Embedded Linux on Zynq using Vivado Lab2

Innova Lee(이상훈)
gcccompil3r@gmail.com

Introduction

임베디드 리눅스 개발에 필요한 가장 기본적인 기술은 크로스 컴파일 환경에서 작동한다:
커널, 라이브러리 및 응용 프로그램을 컴파일하고 결과 이미지를 Target 으로 다운로드 한다.
이 Lab 의 목적은 이 과정을 익히는 것이다.

이 Lab 에서는 임베디드 리눅스 작업의 가장 기본적인 작업인 운영체제 및
응용 프로그램을 빌드하고 부팅하는 방법에 대해 준비할 것이다.
ARM Cortex-A9 MPcore 와 같은 임베디드 리눅스 타겟 프로세서는 일반적으로 크로스 컴파일 환경에서 개발된다.
즉 커널과 응용 프로그램은 개발 컴퓨터(이 경우 Target 이 아닌 프로세서가 있는 Linux PC)에서
컴파일 된 다음 Target 에 다운로드 된다.

PetaLinux 도구는 이 프로세스의 대부분을 자동화하는 여러 가지 구성 아키텍처를 지원한다.
이 Lab 에서는 이러한 도구를 사용하는 방법과 결과로 생성되는
임베디드 Linux 이미지를 하드웨어 플랫폼에 다운로드하는 방법을 배우게 된다.

QEMU 는 PetaLinux 툴에 통합 된 포괄적이고 오픈 소스인 System Emulator 이다.
이 Lab 에서는 QEMU 를 사용하여 ARM Cortex-A9 MPcore System 전용으로 빌드 된 Linux 를 실행한다.
Host CPU 에서 Guest Code 를 직접 실행하여 원 성능에 근접한 결과를 얻을 수 있다.

Objectives

이 Lab 을 완료하면 아래를 수행 할 수 있다:

- ARM Cortex-A9 Mpcore Linux 커널 및 응용 프로그램 빌드
- QEMU 에서 결과 시스템 이미지 부팅
- 결과 시스템 이미지를 개발 보드에 다운로드 한다.

Preparation

이것이 첫 번째 Lab 인 경우 환경 설정 방법에 대한 필수 준비 정보는 Lab 1 의 "Before You Start" 섹션을 참조하라.

워크 스테이션이 다시 시작되거나 로그 아웃되면 아래 명령을 실행하여 Host 에서 DHCP 서버를 시작한다:

[host]$ sudo service isc-dhcp-server restart

자세한 내용은 Lab 1 의 "Initializing the Workshop Environment" 섹션을 참조하라.

General Flow for this Lab

Choosing a Linux Platform

Linux 플랫폼은 Linux 이미지로 무엇을 구축해야 하는지 알려준다;
다음 정보를 알려준다:

* 주소 맵핑, 인터럽트 및 프로세서 특성과 같은 하드웨어 플랫폼 정보이며 예는 아래와 같다.
* 리눅스 커널 설정
* 사용자 공간 응용 프로그램 설정
* 파일 시스템 설정
* 플래시 파티션 테이블 설정

1-1. 경로를 프로젝트 디렉토리로 변경한다.

1-1-1. 아래 명령을 실행하여 프로젝트 디렉토리 경로를 만들고 변경한다:

```
[host]$ mkdir ~/emblnx/labs/lab2
[host]$ cd ~/emblnx/labs/lab2
```

이 워크숍의 각 Lab 은 ~/emblnx/labs 디렉토리에 설치된다.
Lab 을 다른 경로에 설치 한 경우 경로를 조정하라.

1-2. petalinux-create 명령을 사용하여 새로운 임베디드 리눅스 플랫폼을 만들고 플랫폼을 선택한다.

1-2-1. lab2 디렉토리에서 아래 명령을 실행하여 새로운 PetaLinux 프로젝트를 작성한다.

```
[host]$ petalinux-create -t project -s /opt/pkg/ZYBO_petalinux_v2015_4.bsp
```

```
petalinux@ubuntu:~/emblnx/labs/lab2$ petalinux-create -t project -s /opt/pkg/ZYBO_petalinux_v2015_4.bsp
INFO: Create project:
INFO: Projects:
INFO:   * ZYBO_petalinux_v2015_4
INFO: has been successfully installed to /home/petalinux/emblnx/labs/lab2/
INFO: New project successfully created in /home/petalinux/emblnx/labs/lab2/
```

Figure 1. Creating a new PetaLinux project

위의 명령은 Board Support Package(BSP) 가 /opt/pkg 디렉토리에 설치되어 있다고 가정한다.
BSP 가 다른 위치에 있으면 경로를 수정한다.

