**ECEN 449 - Lab Report**

**Lab Number:** 2

**Lab Title:** Using the Software Development Kit (SDK)

**Section Number:** 508

**Student's Name:** Samuel Fafel

**Date Due:** 02-16-2023

**TA:** Prajwal Holla

#### Purpose/Introduction:

The purpose of this lab is to introduce us to the Vivado Block Design Builder and SDK software during the development of a LED control system. Rather than program the FPGA board using Verilog like in the previous lab, we will write a program in C and have the SDK program the board accordingly.

#### Procedure:

Part 1:

1. Created a Block Design Diagram and added/edited the appropriate parts
2. Created an XDC file. Created a clock and assigned ports to appropriate pins for the LEDs on the FPGA Board
3. Generated and exported bitstream to SDK where we created a new application project.
4. Wrote C code which incremented a count every second and displayed that count on the LEDs
5. Programmed the FPGA with that code and ran it. Demoed to TA.

Part 2:

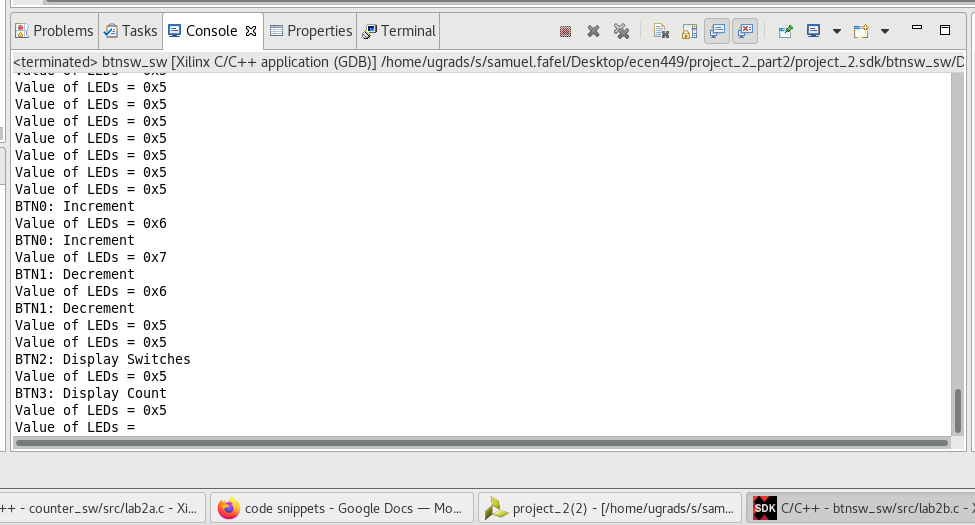
1. Created a backup of Part 1
2. Added an 8-bit Input GPIO block to the Design Diagram and assigned switches and buttons to the appropriate ports in the XDC file.
3. Generated and exported bitstream to SDK where we created a new application project.
4. Wrote C code which used the Switches and Buttons to change the status of the internal count or the LEDs status.
5. Programmed the FPGA with that code and ran it. Demoed to TA.

#### Results:

Part 1 of this lab was relatively straightforward. After programming the FPGA board via SDK, the count incremented every second and the LEDs displayed the count. The most difficult part was working with the Vivado and SDK softwares and figuring out how to fix the errors that often occurred between each step. Following the lab manual, one often runs into errors that are unexplained, and troubleshooting those can be very difficult with the complicated user interface of both Vivado and SDK.

Part 2 was much more difficult. I had a difficult time understanding exactly how to update the Block Design Diagram and the C code. For the C code, the hardest part was figuring out how to detect when a button or switch was being pressed. It took me a long time to realize that I had to look in the <xparameters.h> file, and even then I did not know what I was looking for. (Especially since there was an error the first time around and the inputs weren't included in the <xparameters.h> file.) It was only after restarting the entire SDK process that I was able to find the appropriate lines of code to include in my C file. Once it was programmed, the buttons performed their desired functions and the switches' status was displayed as desired, when desired.

Screenshot of TCL Console output:



Every second, the program printed the current status of the Count to the console and also printed if any buttons were pressed. Ex: Holding down BTN0 for 2 seconds incremented the Count from 0x5 to 0x6 to 0x7. Holding down BTN1 for 1 second decremented to Count back to 0x6. BTN2 Displayed the status of the switches via the LEDs on the FPGA board. BTN3 displayed the Count via the LEDs.

#### Conclusion:

The most important thing this lab taught me is how to troubleshoot errors in Vivado and SDK. It also taught me to pay close attention to detail in the Lab Manual, as the slightest typo can result in a tremendous headache while trying to find the error(s).

