

ECEN 749

Laboratory Exercise #4

Linux boot-up on ZYBO Z7-10 board via SD Card

Name: Pranav Anantharam

UIN: 734003886

Course: ECEN 749 Microprocessor System Design

Section: 603

Introduction:

In the fourth laboratory exercise, we focused on initializing Linux on the ZYBO Z7-10 board. This exercise was essential for understanding how to implement an Operating System (OS) within an embedded processor environment, with Linux serving as an open-source platform for further development. Throughout this exercise, we utilized Vivado to construct a Zynq-based microprocessor system optimized for Linux operation. Additionally, we employed PetaLinux tools to compile the Linux kernel to suit the specifications of our custom microprocessor system. By integrating components such as FSBL (First Stage Boot Loader) and u-boot (Universal Boot Loader), we created a Zynq Boot Image essential for booting Linux on the ZYBO Z7-10 board via an SD card. This exercise provided us with practical insights into configuring embedded systems with Linux, a skillset highly relevant across engineering and computing disciplines.

Procedure:

- 1) Create a working directory in the home folder.
- 2) Load the environment variables on the workstation using appropriate commands and open Vivado 2022.1.
- 3) Create a new project. Select 'Boards' and choose the 'Zybo Z7-10' board option.
- 4) Create a new block design and add the 'ZYNQ7 Processing System' IP to our design.
- 5) Run block automation and import 'ZYBO_Z7_B2.tcl'.
- 6) Open the 'Re-customize IP' window for the PS IP.
- 7) Enable SD 0, UART 1 and TTC 0 peripherals.
- 8) Copy the 'ip_repo' folder from Lab 3 into Lab 4 directory.
- 9) Add the multiply IP to the current project IP catalog. Next, add the multiply IP to the base system.
- 10) Run connection automation.
- 11) Create HDL wrapper for the base system.
- 12) Generate bitstream and export the hardware design.
- 13) Insert USB drive and copy PetaLinux installer from the shared path into the USB drive.
- 14) Install PetaLinux in a directory within the USB drive.
- 15) Source PetaLinux environment settings.
- 16) Create PetaLinux project with a name using the zynq template.
- 17) Move the PetaLinux project directory and reconfigure PetaLinux project with the exported hardware.
- 18) Build the PetaLinux project.
- 19) Generate the zynq boot image using commands given in the lab manual.
- 20) Copy BOOT.bin, image.ub and boot.scr files to the SD card.
- 21) Insert SD card into the Zybo Z7-10 FPGA board.
- 22) Change JP5 into SD card mode to boot from the SD card and power on the ZYBO Z7-10 board.
- 23) Use picocom serial console application to view the output of printf statements on the machine using the commands given in the manual.

(Output Screenshots given in Appendix Section)

Results:

We created a Zynq processing system with the required components (Zynq processor, UART peripheral and multiplier IP). Additionally, peripherals such as SD card, timer and DDR3 controller were also integrated. The multiplier IP was integrated into our base system design (Zynq processing system) using Vivado.

Using PetaLinux toolchain, an optimized version of Linux kernel for the Zybo board was built and the required files (FSBL, u-boot) for booting Linux kernel on the board were prepared. A zynq boot image was prepared and loaded onto the SD card. Using the SD card, Linux kernel was booted on the FPGA board.

Outputs were obtained and displayed on a serial console using picocom application. The outputs were demonstrated to the TA as well.

(Output Screenshots given in Appendix Section)

Conclusion:

In conclusion, the fourth laboratory exercise provided practical insights into initializing Linux on the ZYBO Z7-10 board, essential for understanding embedded OS implementation. Using Vivado and PetaLinux tools, we constructed a Zynq-based microprocessor system optimized for Linux operation, crucial skills applicable across engineering and computing disciplines. Integrating components like FSBL and u-boot, we created a Zynq Boot Image for booting Linux on the ZYBO Z7-10 board via an SD card. This hands-on experience highlights the importance of configuring embedded systems with Linux, offering fundamental skills for future development.

Questions:

- (a) 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?

Answer:

The 512MB of SDRAM is utilized by the Linux operating system. The local memory present on our system is the CPU cache. This facilitates easy data access and storage of temporary files. This local memory exists on all standard motherboards, such as the Zynq ARM chip in our case, and resides in the registers.

- (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 can create a file in one of these directories. What happens to this file when you restart the ZYBO Z7-10 board? Why?

