Laboratory Exercise 3
ECE 441/541

Spring 2025

Due: April 1, 2025 by 11:59 PM

Notes on lab grading. The lab report will contribute 40 points to the lab grade. The lab workspace will contribute 60 points to the lab grade. The individual parts of a lab will have equal point contributions unless otherwise noted.

Here are the objectives for this lab:

- Understand how to synthesize state machines suitable for FPGAs
- Understand basics of computer arithmetic
- Understand how to implement simple a dataflow model

Background.

One of the things we suggest FPGAs can have an impact is in performing mathematical calculations. Of course, we understand the basic calculations, addition, subtraction, multiplication, and division. Further, it is important that we understand how these operations can be used on an FPGA platform. In addition, understanding how complex transcendental functions are calculated is also important. Consider a scientific calculator. We require the four basic operations, but we also need more sophisticated functions. We can also envision the need to group calculations using parentheses or using the stack based reverse Polish notation (RPN).

Part I

In Part I, you are going to create a simple digital calculator in a new design named **`Part1`**. The calculator takes as input two 8-bit signed two's complement operands and outputs a 16-bit signed two's complement result for addition, subtraction, and multiplication. For division, the calculator outputs an 8-bit quotient and 8-bit remainder, both signed two's complement numbers.

The calculator operates using a 50MHz clock which is used to store values in registers for calculation and also for the result. Note that all registers must be clocked using the 50MHz clock. Operands will be input two hex digits at a time on $\mathbf{SW}_{7..0}$. Each 8-bit quantity is entered by setting the switch positions and pressing $\mathbf{KEY}_0$. After pressing $\mathbf{KEY}_0$, values should be displayed on **`HEX1`** and **`HEX0`** and the remaining displays should be blank. The calculator function is performed by pressing $\mathbf{KEY}_1$. The hardware reset for the calculator is performed by pressing $\mathbf{KEY}_3$. Once the values are entered, calculator function is set with the switches. Arithmetic operations are performed by setting the switches as follows and then depressing $\mathbf{KEY}_1$.

- $\mathbf{SW}_0$**`='1'`**: addition
- $\mathbf{SW}_1$**`='1'`**: subtraction
- $\mathbf{SW}_2$**`='1'`**: multiplication (create a combinational multiplier). Multiplication should detect overflow and display an E in **`HEX5`**, but show the resulting value in the display.

- **SW$_3$='1'**: division (create a combinational divider). Division should detect a divide by zero, display an E in **HEX5**, and blank the rest of the display.

Note that the switches have dual use depending on whether an operand is being entered or you are performing a calculation. In addition, when setting the operation, the switch settings must be mutually exclusive, i.e. only one of **SW$_{0..3}$** can be on at any given time. Should more than one be on, you should report an error by displaying an E in **HEX5** and blank the rest of the display.

Your calculator must not be required a hardware reset to perform the next calculation. Furthermore, once the calculation is complete, the result must not change if the function switch positions are changed.

With the exception of the reset, all inputs should be synchronized to the clock using double sampling. Use **HEX3**, **HEX2**, **HEX1**, and **HEX0** to display results while blanking **HEX5** and **HEX4** unless otherwise an overflow occurs. For addition, subtraction, and multiplication, show the 16-bit result in the display. For division, display the quotient in **HEX3** & **HEX2** and the remainder in **HEX1** & **HEX0**. Note that you will need to clock internal registers in order to achieve the required functionality.

Name your top level VHDL file **part1.vhd** and use the VHDL entity given in Lab 2 with the entity name changed from **fpga** to **part1**. Simulate your model using Aldec. Simulation is really important for this part because your first implementation may not work exactly as you think. After you verify your design, synthesize your design using the Aldec design flow tab with Quartus as your back end. Include in your lab report the utilization of the various FPGA resources and and note anything that appears unusual.

Part II

FACTORIAL SHOULD BE A SEPARATE MODULE WITH START AND DONE SIGNALS

FOR +, -, *
FOR ÷ → CALCULATE FACTORIAL OF REMAINDER.

In Part II, you are going to add the ability to calculate the factorial function. Note that calculating factorial is iterative and you are required to develop a SM chart and datapath to define the state machine as well as other registers to facilitate performing the calculation. The function is performed on ~~either the most recent value entered or~~ the value resulting from the last calculation when **SW$_4$='1'** and **KEY$_1$** is depressed. Internally, you should be able to calculate the factorial whose value can fit in a 64 bit unsigned value. Should the result exceed the maximum value that can be displayed, you should display E in **HEX5**, O in **HEX4**, and blank the rest of the display. After the calculation, you should display the answer four hexadecimal digits at a time where **HEX5** displays which group of 16 (0, 1, 2, or 3) and **HEX4** is blank**,** cycling every second until **SW$_4$='0'**.

Create a new design, part2, and copy your files from part1. Name your top level VHDL file **part2.vhd** and use the VHDL entity given in Lab 2 with the entity name changed from **fpga** to **part2**. Simulate your model using Aldec and verify that your model correctly calculates the factorial function for several values and is able to use this value in subsequent calculations. After you verify your design, synthesize your design using the Aldec design flow tab with Quartus as your back end. Include in your lab report the utilization of the various FPGA resources and note anything that appears unusual.

Other Requirements

Include in your lab report a comparison of the FPGA utilization for Parts I & II. Recall, the upload link is posted on Blackboard.

Please address the following in your workspace and/or lab report:

For submissions that include Aldec simulations, be sure to make sure the waveforms are properly saved and the waveforms appear in the workspaces that you submit

Provide the details of your design in Part II. Note also the largest factorial that can be computed.

- Our expectation of a scientific calculator includes many other functions, such as e^x, sin(x), cos(x), and etc. Describe why these functions would be challenging to implement given the calculator specification described above.
- Is the utilization higher, lower, or just what you expect? Why?
- Is the fastest clock frequency your model can operate at higher, lower, or just what you expect? Why?

Provide at least one suggestion on how calculator defined in this lab can be modified to be useful. Assess the complexity calculator of your suggestion.

- Note anything interesting or unusual you observe.