Department of Electrical and Computer Engineering
University of California, Davis

Lab 6: An Add/Sub Calculator with an Accelerometer-Based Input

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

I. Prelab

Read the tutorial "*Tutorial: Using the accelerometer on the DE10-LITE board*" posted on the course web page. Re-read handouts "*Finite State Machines*" and "*Variable-frequency clock hardware*."

[10 pts] Perform and document a preliminary design of the BCD converter and smoothing filter including a pipelined block diagram and preliminary verilog.

II. Project Overview

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

- KEY0 *reset* – resets all state and begins operation
- KEY1 *load* – loads the current value shown on the display
- 2 SW switches Choose 2 switches to control the length of the Smoothing Filter
- 5 SW switches Choose 5 switches to select the value shown on the calculator's display:
  smooth_out
  operand1
  operand2
  operand1 + *operand2*
  operand1 – *operand2*

  Show the indicated value when one of the five switches is high. Show four dashes if the switches are in any other state. Show "8 8 8 8" if *smooth_out* < –512 or *smooth_out* > +511, which will be visible during simulation but depending on your specific accelerometer, may not be visible on your board.
- HEX display all values are shown in BCD

The on-board accelerometer is used to enter two 2's complement values into a simple calculator which shows the sum and difference on the HEX display in BCD.

The calculator works as follows: KEY0 resets the board. After reset, LED0 lights up indicating the next press of KEY1 will load the value shown on the Hex displays into a register *operand1*. The FPGA board is tipped to generate values that typically range between [–500, +500]. After KEY1 is pressed, LED0 turns off and LED1 lights up indicating the next press of KEY1 will lock the value shown on the Hex displays into register *operand2*. After KEY1 is pressed a second time, the process repeats allowing the user to alternately load *operand1* and *operand2*.

III. BCD Converter

A combinational logic block inputs a 10-bit 2's complement number and outputs a minus sign which is high only when the input number is negative, plus 3 BCD digits (7 segments each), for a total of 22 outputs. An input, *nothing*, produces a dash on the four HEX displays "– – – –" when it is high. Another input, *outofrange*, produces the pattern "8 8 8 8" when it is

high. Your design must be purely combinational—not iterative or recursive as is often found in designs on the internet. As mentioned previously in class, do not use the verilog "/" divide or "%" modulus operators.

IV. Smoothing Filter

Without any filtering, the accelerometer's output value will jump around in an unstable manner. To stop this jittering, implement a smoothing filter in its own module with a selectable length chosen by 2 SW switches:

- 00 No filtering – *smooth_out* = last accelerometer sample
- 01 Filter length = 2; sum = add last 2 accelerometer samples; *smooth_out* = sum >> 1;
- 10 Filter length = 4; sum = add last 4 accelerometer samples; *smooth_out* = sum >> 2;
- 11 Filter length = 16; sum = add last 16 accelerometer samples; *smooth_out* = sum >> 4;

Name the output of the smoothing filter *smooth_out*.

V. Clocking and Timing

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

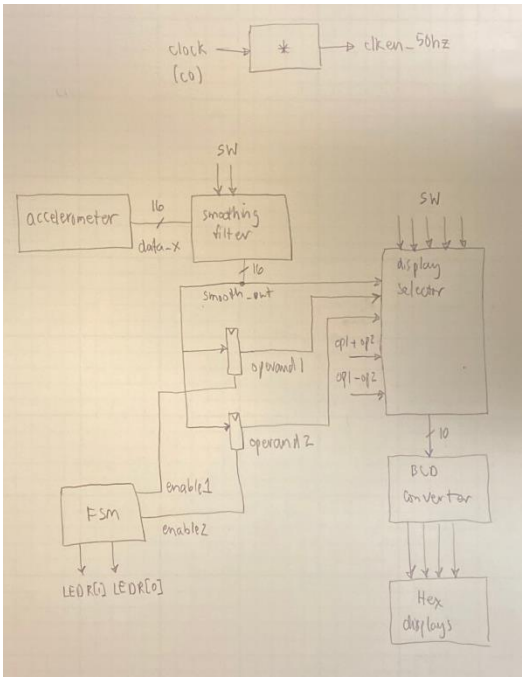
In this lab, all of your circuits must use a 25 MHz clock.

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. To match this data rate and to make the display easier to read, some circuits after the accelerometer's output must process data at 50 Hz also. Hint: do this by using enable-able registers that are enabled by a circuit that produces a single-cycle pulse 50 times per second, or possibly use *data_update* which is output by the accelerometer's module.

Accelerometer outputs are generated at 50 Hz and new values of *smooth_out* must also be generated at 50 Hz.

VI. System Block Diagram

The figure to the right shows the interconnections of the major components.



VI. Testing in Simulation

Design and write a testbench for your design that exercises all major functions.

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

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.

VII. 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.

VIII. Extra Credit

[5 points] Add a flashy (dynamic) pattern on the Hex display when more than one of the five SW switches is high.

Submitted Work [100 pts total]

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

[10 pts] **Prelab**

[80 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

[10 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

2021/05/18 "8 8 8 8" display
2021/05/12 Posted