ECEN 749 Lab 4 Report

Tong Lu
UIN 621007982

October 5, 2018

TEXAS A&M
UNIVERSITY®

Introduction

In this lab, we configured a ZYNQ ARM processor similar to lab 3, and then compile an linux kernel to run on it.

Procedure

1. Launch Vivado and create a block design with ZYNQ processor.
2. Re-customize the IP design and enable UART1 on peripheral pins for USB communication with PC.
3. Copy ip_repo from lab 3 into lab 4 directory.
4. Create HDL wrapper and generate bitstream for the block design.
5. Extract the u-boot .tar.gz

```
tar -xvzf /homes/faculty/shared/ECEN449/u-boot.tar.gz
```

6. Configure and build U-boot

```
make CROSS_COMPILE=arm-xilinx-linux-gnueabi- zynq_zybo_config
```

7. Cross-compile u-boot

```
make CROSS_COMPILE=arm-xilinx-linux-gnueabi-
```

8. Launch SDK and create a First Stage Boot Loader (FSBL).
9. Create boot image based on FSBL (as bootloader), Bitstream (as datafile), and u-boot file (as datafile).
10. Untar linux-3.14.tar.gz and configure.

```
make ARCH=arm CROSS_COMPILE=arm-xilinx-linux-gnueabi- xilinx_zynq_defconfig
```

11. Cross-compile the kernel

```
make ARCH=arm CROSS_COMPILE=arm-xilinx-linux-gnueabi-
```

12. Set up path variable

```
PATH=$PATH:<directory_to_u_boot>/tools
```

13. Convert the compiled zImage into uImage.

```
  make ARCH=arm CROSS_COMPILE=arm-xilinx-linux-gnueabi- UIMAGE_LOADADDR=0x8000
    uImage
```

14. Add multipy IP in devicetree file and convert it into .dtb format.

```
./scripts/dtc/dtc -I dts -O dtb -o ./devicetree.dtb arch/arm/boot/dts/zynq-zybo.dts
```

15. Copy ramdisk file from ECEN 449 directory and create the uramdisk file base on it.

```
./u-boot/tools/mkimage -A arm -T ramdisk -c gzip -d ./ramdisk8M.image.gz uramdisk.image.gz
```

16. Copy BOOT.BIN, uImage, devicetree.dtb, and uramdisk.image.gz into an SD card.
17. Insert SD card into FPGA, push Reset button, and then use picocom to monitor USB1 port and check the printout.

