Lab Report 4
Linux Boot-Up on ZYBO Board via SD Card

Christopher Padilla

ECEN 449 - 504

Purpose/Introduction

- This lab is intended to guide students through the process of getting Linux up and running on the FPGA. We will use Vivado to build a Zynq processor system suitable for running Linux; this will all be done using the hard processor (ARM Processor in Zynq Chip on Zybo Board) by using open-source tools to compile Linux based on our specifications.

Procedure

- We begin by creating a block design and adding ZYNQ7 PS IP and add the necessary peripherals just like in the previous lab and create an HDL wrapper to the base system, with a few minor differences in peripherals.
- When compiling U-Boot we need the cross-compile tools that Vivado provides, so we source Vivado's settings and configure the universal boot loader to our target device (the ZYBO board). Our u-boot file successfully generates for the Xilinx SDK to read. This allows us to create BOOT.bin by using a First-Stage Boot Loader application project.
- The last step was to work with the Linux Kernel source code. We configure it for our ZYBO board and compile it. We generate a Linux kernel image (uImage) and we edit the .dtb to include 'multiply' IP and combine our four files and put them in the SD card to boot on ZYBO.

Results

- After plugging the SD card into the ZYBO board and pressing the reset button, Linux starts booting up on the board. The PICOCOM console shows this is in fact the case, meaning we have successfully booted Linux on our FPGA. A screenshot of the PICOCOM are included as part of this report.

Conclusion

- In this lab, I was able to get Linux up and running on the ZYBO board by creating a device tree block, a boot bin file, and a RAMDISK temporary file system. This lab took several hours to complete, but once I got all the parts together, it worked like a charm. Overall, there are many advantages to running an OS on an embedded processor, and it is clear that using Linux's open-source software can be useful for accomplishing a multitude of tasks.

Questions

- The DDR3 SDRAM controller provides 512 megabytes for the Linux kernel to reside in. However, the local memory that our system includes is so data can be stored so future requests for that data can be served faster (CPU cache). Yes, local memory does exist on the motherboard; this memory is located in registers on our Zynq ARM chip and reduces the cost of accessing data from RAM.
- /proc is not writable and /sys is not writable; all other directories are writeable. If we try to write to one of these, we lose the changes because it is not written to our bootup settings (on the SD card); it is only recorded in RAM.
- For this lab we started configuring and creating the .dtb and the kernel for our current peripherals, and those are the steps we would have to repeat. This has to do with the fact that we have to configure the OS to know the addresses of the hardware we're using.