LAB 6: An Introduction to Character Device Driver Development

Michael Mengistu

ECEN-449-508

Due Date: 3/16/2020

Introduction

The goal of this lab is to make a character device driver that interacts with my multiplication IP for a user line application. In doing so, I will make a character device driver that will interact with my multiply module which reads and writes to my multiplication IP. I will then make a user application that will read and write to my character device driver in order to test it.

Procedure

How I built my character device driver for the first part of lab:

1. Made a directory on my computer called modules
2. Created a multiplier.c character device driver that interacts with my multiplication peripheral and placed the file and all the files needed to compile it inside the modules directory
3. Created a makefile and used it to compile the multiplier.c file to make a multiplier. ko file
4. Booted up Linux on my FPGA using a SD card
5. Unmounted the SD card from the FPGA to transfer the multiplier.ko file onto the SD card
6. Placed the SD card back into the FPGA and mounted the files
7. Loaded the multiplier.ko file and viewed the output of the kernel module.

How I built the user line application to test my device driver for the second part of the lab:

1. Created devtest.c file in my modules folder that interacts with my character device driver to multiply numbers 0 to 16 and check if the result of the multiplication is correct.
2. Complied the devtest.c to make an executable called devtest
3. Unmounted the SD card from the FPGA to transfer the devtest onto the SD card
4. Placed the SD card back into the FPGA and mounted devtest
5. Ran the devtest executable to test if my character device driver is working correctly

Results

For the first part of the lab, I created a custom character device diver to read and write from my multiplication IP which was able to compile successfully. For the second part of the lab, I created a user application that uses the character diver I created to multiply numbers from 0 to 16 together to test my device, which also complied and ran successfully. The hardest part for me in this lab was creating the custom character diver. However, I was able to build it correctly by reading about the operating system's built in character devices and seeing how other example custom character device divers worked.

Conclusion

In conclusion, I was able to design both the custom character device diver and the user line application to interact with my multiplication IP. In doing so, I learned about divers and how they work on operating systems and how to create my own device diver. The knowledge I gained from this is important because I can use what I learned to create device drivers for other operating systems.

Questions

a) The ioremap command is required because you must map the multiplier's physical address with the kernel's virtual address to be able to use the multiplier on the kernel. This is because the multiplier hardware is in physical memory while the kernel is in virtual memory.
b) I think the overall time to perform multiplication in this lab is faster than in lab 3. This is because the multiplication in this lab works more closely to the hardware then in lab 3, which uses software, and hardware implementation is usually always faster than software.
c) The benefits of lab 6 implementation are that you have more ability to customize your design, the cost would be that you can cause fatal errors in your system since changing the operating system. The benefits of lab 3 implementation are that there would be less fatal errors you could to your system, the cost would be the performance time.
d) The explanation for both parts is that it's important because the memory acts like a stack, and because of this you must make sure you allocate and deallocate using a Last-in-first-out approach.