이 명령은 PetaLinux 소프트웨어 프로젝트 디렉토리를 생성한다:
~/emblnx/labs/lab2 밑에 ZYBO_petalinux_v2015_4 에 해당한다.

PetaLinux 프로젝트 디렉토리는 프로젝트, Linux 서브 시스템 및 서브 시스템의 구성 요소의 구성 파일을 포함한다.
petalinux-build 는 해당 설정 파일로 프로젝트를 빌드한다.
사용자는 petalinux-config 를 실행하여 수정할 수 있다.
아래는 PetaLinux 프로젝트 디렉토리다.

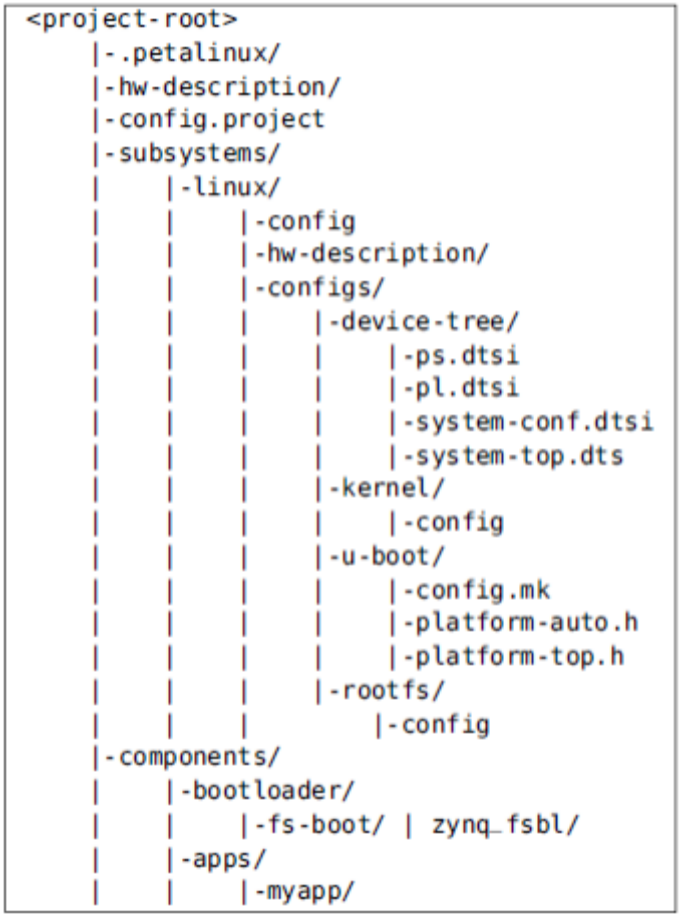

Figure 1: PetaLinux Project Directory

1-2-2. 디렉토리를 ~/emblnx/labs/lab2/ZYBO_petalinux_v2015_4 로 변경한다.

Building the Linux Image

2-1. 이제 미리 빌드 된 플랫폼을 선택 했으므로 이 플랫폼을 기반으로 Linux 이미지를 빌드한다.

2-1-1. Linux 이미지를 빌드하려면 아래 명령을 입력한다:

