever the *reset* button (KEY[0]) is pushed, all seven segment displays must display "0000". This is an initial state of the timer and whenever you push the *reset* button, the timer should go back to this state. No LEDs will be on or flashing in this state.

Step 2: Set Seconds
HEX1 will display the tens place while HEX0 will display the units place of the seconds. You will use switches (SW) to set the seconds. Refer to user interface bullet 6.

While you are setting the switches, the 7 segment displays -- HEX1 and HEX0 -- should change reactively, depending on the current input. When you push the *set* button (KEY[1]), the seconds are set and the timer goes to the next step.

Step 3: Set Minutes
HEX3 will display the tens place while HEX2 will display the units place of the minutes. You will use switches (SW) to set the minutes. Refer to user interface bullet 6.

While you are setting the switches, the 7 segment displays – HEX3 and HEX2 -- should change reactively, depending on the current input. When you push the *set* button (KEY[1]), the initial timer value is set.

Step 4: Run Timer
When the *start/stop* button (KEY[2]) is pressed, the time you set in the seven segment displays will start decreasing until it reaches to 0000. You should always be able to stop the timer by pushing the *start/stop* button.

Step5: Turn Lights
Right after the timer counts down to "0000", *all red lights* (LEDR[9:0]) should be flashing until you push the *reset* button. The flashing period of the red lights should be 1 second. That is, the LEDR[9:0] leds will be on for 0.5 seconds and then off for 0.5 seconds. The flashing will continue until you push the reset button.

Notes:

(1) This project is a team project so you must work together with your teammate.
(2) Your design should include a separate module for the controller of the egg timer. You should name this file TimerController.v
(3) Submission through t-square.
(4) Use Piazza for any questions.

What to hand in via T-Square:

There are two deliverables:

(1) Block diagrams of the design (PDF file)

You must submit block diagrams describing the entire design. The easiest way to do this would be to take a picture of a hand-drawn diagram. The file should be transformed into the pdf format before submission. You should also include a separate diagram in the same pdf file showing the state machine of the controller for your egg timer.

You should adhere to the following naming format:

Timer[Student's Full Name].pdf (e.g. TimerJongsePark.pdf).

(2) Quartus Project that includes Verilog files (ZIP file)

You must submit a zip file containing the entire Quartus project directory containing your Verilog code. The project directory name and the zip file name should adhere with the following format:

Directory: Timer[Student's Full Name] (e.g. TimerJongsePark)
Zip file: Timer[Student's Full Name].zip (e.g. TimerJongsePark.zip)

Good luck.

Annex: Verilog code for BCD to 7-segment conversion:

You can use the following code for the BCD to 7-segment conversion:

```
module dec2_7seg(
	input [3:0] num,
	output [6:0] display
);
	assign display =
		num == 0 ? ~7'b0111111 :
		num == 1 ? ~7'b0000110 :
		num == 2 ? ~7'b1011011 :
		num == 3 ? ~7'b1001111 :
		num == 4 ? ~7'b1100110 :
		num == 5 ? ~7'b1101101 :
		num == 6 ? ~7'b1111101 :
		num == 7 ? ~7'b0000111 :
		num == 8 ? ~7'b1111111 :
		num == 9 ? ~7'b1100111 :
		7'bxxxxxxx;   // Output is a don't care if illegal input
endmodule
```