Result

All the programs was finished and demonstrated to TA. The programs are working well and meet all the requirement on lab manual

```
[lvtongtom305@lin05-424cvlb ~]$ picocom -b 115200 -r -l /dev/ttyUSB1
picocom v2.3a

port is        : /dev/ttyUSB1
flowcontrol    : none
baudrate is    : 115200
parity is      : none
databits are   : 8
stopbits are   : 1
escape is      : C-a
local echo is  : no
noinit is      : no
noreset is     : yes
nolock is      : yes
send_cmd is    : sz -vv
receive_cmd is : rz -vv -E
imap is        :
omap is        :
emap is        : crcrlf,delbs,
logfile is     : none

Type [C-a] [C-h] to see available commands

Terminal ready
 0
Device: zynq_sdhci
Manufacturer ID: 3
OEM: 5344
Name: SS08G
Tran Speed: 50000000
Rd Block Len: 512
SD version 3.0
High Capacity: Yes
Capacity: 7.4 GiB
Bus Width: 4-bit
```

```
reading uEnv.txt
** Unable to read file uEnv.txt **
Copying Linux from SD to RAM...
reading uImage
3447904 bytes read in 303 ms (10.9 MiB/s)
reading devicetree.dtb
7446 bytes read in 16 ms (454.1 KiB/s)
reading uramdisk.image.gz
3693174 bytes read in 326 ms (10.8 MiB/s)
## Booting kernel from Legacy Image at 03000000 ...
   Image Name:   Linux-3.18.0-xilinx
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    3447840 Bytes = 3.3 MiB
   Load Address: 00008000
   Entry Point:  00008000
   Verifying Checksum ... OK
## Loading init Ramdisk from Legacy Image at 02000000 ...
   Image Name:
   Image Type:   ARM Linux RAMDisk Image (gzip compressed)
   Data Size:    3693110 Bytes = 3.5 MiB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 02a00000
   Booting using the fdt blob at 0x2a00000
   Loading Kernel Image ... OK
   Loading Ramdisk to 1f7aa000, end 1fb2fa36 ... OK
   Loading Device Tree to 1f7a5000, end 1f7a9d15 ... OK

Starting kernel ...

Booting Linux on physical CPU 0x0
Linux version 3.18.0-xilinx (lvtongtom305@lin05-424cvlb.ece.tamu.edu) (gcc version 4.9.1 (Sourcery CodeBench Lite 2014.11-30) ) #1 SMP PREEMPT Fri Sep 28 21:04:24 CDT 2018
CPU: ARMv7 Processor [413fc090] revision 0 (ARMv7), cr=18c5387d
CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
Machine model: Xilinx Zynq
cma: Reserved 16 MiB at 0x1e400000
Memory policy: Data cache writealloc
PERCPU: Embedded 10 pages/cpu @5fbd3000 s8768 r8192 d24000 u40960
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 130048
Kernel command line: console=ttyPS0,115200 root=/dev/ram rw earlyprintk
PID hash table entries: 2048 (order: 1, 8192 bytes)
Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
Memory: 492632K/524288K available (4650K kernel code, 258K rwdata, 1616K rodata, 212K init, 219K bss, 31656K reserved, 0K highmem)
Virtual kernel memory layout:
    vector  : 0xffff0000 - 0xffff1000   (   4 kB)
    fixmap  : 0xffc00000 - 0xffe00000   (2048 kB)
    vmalloc : 0x60800000 - 0xff000000   (2536 MB)
    lowmem  : 0x40000000 - 0x60000000   ( 512 MB)
    pkmap   : 0x3fe00000 - 0x40000000   (   2 MB)
    modules : 0x3f000000 - 0x3fe00000   (  14 MB)
      .text : 0x40008000 - 0x40626b1c   (6267 kB)
      .init : 0x40627000 - 0x4065c000   ( 212 kB)
      .data : 0x4065c000 - 0x4069cb60   ( 259 kB)
       .bss : 0x4069cb60 - 0x406d3a78   ( 220 kB)
```

```
Preemptible hierarchical RCU implementation.
    Dump stacks of tasks blocking RCU-preempt GP.
    RCU restricting CPUs from NR_CPUS=4 to nr_cpu_ids=2.
RCU: Adjusting geometry for rcu_fanout_leaf=16, nr_cpu_ids=2
NR_IRQS:16 nr_irqs:16 16
L2C-310 erratum 769419 enabled
L2C-310 enabling early BRESP for Cortex-A9
L2C-310 full line of zeros enabled for Cortex-A9
L2C-310 ID prefetch enabled, offset 1 lines
L2C-310 dynamic clock gating enabled, standby mode enabled
L2C-310 cache controller enabled, 8 ways, 512 kB
L2C-310: CACHE_ID 0x410000c8, AUX_CTRL 0x76360001
ps7-slcr mapped to 60804000
zynq_clock_init: clkc starts at 60804100
Zynq clock init
sched_clock: 64 bits at 325MHz, resolution 3ns, wraps every 3383112499200ns
ps7-ttc #0 at 60806000, irq=43
Console: colour dummy device 80x30
Calibrating delay loop... 1292.69 BogoMIPS (lpj=6463488)
pid_max: default: 32768 minimum: 301
Mount-cache hash table entries: 1024 (order: 0, 4096 bytes)
Mountpoint-cache hash table entries: 1024 (order: 0, 4096 bytes)
CPU: Testing write buffer coherency: ok
CPU0: thread -1, cpu 0, socket 0, mpidr 80000000
Setting up static identity map for 0x467598 - 0x4675f0
CPU1: Booted secondary processor
CPU1: thread -1, cpu 1, socket 0, mpidr 80000001
Brought up 2 CPUs
SMP: Total of 2 processors activated.
CPU: All CPU(s) started in SVC mode.
devtmpfs: initialized
VFP support v0.3: implementor 41 architecture 3 part 30 variant 9 rev 4
regulator-dummy: no parameters
NET: Registered protocol family 16
DMA: preallocated 256 KiB pool for atomic coherent allocations
cpuidle: using governor ladder
cpuidle: using governor menu
hw-breakpoint: found 5 (+1 reserved) breakpoint and 1 watchpoint registers.
hw-breakpoint: maximum watchpoint size is 4 bytes.
zynq-ocm f800c000.ps7-ocmc: ZYNQ OCM pool: 256 KiB @ 0x60880000
vgaarb: loaded
SCSI subsystem initialized
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
media: Linux media interface: v0.10
Linux video capture interface: v2.00
pps_core: LinuxPPS API ver. 1 registered
pps_core: Software ver. 5.3.6 - Copyright 2005-2007 Rodolfo Giometti <giometti@linux.it>
PTP clock support registered
EDAC MC: Ver: 3.0.0
Advanced Linux Sound Architecture Driver Initialized.
Switched to clocksource arm_global_timer
NET: Registered protocol family 2
TCP established hash table entries: 4096 (order: 2, 16384 bytes)
TCP bind hash table entries: 4096 (order: 3, 32768 bytes)
TCP: Hash tables configured (established 4096 bind 4096)
TCP: reno registered
```

```
UDP hash table entries: 256 (order: 1, 8192 bytes)
UDP-Lite hash table entries: 256 (order: 1, 8192 bytes)