Answer:

In our Linux system, only the "proc" and "sys" directories have read-only permissions. All other directories are writable. If we try to create a file using the "touch" command in either of these two directories, the file is not saved permanently, and we lose it when we restart the Zybo board. This happens because any files created in these directories are only stored in RAM and not written to the SD card due to the lack of permissions. Hence, upon board restart, these files are deleted, requiring us to recreate them if we want to reuse them.

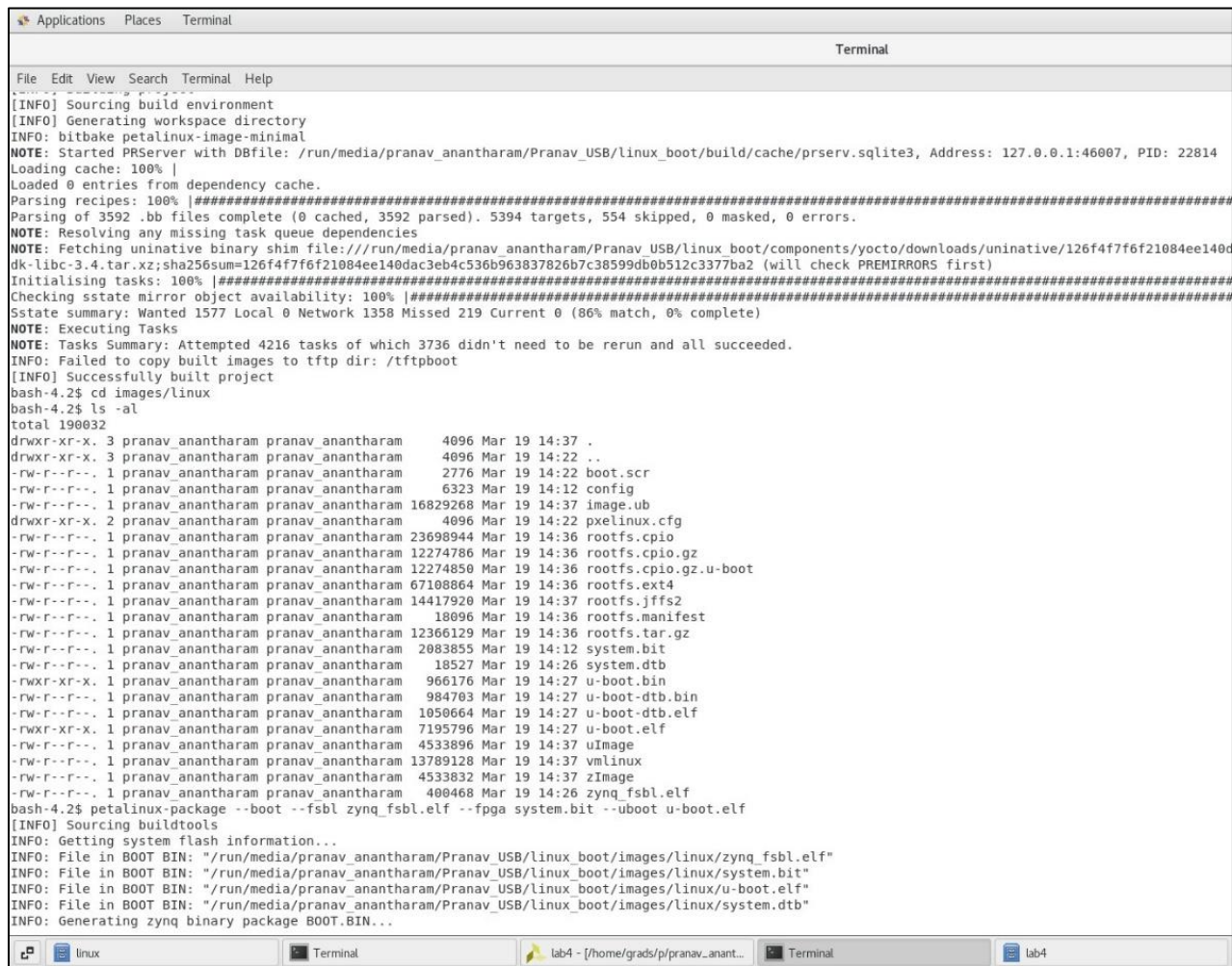
(c) If you were to add another peripheral to your system after compiling the kernel, which of the above steps would you have to repeat? Why?

Answer:

If we were to add another peripheral to our system after compiling the kernel using PetaLinux tools, we would need to repeat the step of configuring and compiling the kernel. This is because adding a new peripheral requires modifying the device tree configuration to include the necessary drivers and settings for the new hardware component. First, we must update the Zynq base system and integrate the new peripheral into the block design, generate bitstream and export the hardware design. Next, we need to use PetaLinux toolchain and reconfigure the PetaLinux project with the exported hardware design. Then, we need to generate the zynq boot image (BOOT.bin) and copy the BOOT.bin, image.ub and boot.scr files into the SD card. Finally, we can boot the new Linux kernel on the Zybo Z7 board via SD card and view the output on the serial console using picocom.

Appendix:

PetaLinux Toolchain build process Console Output Screenshots:



```
Applications  Places  Terminal

Terminal

File Edit View Search Terminal Help

[INFO] Sourcing build environment
[INFO] Generating workspace directory
INFO: bitbake petalinux-image-minimal
NOTE: Started PRServer with DBfile: /run/media/pranav_anantharam/Pranav_USB/linux_boot/build/cache/prserv.sqlite3, Address: 127.0.0.1:46007, PID: 22814
Loading cache: 100% |
Loaded 0 entries from dependency cache.
Parsing recipes: 100% |#####
Parsing of 3592 .bb files complete (0 cached, 3592 parsed). 5394 targets, 554 skipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies
NOTE: Fetching uninitative binary shim file:///run/media/pranav_anantharam/Pranav_USB/linux_boot/components/yocto/downloads/uninitative/126f4f7f6f21084ee140d
dk-libc-3.4.tar.xz;sha256sum=126f4f7f6f21084ee140d3eb4c536b963837826b7c38599db0b512c3377ba2 (will check PREMIRRORS first)
Initialising tasks: 100% |#####
Checking sstate mirror object availability: 100% |#####
Sstate summary: Wanted 1577 Local 0 Network 1358 Missed 219 Current 0 (86% match, 0% complete)
NOTE: Executing Tasks
NOTE: Tasks Summary: Attempted 4216 tasks of which 3736 didn't need to be rerun and all succeeded.
INFO: Failed to copy built images to tftp dir: /tftpboot
[INFO] Successfully built project
bash-4.2$ cd images/linux
bash-4.2$ ls -al
total 190832
drwxr-xr-x. 3 pranav_anantharam pranav_anantharam 4096 Mar 19 14:37 .
drwxr-xr-x. 3 pranav_anantharam pranav_anantharam 4096 Mar 19 14:22 ..
-rw-r--r--. 1 pranav_anantharam pranav_anantharam 2776 Mar 19 14:22 boot.scr
-rw-r--r--. 1 pranav_anantharam pranav_anantharam 6323 Mar 19 14:12 config
-rw-r--r--. 1 pranav_anantharam pranav_anantharam 16829268 Mar 19 14:37 image.ub
drwxr-xr-x. 2 pranav_anantharam pranav_anantharam 4096 Mar 19 14:22 pxelinux.cfg
-rw-r--r--. 1 pranav_anantharam pranav_anantharam 23698944 Mar 19 14:36 rootfs.cpio
-rw-r--r--. 1 pranav_anantharam pranav_anantharam 12274786 Mar 19 14:36 rootfs.cpio.gz
-rw-r--r--. 1 pranav_anantharam pranav_anantharam 12274850 Mar 19 14:36 rootfs.cpio.gz.u-boot
-rw-r--r--. 1 pranav_anantharam pranav_anantharam 67108864 Mar 19 14:36 rootfs.ext4
-rw-r--r--. 1 pranav_anantharam pranav_anantharam 14417920 Mar 19 14:37 rootfs.jffs2
-rw-r--r--. 1 pranav_anantharam pranav_anantharam 18096 Mar 19 14:36 rootfs.manifest
-rw-r--r--. 1 pranav_anantharam pranav_anantharam 12366129 Mar 19 14:36 rootfs.tar.gz
-rw-r--r--. 1 pranav_anantharam pranav_anantharam 2083855 Mar 19 14:12 system.bit
-rw-r--r--. 1 pranav_anantharam pranav_anantharam 18527 Mar 19 14:26 system.dtb
-rwxr-xr-x. 1 pranav_anantharam pranav_anantharam 966176 Mar 19 14:27 u-boot.bin
-rw-r--r--. 1 pranav_anantharam pranav_anantharam 984703 Mar 19 14:27 u-boot-dtb.bin
-rw-r--r--. 1 pranav_anantharam pranav_anantharam 1050664 Mar 19 14:27 u-boot-dtb.elf
-rwxr-xr-x. 1 pranav_anantharam pranav_anantharam 7195796 Mar 19 14:27 u-boot.elf
-rw-r--r--. 1 pranav_anantharam pranav_anantharam 4533896 Mar 19 14:37 uImage
-rw-r--r--. 1 pranav_anantharam pranav_anantharam 13789128 Mar 19 14:37 vmlinux
-rw-r--r--. 1 pranav_anantharam pranav_anantharam 4533832 Mar 19 14:37 zImage
-rw-r--r--. 1 pranav_anantharam pranav_anantharam 400468 Mar 19 14:26 zynq_fsbl.elf
bash-4.2$ petalinux-package --boot --fsbl zynq_fsbl.elf --fpga system.bit --u-boot u-boot.elf
[INFO] Sourcing buildtools
INFO: Getting system flash information...
INFO: File in BOOT BIN: "/run/media/pranav_anantharam/Pranav_USB/linux_boot/images/linux/zynq_fsbl.elf"
INFO: File in BOOT BIN: "/run/media/pranav_anantharam/Pranav_USB/linux_boot/images/linux/system.bit"
INFO: File in BOOT BIN: "/run/media/pranav_anantharam/Pranav_USB/linux_boot/images/linux/u-boot.elf"
INFO: File in BOOT BIN: "/run/media/pranav_anantharam/Pranav_USB/linux_boot/images/linux/system.dtb"
INFO: Generating zynq binary package BOOT.BIN...
```

Picocom Serial Console Output Screenshots:

```
Applications  Places  Terminal

Terminal

File Edit View Search Terminal Help
mmcblk0: mmc0:aaaa S508G 7.40 GiB
mmcblk0: p1
Freeing initrd memory: 11988K
Freeing unused kernel image (initmem) memory: 1024K
Run /init as init process
INIT: version 2.99 booting
Starting udev
Starting version 249.7+
random: fast init done
FAT-fs (mmcblk0p1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
bootlogd: /dev/ttyPS0hwclock: can't open '/dev/misc/rtc': No such file or directory
Fri Mar 9 12:34:56 UTC 2018
hwclock: can't open '/dev/misc/rtc': No such file or directory
random: dd: uninitialized urandom read (512 bytes read)
Configuring packages on first boot...
(This may take several minutes. Please do not power off the machine.)
Running postinst /etc/rpm-postinsts/100-sysvinit-inittab...
update-rc.d: /etc/init.d/run-postinsts exists during rc.d purge (continuing)
Removing any system startup links for run-postinsts ...
/etc/rcS.d/S99run-postinsts
INIT: Entering runlevel: 5
Configuring network interfaces... Cannot find device "eth0"
Starting haveged: haveged: command socket is listening at fd 3
haveged: haveged starting up
[ OK ]
Starting Dropbear SSH server: Waiting for kernel randomness to be initialised...
haveged: haveged: ver: 1.9.14; arch: generic; vend: ; build: (gcc 11.2.0 CTV); collect: 128K

haveged: haveged: cpu: (VC); data: 16K (D); inst: 16K (D); idx: 12/40; sz: 15126/57786

haveged: haveged: tot tests(BA8): A:1/1 B:1/1 continuous tests(B): last entropy estimate 8.00324

haveged: haveged: fills: 0, generated: 0

random: crng init done
Generating 2048 bit rsa key, this may take a while...
Public key portion is:
ssh-rsa AAAAB3NzaClyc2EAAAADAQABAAQCDBmu+RH9Z+w5hJpRpglwZ6Dc4W0cmTELrtwx0wzSPkAY6I2mWxug/y12G9sHinyPTGbJqP+VTucaK4ZR+wxyoRzs
SYFYRwRrXkiQPopKsU0cMf6HD6/Q7l0K9dxKaMdGoyCtoeH4k4RHDIEk2RoUh1pARgxMZD8WTEhjvUmUHlbsnevu5+Tazj9ZLvFiutrGoBw8NgY7I9gRGfrgv12yzf1l
Fingerprint: sha1!! 5e:cc:e9:14:07:51:12:bd:e3:51:23:7c:ef:b7:d6:a1:94:50:83:12
dropbear.
Starting rpcbind daemon...done.
starting statd: done
hwclock: can't open '/dev/misc/rtc': No such file or directory
Starting internet superserver: inetd.
NFS daemon support not enabled in kernel
Starting syslogd/klogd: done
Starting tcf-agent: OK

PetaLinux 2022.1_release_S04190222 linux_boot /dev/ttyPS0

linux_boot login: █
```