$petalinux-build

```
petalinux@ubuntu:~/emblnx/labs/lab2/ZYBO_petalinux_v2015_4$ petalinux-build
INFO: Checking component...
INFO: Generating make files and build linux
INFO: Generating make files for the subcomponents of linux
INFO: Building linux
[INFO ] pre-build linux/rootfs/fwupgrade
[INFO ] pre-build linux/rootfs/gpio-demo
[INFO ] pre-build linux/rootfs/peekpoke
[INFO ] build system.dtb
[INFO ] build linux/kernel
[INFO ] generate linux/u-boot configuration files
[INFO ] update linux/u-boot source
[INFO ] build linux/u-boot
[INFO ] build zynq_fsbl
[INFO ] Setting up stage config
[INFO ] Setting up rootfs config
[INFO ] Updating for cortexa9-vfp-neon
[INFO ] Updating package manager
[INFO ] Expanding stagefs
[INFO ] build linux/rootfs/fwupgrade
[INFO ] build linux/rootfs/gpio-demo
[INFO ] build linux/rootfs/peekpoke
[INFO ] build kernel in-tree modules
[INFO ] modules linux/kernel
[INFO ] post-build linux/rootfs/fwupgrade
[INFO ] post-build linux/rootfs/gpio-demo
[INFO ] post-build linux/rootfs/peekpoke
[INFO ] pre-install linux/rootfs/fwupgrade
[INFO ] pre-install linux/rootfs/gpio-demo
[INFO ] pre-install linux/rootfs/peekpoke
[INFO ] install system.dtb
[INFO ] install linux/kernel
[INFO ] generate linux/u-boot configuration files
[INFO ] update linux/u-boot source
[INFO ] build linux/u-boot
[INFO ] install linux/u-boot
[INFO ] Expanding rootfs
[INFO ] install sys_init
[INFO ] install linux/rootfs/fwupgrade
[INFO ] install linux/rootfs/gpio-demo
[INFO ] install linux/rootfs/peekpoke
[INFO ] install kernel in-tree modules
[INFO ] modules_install linux/kernel
[INFO ] post-install linux/rootfs/fwupgrade
[INFO ] post-install linux/rootfs/gpio-demo
[INFO ] post-install linux/rootfs/peekpoke
[INFO ] package rootfs.cpio to /home/petalinux/emblnx/labs/lab2/ZYBO_petalinux_v2015_4/images/linux
[INFO ] Update and install vmlinux image
[INFO ] vmlinux linux/kernel
[INFO ] install linux/kernel
[INFO ] package zImage
[INFO ] zImage linux/kernel
[INFO ] install linux/kernel
[INFO ] Package HDF bitstream
```

이 작업은 몇 분 정도 걸릴 수 있다.
이 시간 동안 아래의 작업이 진행된다.

- 리눅스 커널의 크로스 컴파일 및 링크
- 기본 사용자 라이브러리 및 응용 프로그램(lib 및 user)의 상호 컴파일 및 링크
- ARM Cortex-A9 프로세서 Linux 루트 파일 시스템의 로컬 복사본 만들기 (romfs)
- 커널과 루트 파일 시스템을 하나의 다운로드 가능한 바이너리 이미지 파일(images)로 조합
- images 에서 tftpboot 로의 image 파일 복사

빌드 로그는 ~/emblnx/labs/lab2/ZYBO_petalinux_v2015_4/build.log 에 저장된다.

2-1-2. 컴파일이 완료되면 프로젝트 디렉토리에서 아래 명령을 실행하여 images/linux 서브 디렉토리의 내용을 보라:

[host]$ cd images/linux
[host]$ ls -la

```
petalinux@ubuntu:~/emblnx/labs/lab2/ZYBO_petalinux_v2015_4/images/linux$ ls -la
total 58820
drwxrwxr-x 2 petalinux petalinux     4096 Jan 22 13:29 .
drwxrwxr-x 3 petalinux petalinux     4096 Jan 22 13:24 ..
-rwxrwxr-x 1 petalinux petalinux 10021136 Jan 22 13:29 image.elf
-rw-rw-r-- 1 petalinux petalinux  6399964 Jan 22 13:29 image.ub
-rw-rw-r-- 1 petalinux petalinux  6974976 Jan 22 13:29 rootfs.cpio
-rw-rw-r-- 1 petalinux petalinux  2887745 Jan 22 13:29 rootfs.cpio.gz
-rw-rw-r-- 1 petalinux petalinux    14680 Jan 22 13:28 system.dtb
-rw-rw-r-- 1 petalinux petalinux  1916015 Jan 22 13:29 System.map.linux
-rwxrwxr-x 1 petalinux petalinux   345128 Jan 22 13:28 u-boot.bin
-rwxrwxr-x 1 petalinux petalinux  2201921 Jan 22 13:28 u-boot.elf
-rwxrwxr-x 1 petalinux petalinux   345128 Jan 22 13:29 u-boot-s.bin
-rwxrwxr-x 1 petalinux petalinux  2201921 Jan 22 13:29 u-boot-s.elf
-rwxrwxr-x 1 petalinux petalinux   992326 Jan 22 13:28 u-boot.srec
-rwxrwxr-x 1 petalinux petalinux   992384 Jan 22 13:29 u-boot-s.srec
-rw-rw-r-- 1 petalinux petalinux  2887809 Jan 22 13:29 urootfs.cpio.gz
-rwxrwxr-x 1 petalinux petalinux 13398545 Jan 22 13:29 vmlinux
-rwxrwxr-x 1 petalinux petalinux  6400528 Jan 22 13:29 zImage
-rw-rw-r-- 1 petalinux petalinux  2083848 Jan 20 15:05 zybo_wrapper.bit
-rwxrwxr-x 1 petalinux petalinux   294857 Jan 22 13:27 zynq_fsbl.elf
```