Troubleshooting and headaches aside, this lab taught me how to use SDK and use C to program hardware, which I was not previously aware was possible. Now that I have experience doing so, I hope to try some of my own experiments on my own time.

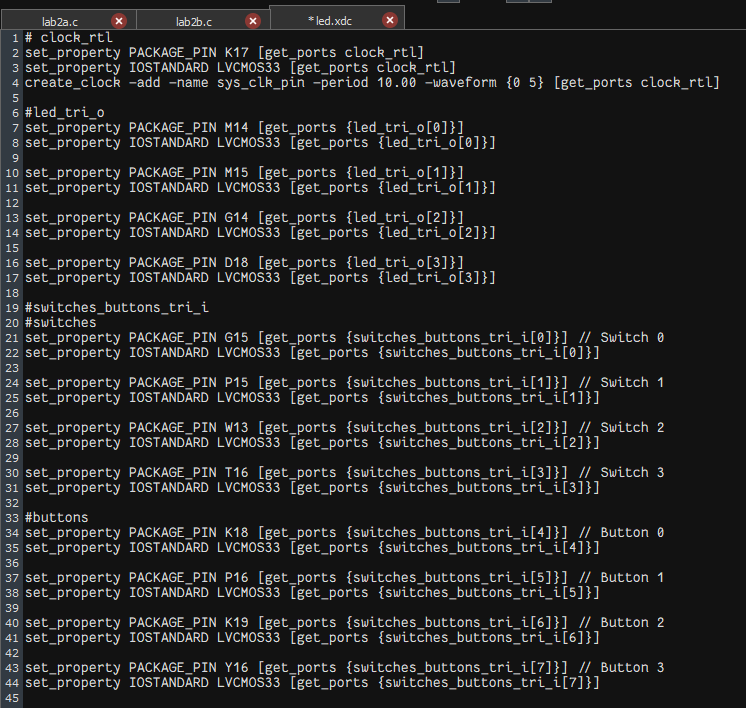
#### 

#### Questions:

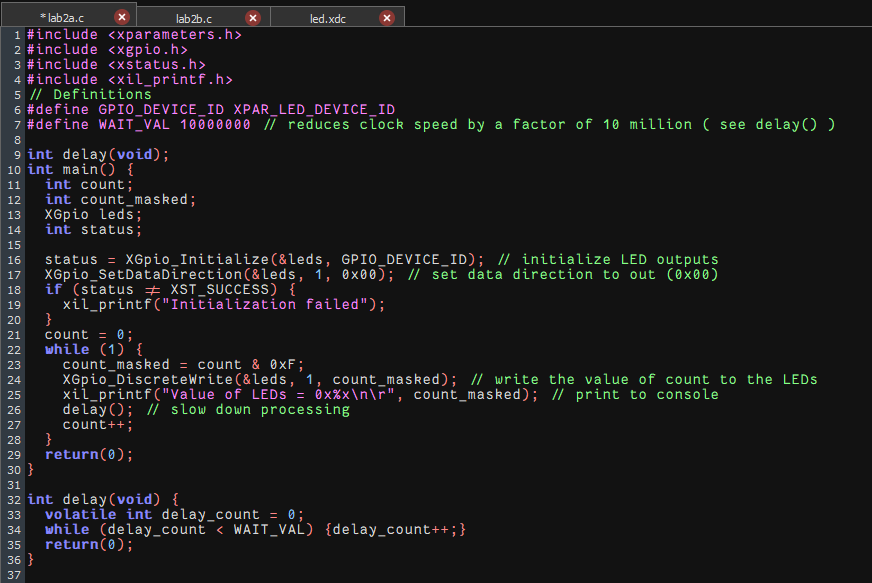
1. The count value in this lab is 10 times greater than that of the previous lab. 10 million clock cycles rather than 1 million works to better slow the output. My output in the last lab displayed rather quickly, although still slow enough to keep track of. The delay for-loop takes 10,000,000 clock cycles to execute one iteration.
2. It is declared as volatile because the value changes extremely quickly and may change at times unexpected by the program, since it is operating off of pure clock cycles.
3. The while(1) expression continues to loop forever, because the condition for the while loop is always true (1).
4. The purely hardware implementation was nice, but simple. Complex programs would be very difficult to do at a purely hardware level. Purely software is clean and efficient, but as obvious in implementation of the lab manual, can result in a lot of unforeseen errors unless one knows exactly what one is doing. The tools for the software implementation are more complex, resulting in increased complexity both in functionality and troubleshooting. Overall, I think that hardware implementation is easier for small, less complex programs, while software implementation is easier for more complex ones.

#### Appendix:

Edited XDC File with Switches and Buttons:



Part 1 C Code:



Part 2 C Code:

