ELECTRICAL & COMPUTER ENGINEERING
TEXAS A&M UNIVERSITY

ECEN 449: Microprocessor System Design

Lab 4: Linux Boot-Up on ZYBO Z7-10 Board via SD Card

Kylan Lewis
UIN: 719001131
ECEN 449 -504
TA: Ashwin Ashokan
Date: 10/16/20

Introduction:

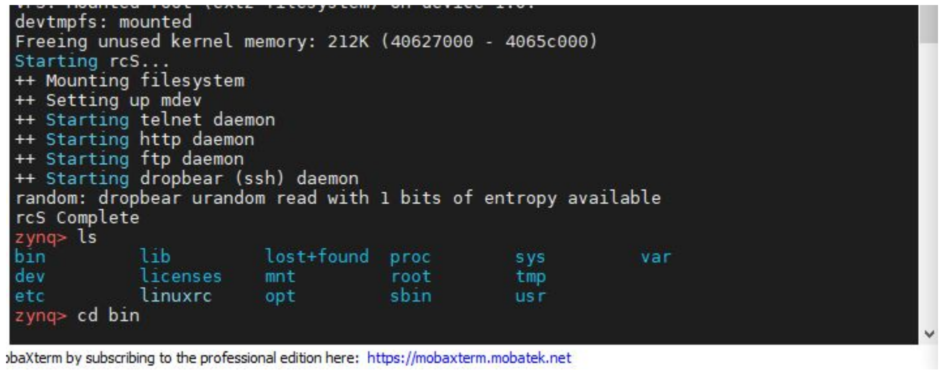
The purpose of this lab is for us to learn the process of getting Linux running on the FPGA. This was done using Vivado to build a Zynq Processing System that is suited to run Linux. Using the hard processor (ARM processor) on the Zybo board and using open-source tools we are able to load and boot Linux through the FPGA.

Procedure:

1. Create a new project and choose the ZYBO Z7-10 board as in Lab 3.
2. Add the IP of the ZynqQ7 Processing System then import ZYBO_Z7_B2.tcl in the Customize IP window.
3. Enable UART1, SD0, TTC0 and disable all other peripheral pins
4. Add multiply IP created from Lab 3 to the base system design.
5. Create HDL wrapper, generate the bitstream, export the hardware and launch the SDK.
6. Untar the u-boot.tar.gz and cross-compile the u-boot.
7. Create and building the FSBL project in SDK to make a BOOT.BIN file.
8. Untar the Linux-3.14.tar.gz file from ECEN449 shared folder.
9. Create a kernel image by running a cross compiler Linux configuration for the ARM processor.
10. Convert zImage into uImage.
11. Modify the zynq-zybo.dts by adding ‘multiply’ IP code.
12. Convert the .dts file to .dtb file.
13. Make the ramdisk by copying the ramdisk file from ECEN449 shared folder.
14. Open MobaXterm, copy BOOT.bin, uImage, uramdisk.image.gz and devicetree.dtb on to SD card then plug it into the Zybo board.
15. Turn on the Zybo board to boot Linux and read the data from COM4 in MobaXterm.

Results:

The FPGA booted Linux.



```
devtmpfs: mounted
Freeing unused kernel memory: 212K (40627000 - 4065c000)
Starting rcS...
++ Mounting filesystem
++ Setting up mdev
++ Starting telnet daemon
++ Starting http daemon
++ Starting ftp daemon
++ Starting dropbear (ssh) daemon
random: dropbear urandom read with 1 bits of entropy available
rcS Complete
zynq> ls
bin         lib         lost+found  proc        sys         var
dev         licenses    mnt         root        tmp
etc         linuxrc     opt         sbin        usr
zynq> cd bin
```

obaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

Conclusion:

From this lab, I learned how to boot up Linux on the FGPA by creating a device tree block, a RAMDISK temporary file system, and a boot bin file. The lab took a fair amount of time from having to untar certain files and making sure that I was in the right directory.

Post-Lab Questions :

1. **Compared to lab 3, the lab 4 microprocessor system shown in Figure 1 has 512 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 DDR3 provides 512 megabytes for the Linux kernel. But the local memory that's included in our system is for data to be stored so that future tasks are faster. Yes, local memory does exist on the standard motherboard. The memory that is here is located in the registers on the Zynq ARM chip and helps reduce the cost of data accessing from the RAM.

2. **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 ZYBO Z7-10 board? Why?**

/proc and /sys is not writeable. All other directories are writeable. If we try to write either of these two, we will lose the changes because it's not written on the SD Card. (Only recorded in RAM)

3. **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?**

The steps I would repeat are the ones where we started configuring and creating the .dtb and kernel. This is because we have to configure the OS to know the address of the hardware we are using.