**Yazan Halawa**

**Adam Hastings**

**ECEn 427 Lab 2**

**Instructor: Dr. Hutchings**

In this lab we constructed a clock that displays time via the terminal emulator via the UART. The time on the clock is set using the push-buttons on the board. Throughout this lab, we gained a lot of practice using interrupts on the Microblaze, the soft microprocessor provided by Xilinx.

Estimation of Time spent on the lab: 6 hours

1. First we had to enable interrupts. In order to do this, we had to connect our interrupts in hardware, and then do some stuff in our main code. In XPS, we implemented some Xilinx primitives. One of these was the *microblaze\_0\_intc*, or rather, the Interrupt Controller. We first connected the Fixed Interval Timer (FIT) to the interrupt controller. Based on the way we configured the FIT, this meant that our system would generate an interrupt 100 times per second. Secondly, we used the Interrupt Connection Dialog to connect the board’s push buttons to the Interrupt Controller as well. Thus pushing a button would also generate an interrupt. With our interrupts configured in hardware, the next part was to write the corresponding software.

We implemented most of our interrupt code in main. After initializing the GPIO and setting the push button peripheral as inputs, we had to enable the global and specific GPIO interrupts by calling the following two functions:

* *XGPIO\_InterruptGlobalEnable()*
* *XGPIO\_InterruptEnable()*

Next we had to set up the microblaze register handler, which is responsible for querying the interrupt controller to see which peripheral fired the interrupt and dispatching the correct interrupt handler. Naturally for this to work, we had to enable interrupts on the interrupt controller, which we did by calling the function

*XIntc\_EnableIntr(),* and passing the appropriate addresses for the registers corresponding to the FIT timer and push buttons peripherals. We also called *XIntc\_MasterEnable()*. Finally, we called the *microblaze\_enable\_interrupts()* to enable the microblaze to handle interrupts.

2.) Moving on the interrupt handler, we had to implement the following steps to have correct operation: In the dispatcher, we check if an interrupt happened by calling *XIntc\_GetIntrStatus().* If an interrupt happened then we acknowledge it and call the FIT interrupt handler if the FIT was the peripheral firing the interrupt, else the push buttons handler.

First for the FIT handler, we had to do two things: advance the time, and debounce the push buttons. For advancing the time, we simply had a global variable which kept the total seconds of time elapsed. Once a 100 FIT timer interrupts happen then we increment the total seconds of time elapsed, and update the time using simple arithmetic. For debouncing, we went through some calibration until we settled on waiting for 4 FIT interrupts to debounce the buttons. Finally, we had to add the functionality to set the clock using the buttons. We did so in the FIT handler by waiting one second, then if a button press persists, increment the appropriate unit of time every 50 FIT ticks (twice the normal rate).

As for the Buttons handler, we mostly just used it to grab the current state of the buttons for use in the FIT handler and resetting the button counters.