NET: Registered protocol family 1
RPC: Registered named UNIX socket transport module.
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
RPC: Registered tcp NFSv4.1 backchannel transport module.
Trying to unpack rootfs image as initramfs...
rootfs image is not initramfs (no cpio magic); looks like an initrd
Freeing initrd memory: 3608K (5f7aa000 - 5fb30000)
hw perfevents: enabled with armv7_cortex_a9 PMU driver, 7 counters available
futex hash table entries: 512 (order: 3, 32768 bytes)
jffs2: version 2.2. (NAND) (SUMMARY)  © 2001-2006 Red Hat, Inc.
msgmni has been set to 1001
io scheduler noop registered
io scheduler deadline registered
io scheduler cfq registered (default)
dma-pl330 f8003000.ps7-dma: Loaded driver for PL330 DMAC-241330
dma-pl330 f8003000.ps7-dma:     DBUFF-128x8bytes Num_Chans-8 Num_Peri-4 Num_Events-16
xuartps e0001000.serial: ttyPS0 at MMIO 0xe0001000 (irq = 82, base_baud = 3125000) is a xuartps
console [ttyPS0] enabled
xdevcfg f8007000.ps7-dev-cfg: ioremap 0xf8007000 to 6086c000
[drm] Initialized drm 1.1.0 20060810
brd: module loaded
loop: module loaded
CAN device driver interface
e1000e: Intel(R) PRO/1000 Network Driver - 2.3.2-k
e1000e: Copyright(c) 1999 - 2014 Intel Corporation.
libphy: XEMACPS mii bus: probed
xemacps e000b000.ps7-ethernet: invalid address, use random
xemacps e000b000.ps7-ethernet: MAC updated 02:74:6d:7c:40:c5
xemacps e000b000.ps7-ethernet: pdev->id -1, baseaddr 0xe000b000, irq 54
ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
ehci-pci: EHCI PCI platform driver
zynq-dr e0002000.ps7-usb: Unable to init USB phy, missing?
usbcore: registered new interface driver usb-storage
mousedev: PS/2 mouse device common for all mice
i2c /dev entries driver
Xilinx Zynq CpuIdle Driver started
sdhci: Secure Digital Host Controller Interface driver
sdhci: Copyright(c) Pierre Ossman
sdhci-pltfm: SDHCI platform and OF driver helper
sdhci-arasan e0100000.ps7-sdio: No vmmc regulator found
sdhci-arasan e0100000.ps7-sdio: No vqmmc regulator found
mmc0: SDHCI controller on e0100000.ps7-sdio [e0100000.ps7-sdio] using ADMA
ledtrig-cpu: registered to indicate activity on CPUs
usbcore: registered new interface driver usbhid
usbhid: USB HID core driver
TCP: cubic registered
NET: Registered protocol family 17
can: controller area network core (rev 20120528 abi 9)
NET: Registered protocol family 29
can: raw protocol (rev 20120528)
can: broadcast manager protocol (rev 20120528 t)
can: netlink gateway (rev 20130117) max_hops=1
zynq_pm_ioremap: no compatible node found for 'xlnx,zynq-ddrc-a05'
zynq_pm_late_init: Unable to map DDRC IO memory.
Registering SWP/SWPB emulation handler
```

```
drivers/rtc/hctosys.c: unable to open rtc device (rtc0)
ALSA device list:
  No soundcards found.
RAMDISK: gzip image found at block 0
mmc0: new high speed SDHC card at address aaaa
mmcblk0: mmc0:aaaa SS08G 7.40 GiB
 mmcblk0: p1
EXT2-fs (ram0): warning: mounting unchecked fs, running e2fsck is recommended
VFS: Mounted root (ext2 filesystem) on device 1:0.
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 /dev
console             ram7                tty39
cpu_dma_latency     ram8                tty4
full                ram9                tty40
i2c                 random              tty41
iio:device0         root                tty42
input               snd                 tty43
kmsg                timer               tty44
loop-control        tty                 tty45
loop0               tty0                tty46
loop1               tty1                tty47
loop2               tty10               tty48
loop3               tty11               tty49
loop4               tty12               tty5
loop5               tty13               tty50
loop6               tty14               tty51
loop7               tty15               tty52
mem                 tty16               tty53
memory_bandwidth    tty17               tty54
mice                tty18               tty55
mmcblk0             tty19               tty56
mmcblk0p1           tty2                tty57
network_latency     tty20               tty58
network_throughput  tty21               tty59
null                tty22               tty6
port                tty23               tty60
psaux               tty24               tty61
ptmx                tty25               tty62
pts                 tty26               tty63
ram0                tty27               tty7
ram1                tty28               tty8
ram10               tty29               tty9
ram11               tty3                ttyPS0
ram12               tty30               urandom
ram13               tty31               vcs
ram14               tty32               vcs1
ram15               tty33               vcsa
ram2                tty34               vcsa1
ram3                tty35               vga_arbiter
```

```
ram4                tty36               xdevcfg
ram5                tty37               zero
ram6                tty38
```

src/console_printout

.

Conclusion

In this lab, I learned to cross-compile an linux kernel for ARM processor using make and vivado, and successfully boot it up on the FPGA. This lab will be very helpful for the future projects.

Answer to 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?**

Local memoey works like cache for the processor. This ″local memory″ does exist on a standard motherboard, and it is normally integrated with CPU these days.

(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 ZYBO board? Why?

/proc and /sys are not writable. Everything else are writable.
We will lose the file and data we created when we restart the ZYBO board.

(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?**

We need to repeat the steps of changing the devicetree file. In this way, the system will understand the hardware and address we are using.