Lab 2: GPIOs + Basic MicroBlaze System

Blake Williams

09/20/2024

**Summary**

The goal of this lab was to Implement 2 GPIO Block designs inside Vivado, then connect these GPIO block designs to external peripherals such as the Switches, Buttons, LEDS, and RGB lights. From this block design (using Vitis) C code is written to interface with the Dip switches and the RGB lights. The goal is when the first switch (SW0) is turned on, it turns the first RGB LED on (LD0). Then if SW1 is activated then LD1 is activated. This continues until SW3 and LD3.

**Design and Implementation**

To Begin, first the Vivado Implementation had to be created. In Vivado’s Block Design two AXI GPIO Blocks were added. The first GPIO accepts outside connections from the led\_4bits and the push\_buttons\_4bits. With the second one accepting the rgb\_led and dip\_switches\_4bits. Once both blocks were added and the Connection Automation was run, the Base Memory addresses of both GPIO blocks were noted. After this the hardware was exported, and Vitis was launched. From Vitis a new Application was created on top of the new Platform. Using the provided C file as a starting point, the Dip switches and rgbLEDs Register memory location was defined and declared. This part was almost identical to the provided code, the only difference being the AXI\_GPIO\_1\_BASE\_ADDR was 0x40010000 instead of 0x40000000. Inside the main loop the Dip Switches and rgbLEDs Tri and Data registers were defined, similar to how the buttons and greenLEDs Tri and Data registers were defined. The main goal of the lab was to toggle the rgbLEDs when the corresponding switch was flipped. In order to do this a “probing” approach was taken. To start if a switch was flipped, the light emitted must be white. Since each rgbLED are 3 bits long, and each bit corresponds to either Red, Green, or Blue. Then by toggling all 3 bits to ON, a white light would be emitted. To achieve this a for() loop was defined which loops four times, one time for each dip switch. Inside this for() loop an if statement checks each bit in the \*switchData variable to see if any bits are 1. It achieves this by taking the \*switchData variable and using a bitwise “and” operator to compare an integer 1 to the current i location. An example would be if the\*switchData = 0010. The operation would look as follows.

Loop1: 0010 & 0001 = 0000

Loop2: 0010 & 0010 = 0010 // Active Dip Switch Detected

Loop3: 0010 & 0100 = 0000

Loop4: 0010 & 1000 = 0000

The Loop “probes” each of the dip switches. If the statement detects an active Dip Switch, it then does the following.

\*rgbLEDsData |= (0x07 << (i \* 3));

Because each of the rgbLEDs are 3 bits wide, and to get white light all three must be toggled on. We shift 3 bits containing ones, left by i \* 3. Where “i” is the current loop value. For example, if the \*switchData = 0010, the code above would shift “111” 3 bits left. Making the new \*rgbLEDsData = 000000111000 which will toggle LD1 on and be white.

**Conclusion**

This lab was a brain bending one. As it challenged you to think about programming in a whole new way. It was a struggle to wrap my head around these new concepts, but I enjoyed doing it. In struggling with this lab, I was forced to learn and understand how to interface with Memory Addresses, use Bitwise Operators to check if certain events occurred, and how to control outcomes when certain events did occur (Turning on a light when a switch was flipped). Along with the C coding, I better understand how to comprehend the Datasheets and use them to interface with physical hardware. Overall, I think this lab was difficult, but it was worth it being so hard.