**ECE 6740 Advanced Embedded Systems \_\_\_Michael Muller\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Prof. D. Hanna Name**

**Winter 2024**

**LAB # 2**

PART A

In Lab 1, we became familiar with both the PS side and PL side of the Xilinx ZYNQ chip. In this lab, we will extend our design to include interrupts rather than having the processor poll the memory-mapped register for the answer. In Lab 1, we performed a basic combinational function, addition. In reality, it may take more than one clock cycle to reach the answer that you wish to pass to the processor and therefore an interrupt is ideal to interrupt the processor when the information is ready, allowing the processor to continue performing other tasks.

Design a state machine and datapath that will monitor a single GPIO that is receiving bits serially (connected to a button), for the sequence *10110*. This sequence is part of a protocol. After the 10110 has been received, a 16-bit number follows. Drive this sequence detector from a clock connected to another button.

So a sequence may be

0010101001010*10110*0000000001000000

This means that the value received is 64.

Take the Integer Square Root of the 16-bit number using the Integer Square Root algorithm from class. You will need to modify the Integer Square Root to accept 16-bit input which will require increasing the size of all of the registers and components in the design.

Pass the 16-bit number and the result to the processor through a memory-mapped register and raise an interrupt. Drive the Integer Square Root using the AXI bus clock.

Simulate this PL using Vivado. For now, make the interrupt a single pin output.

Side note: The Integer Square Root component was designed to run once; it ends in the done state and the *SqReg* and *DeltaReg* are initialized on *reset* to their starting values of 1 and 3, respectively. This lab does not require you to run it more than once in a single run. However, if you wanted to be able to compute the integer square root for an input and compute it again, you would either need to (1) control the reset signal for the integer square root separate from the global/bus reset so that your circuit could reset the Integer Square Root in between inputs or (2) change the Integer Square Root design to be able to explicitly load 1 and 3 into the respective registers while in the *Start* state which would add a MUX to each register that would need to be controlled and change the next state for the *Done* state to go to *Start* so it would be ready for another input in succession. This lab only requires you to run it once, so you do not have to do this modification but you may do so for brownie points.

PART B

Wrap your component from Part A in an AXI wrapper and import it into a project connecting it to an AXI Interface and ZYNQ Processor, with interrupts. Keep the sequence detector clock connected to a button as in part A, but use the AXI bus reset as a global reset for all sequential components.

Write a program that will count, every second, and display the updated stop watch time on the screen through stdout (the UART) using one of the built-in timers in the ZYNQ processor. This should be interrupt driven.

Write an interrupt subroutine that will take the 16-bit number and the result from the Integer Square Root computed in PL and place it on the screen through stdout next to the stop watch timer. So the screen will say:

00:32

00:33

00:34

Etc…

Then after the number has been received and integer square root calculated, it will read:

01:04

data recvd: 64, square root: 8

01:05

01:06

Download the entire design to the board using a button for the input GPIO and a button for the sequence detector clock. Test your design.

Submit the following items when you are finished:

* A copy of this lab instruction sheet
* A listing of the parts of the IP AXI interface that you modified
* A listing of your main c-files
* A waveform from the simulation in Part A
* A screenshot of the results from Part B
* A short narrative on challenges that you encountered and how you overcame them –or-a posting to the tips forum on Moodle.