Department of Electrical and Computer Engineering
University of California, Davis

Lab 7: A 10-LED version of "Whack A Mole"

Details not specified in this lab should be chosen by you and stated in your lab report.

I. Prelab

Re-read the following documents posted on the course web page:

- *Control Circuits and Counters* – see new Ring Counter slide
- *Finite State Machines*
- *Interfacing Input Signals*

[10 pts] Perform and document a preliminary design of the BCD converter and Remapper including a block diagram and preliminary verilog.
[5 pts] Determine how many counters you will use, their format (unsigned, 2's complement, one-hot, etc.), and size in bits.

II. Project Overview

This lab is a 10-LED version of the arcade game "Whack A Mole". https://youtu.be/VoP1E9J4jpg https://youtu.be/fP0ETbGl6Ns

The board-level input and output switches and displays for the design are:

- KEY[1] *reset* – resets all state and begin game
- KEY[0] *hammer* – the "hammer" used to whack the mole (use KEY[0] since it is more out of the way)
- (2) SW switches Choose 2 switches to control the length of the Smoothing Filter
- (10) LEDR These show both the location of the "mole" and allow the user to position their "hammer"
- HEX5 FSM *state* in any format you like. This display can be distracting, so feel free to control it with an available SW switch to optionally blank it (feature not required).
- HEX4 The "game" number (1–4), or re-display of the four game times (A–D)
- HEX2–HEX0 timer shown in BCD

The game works by the following steps:

1. Press KEY[1] to reset the system
2. The HEX3–HEX0 displays show "– – – –" (decimal point is ok). The LEDRs are off.
   It pauses in this condition for exactly 3 seconds.
3. For exactly 0.2 seconds, the LEDR corresponding to the location of the "mole" lights up. A timer of the form X.XX (in seconds) appears on HEX2–HEX0, begins counting up at the beginning of this 0.2 period, and counts with a resolution of 0.01 seconds up to a maximum of 9.99 seconds.
4. Next, the position of the "hammer" is shown by one of the ten LEDRs. The user quickly positions the hammer over the mole's position using the on-board accelerometer, and then presses KEY[0] to whack the mole. Pressing the "hammer" at the correct LED position stops the timer. The timer also stops if the time reaches 9.99 seconds.
5. After a hammer press at the correct location, the measured time is shown on HEX2–HEX0 for exactly 3 seconds.
6. The game then automatically repeats by returning to step #2, for a total of four games.

7. At the end of the fourth game, the system enters a 4-step loop that runs until the system is reset, with LEDRs off.
   A. HEX2–HEX0 shows time of game 1 HEX4 shows "A"
   B. HEX2–HEX0 shows time of game 2 HEX4 shows "B"
   C. HEX2–HEX0 shows time of game 3 HEX4 shows "C"
   D. HEX2–HEX0 shows time of game 4 HEX4 shows "D"

The location of the "mole" should appear random to the user. Do this by building a ten-state counter that runs freely and is sampled during one clock cycle at the beginning of each game.

When pressing the KEY[0] "hammer", only the first cycle it is pressed may count. In other words, it will not be possible to hold the KEY[0] button and rotate the board to get the same effect as hitting the hammer on all ten locations.

III. BCD Converter version 2

Starting with your BCD-to-7seg-display converter from Lab 6, modify it with the following minor changes:

- input a 10-bit **unsigned** number (range 0 to +1023) rather than a 10-bit **2's complement** number (range –512 to +511)
  - numbers greater than 999 (999 to 1023) should display as "999"
- turn on the decimal point in the third digit so the display has the form: X.XX

As before, an additional input, *nothing*, produces a dash on the four HEX displays "– – – –" when it is high. The input *outofrange* can be deleted.

IV. Smoothing Filter

Use your smoothing filter from Lab 6.

V. Remapper

Design a module which converts the 16-bit 2's complement *smooth_out* to a 10-bit one-hot output vector *board_posit*. Create a mapping that works well for your board's accelerometer chip. If your chip has an output range [–250, +250], then a reasonable mapping would be something like:

smooth_out values	*board_posit*
smooth_out ≤ –80	0000000001
–80 < *smooth_out* ≤ –60	0000000010
–60 < *smooth_out* ≤ –40	0000000100
....	....
+60 < *smooth_out* ≤ +80	0100000000
+80 < *smooth_out*	1000000000

VI. Memory array to remember the times for each game

A small memory array of four (or a few more if convenient) 10-bit words holds the final times for games 1–4. Circuits store the times at a convenient time during each game. Hint: use "combinational (asynchronous)" read port(s).

VII. Clocking and Timing

Follow the instructions for setting up the three necessary clocks in the accelerator's tutorial.

In this lab, all of your circuits must use a 25 MHz clock.

As described in the Accelerometer's tutorial, the accelerometer's output values change at approximately 50 Hz and they are synchronized to the main clock which is 25 MHz in this design. Therefore, new values of *smooth_out* must also be generated at 50 Hz. Hint: do this by using enable-able registers that are enabled by a circuit that produces a single-cycle pulse 50 times per second.

VIII. Testing in Simulation

Design and write a testbench for your design that exercises several different complete games.

Once your test benches are working correctly, demonstrate them to your TA and have them checked off.

Strong recommendation: test modules *one at a time* starting with the ones nearest the inputs. This can be done with a single testbench for the complete system in cases. For example, test the smoothing filter first, then write a different set of inputs to test the remapper, etc.

Suggestion: implement multiple independent tests that are shorter in length and focus on specific features.

Suggestion: divide the clock by a much smaller amount in simulation.

IX. Implementation and Verification on the DE10-Lite

Once your design is working correctly on the DE10-Lite FPGA board, demonstrate it exercising your module to your TA and have it checked off.

X. Extra Credit

[10 points] Add a Step #7.E to the loop in Section II composed of the following:

E. HEX2–HEX0 shows the average of times for games 1–4. HEX4 shows "E"

Submitted Work [200 pts total]

With the exception of instructor-provided code, all work must be yours alone.

[15 pts] **Prelab**

[160 pts] **Lab Checkoffs**

- [40 pts] Demonstrate your ModelSim simulation to your TA
- [40 pts] Demonstrate your design compiling and operating on the DE10-Lite board, to your TA
- Your TA will ask you to generate and will record a `certutil` hash of your top-level files. This hash must match the hash of the files you upload to canvas so no file modifications are possible after your demo. The hash is calculated using the following commands:
    1. in Quartus, double click a few key modules determined by your TA, into the Project Navigator Hierarchy pane to open its Verilog
    2. right click the file's tab and select Copy Full Path
    3. open a Command Prompt window (by typing "cmd" into the Windows search bar)
    4. type "`certutil -hashfile` *`filepath`*" where *`filepath`* is the file path that was copied earlier

[25 pts] **Lab Report**

- Draw a "pipelined block diagram" of your design.

Upload the following two files to Canvas by the end of your lab session—this is essential to receive any credit for the entire lab.

I. A single Zip file. Submit all Verilog hardware and testbench code that you wrote in a single zip file. Do not include any code that you did not write such as files generated by Quartus or IP components.

- Make a folder on your computer
- Copy all files to be submitted into the folder
- "zip" the folder into a single .zip file
- Log onto Canvas, click Assignments, find the correct lab number
- Upload the .zip file

II. A single PDF or Word file.

1. Any required diagrams
2. Required hardware verilog
3. Required testbench verilog
4. Text and/or waveform printouts

2021/06/02 Minor typo: HEX2 should be HEX3 in step 2
2021/05/24 The game description step 2 should have 4 dashes not 3
2021/05/19 Minor wording clarifications
2021/05/18 Posted