Figure 4. Various generated files

2-1-3. 아래를 실행하여 /tftpboot 디렉토리의 내용을 검사한다:

[host]$ ls /tftpboot

~/emblnx/labs/lab2/ZYBO_petalinux_v2015_4/images/linux 디렉토리에 있는
모든 파일은 빌드 프로세스의 일부로 이미지 파일도 복사되므로 /tftpboot 에 복사본이 있다.
개발 머신은 TFTP 서버로 구성되어 있기 때문에 보드가 프로젝트 디렉토리의 실제 경로를 알지 않고
고정 된 알려진 위치에서 네트워크를 통해 직접 새 커널 이미지를 가져올 수 있다.

다음 실습에서 이 기능을 사용한다.

Image Name **Linux Kernel and System Images**	**Descriptions**
image.elf	Linux image in ELF format
image.ub	Linux image in U-Boot format
rootfs.cpio	Root file system image
u-boot.bin	U-Boot image in binary format
u-boot.srec	U-Boot image in SREC format
u-boot.elf	U-Boot image in ELF format
u-boot-s.*	Relocatable U-Boot image

Booting the System

3-1. 앞서 언급했듯이 QEMU 에서 ARM Cortex-A9 MPcore 시스템 전용 Linux 를 실행할 수 있다. QEMU 에 ARM Cortex-A9 MPcore Linux 를 로드한다.

3-1-1. Host 터미널 창에서 다음 명령을 입력하여 커널만 로드한다.

[host]$ petalinux-boot --qemu --kernel

```
petalinux@ubuntu:~/emblnx/labs/lab2/ZYBO_petalinux_v2015_4$ petalinux-boot --qemu --kernel
INFO: The image provided is a zImage
INFO: TCP PORT is free
INFO: Starting arm QEMU
INFO:  qemu-system-aarch64 -L /opt/pkg/petalinux-v2015.4-final/etc/qemu -M arm-generic-fdt
   -serial /dev/null -serial mon:stdio -display none -kernel /home/petalinux/emblnx/labs/l
5_4/build/qemu_image.elf -gdb tcp::9000 -dtb /home/petalinux/emblnx/labs/lab2/ZYBO_petalin
/system.dtb -tftp /tftpboot -device loader,addr=0xf8000008,data=0xDF0D,data-len=4 -device
data=0x00500801,data-len=4 -device loader,addr=0xf800012c,data=0x1ed044d,data-len=4 -devic
8,data=0x0001e008,data-len=4
----------------------------------------------------------------
Xilinx QEMU Dec 10 2015 13:38:20.
----------------------------------------------------------------
Uncompressing Linux... done, booting the kernel.
Booting Linux on physical CPU 0x0
Linux version 4.0.0-xilinx (petalinux@ubuntu) (gcc version 4.9.2 (Sourcery CodeBench Lite
EEMPT Fri Jan 22 13:29:07 UTC 2016
CPU: ARMv7 Processor [414fc091] revision 1 (ARMv7), cr=10c5387d
CPU: PIPT / VIPT nonaliasing data cache, VIPT nonaliasing instruction cache
Machine model: ZYBO_petalinux_v2015_4
bootconsole [earlycon0] enabled
cma: Reserved 16 MiB at 0x1f000000
Memory policy: Data cache writealloc
PERCPU: Embedded 11 pages/cpu @debce000 s12672 r8192 d24192 u45056
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 130048
Kernel command line: console=ttyPS0,115200 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: 493208K/524288K available (4759K kernel code, 224K rwdata, 1708K rodata, 3028K ini
erved, 16384K cma-reserved, 0K highmem)
Virtual kernel memory layout:
    vector  : 0xffff0000 - 0xffff1000   (   4 kB)
    fixmap  : 0xffc00000 - 0xfff00000   (3072 kB)
    vmalloc : 0xe0800000 - 0xff000000   ( 488 MB)
    lowmem  : 0xc0000000 - 0xe0000000   ( 512 MB)
    pkmap   : 0xbfe00000 - 0xc0000000   (   2 MB)
    modules : 0xbf000000 - 0xbfe00000   (  14 MB)
      .text : 0xc0008000 - 0xc0658efc   (6468 kB)
      .init : 0xc0659000 - 0xc094e000   (3028 kB)
      .data : 0xc094e000 - 0xc0986020   ( 225 kB)
       .bss : 0xc0986020 - 0xc09ba1b4   ( 209 kB)
Preemptible hierarchical RCU implementation.
        Additional per-CPU info printed with stalls.
        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: platform modifies aux control register: 0x00000000 -> 0x30400000
L2C: DT/platform modifies aux control register: 0x00000000 -> 0x30400000
L2C-310 errata 588369 769419 enabled
L2C-310 full line of zeros enabled for Cortex-A9
L2C-310 cache controller enabled, 8 ways, 64 kB
L2C-310: CACHE_ID 0x00000000, AUX_CTRL 0x00000000
slcr mapped to e0804000
zynq_clock_init: clkc starts at e0804100
Zynq clock init
```

