Alan Achtenberg

Lab 4 Booting Linux

ECEN 449 Sec:505

Due: 10/10/2014

Introduction:

In Lab 4 we added the hardware peripherals to our microblaze system required to install linux. Then we compiled the linux kernel and loaded it onto our compact flash card before booting it onto the fpga.

Procedure:

To set up the hardware we had to add a barrel shifter a multiplier and more to our system.

Then we had to had the RAM component of 256 MBs. Once we did this we changed the Instruction memory and the data memory to be only 8 KB instead of the 128 KB it was in previous labs. These memories are actually implemented the same way in the BRAM so there is no speed up to using one or the other.

Once the hardware system was complete, we had to step through the long process of creating our linux kernel. To do this we had to copy our linux device tree to the correct directory and setup our compiler to cross compile for the microblaze system instead of the x86. Finally after completing the set up we had to boot the linux system by inserting the mem card into the fpga.

Results:

Verilog and C Source Files

Conclusion:

Setting up the linux kernel is one of the most painstaking labs yet, but the power an operating system provides is unquestionable. It will allow us to do many things and improve our fpga as a development board by allowing us to directly run and test software on the operating system of the fpga.

Questions:

(a) Compared to lab 3, the lab 4 microprocessor system shown in Figure 1 has 256 MB of SDRAM.

However, our system still includes a small amount of local memory. What is the function of the

local memory? Does this “local memory” exist on a standard motherboard? If so, where?

The function of the local memory serves as an instruction/data cache for our processor. Usually this memory is in registers located either on the microprocessor chip or in very close proximity to the processor

(b) After your Linux system boots, navigate through the various directories. Determine which of

these directories are writable. (Note that the man page for ’ls’ may be helpful).

Test the permissions by typing ‘touch <filename>’ in each of the directories. If the file,

<filename>, is created, that directory is writable. Suppose you are able to create a file in one of

these directories. What happens to this file when you restart the XUP board? Why? Proc is not writable currently, but all of the directories are writable as long as you change either a have super user permissions or b use chmod to change modify the permissions of the folder. The file is lost because the data is not written back to the compact flash but it is written to the ram.

c) If you were to add another peripheral to your system after compiling the kernel, which of the

above steps would you have to repeat? Why?

Would have to repeat all the way from creating the device tree to compiling the kernel to creating the system ace. That is because the addresses of specific hardware must be known by the operating system for them to function correctly.

(d) Suppose during the kernel configuration process you left the field ‘USE BARREL’ set to 0.

Would the system still function? Why or why not?

Yes the system would still function but would be extremely slow for certain bitwise operations. Because it can be implemented by the alu.