Nicholas Tann

Lab 6

ECEN 449-503

3/3/2020

Introduction:

The purpose of this lab is to guide students through the process of running a simple character device driver using a Linux kernel module on the FPGA. It will build off the last lab, where first we will need to get a Linux kernel module running on the board. The module will be used to initiate a device driver for user interaction with our multiplication peripheral from lab 3.

Procedure:

We will once again use the contents of lab 4 which includes the Linux kernel source and boot Linux on the board to read and write files to it and test out the mount operation on the SD card as we did in lab 5; we will, however, add the functionality for this module to read from and write to the multiplication peripheral. In part 1, all the reading and writing was done directly within our init_module() function. Part 2 is where we add the file operations to read and write (this is where the user can interact directly with the character device). The final part was to create a means of testing our character device using many multiplications and calculating new products every time we receive a user input.

Results:

After cross-compiling and multiplier kernel module. I was able to see the correct results of the multiplication, as well as the physical and virtual base addresses of the multiplication peripheral. I cross-compiled the kernel module in conjunction with the character device driver. I was able to register my device using the dynamically allocated major number. The last part was to develop

a way for user interaction with the device by means of the character device driver. This part required much debugging. The correct results were then displayed on the PICOCOM terminal.

Conclusion:

In this lab, I was able to get a simple Linux kernel module and character device driver up and running on the ZYBO board. I was guided through the process of creating a device driver whose purpose was to display the result of multiplying several numbers together. Using user input, the driver knew when to continue to read and write to the multiplication peripheral.

Questions:

1. Using ioremap() was required to map the physical address of the multiplication peripheral to a virtual address. Virtual memory is necessary because this way we can assume that our peripheral's addresses are located contiguously in memory. This may not be the case for the physical memory.
2. The original hardware in lab 3 was mapped directly to the ARM processor. Since there is no operating system in the way, our original lab 3 implementation may perform quicker. The interface may not be as nice as compared to lab 6 however.
3. As mentioned above the only negative is that in lab 6 with Linux as a middleman it may perform marginally slower. Our original lab 3 implementation may be harder to interact with because Linux device drivers are easier for the user to interact with the hardware, Reading and writing to the device is much easier to do in lab 6 than lab 3.