Figure 5. Console output 1

```
zynq_clock_init: clkc starts at e0804100
Zynq clock init
sched_clock: 64 bits at 325MHz, resolution 3ns, wraps every 3383112499200ns
timer #0 at e0808000, irq=17
Console: colour dummy device 80x30
Calibrating delay loop... 2341.27 BogoMIPS (lpj=11706368)
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 0x481788 - 0x4817e0
CPU1: thread -1, cpu 1, socket 0, mpidr 80000001
Brought up 2 CPUs
SMP: Total of 2 processors activated (6819.84 BogoMIPS).
CPU: WARNING: CPU(s) started in wrong/inconsistent modes (primary CPU mode 0x13)
CPU: This may indicate a broken bootloader or firmware.
devtmpfs: initialized
VFP support v0.3: implementor 41 architecture 3 part 30 variant 9 rev 0
pinctrl core: initialized pinctrl subsystem
NET: Registered protocol family 16
DMA: preallocated 256 KiB pool for atomic coherent allocations
cpuidle: using governor ladder
cpuidle: using governor menu
zynq_get_revision: no devcfg node found
hw-breakpoint: debug architecture 0x4 unsupported.
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.
futex hash table entries: 512 (order: 3, 32768 bytes)
jffs2: version 2.2. (NAND) (SUMMARY)  © 2001-2006 Red Hat, Inc.
io scheduler noop registered
io scheduler deadline registered
io scheduler cfq registered (default)
```

Figure 6. Console output 2

```
macb e000b000.ethernet eth0: Cadence GEM rev 0x00020118 at 0xe000b000 irq 27 (00:0a:35:00:1e:53)
macb e000b000.ethernet eth0: attached PHY driver [Marvell 88E1111] (mii_bus:phy_addr=e000b000.etherne:00, irq=-1)
e1000e: Intel(R) PRO/1000 Network Driver - 2.3.2-k
e1000e: Copyright(c) 1999 - 2014 Intel Corporation.
ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
ehci-pci: EHCI PCI platform driver
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
Driver 'mmcblk' needs updating - please use bus_type methods
sdhci: Secure Digital Host Controller Interface driver
sdhci: Copyright(c) Pierre Ossman
sdhci-pltfm: SDHCI platform and OF driver helper
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.
zynq_pm_remap_ocm: no compatible node found for 'xlnx,zynq-ocmc-1.0'
zynq_pm_suspend_init: Unable to map OCM.
Registering SWP/SWPB emulation handler
/opt/pkg/petalinux-v2015.4-final/components/linux-kernel/xlnx-4.0/drivers/rtc/hctosys.c: unable to open rtc devic
e (rtc0)
ALSA device list:
  No soundcards found.
Freeing unused kernel memory: 3028K (c0659000 - c094e000)
INIT: version 2.88 booting
Creating /dev/flash/* device nodes
random: dd urandom read with 0 bits of entropy available
Starting internet superserver: inetd.
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... done.

Built with PetaLinux v2015.4 (Yocto 1.8) ZYBO_petalinux_v2015_4 /dev/ttyPS0
ZYBO_petalinux_v2015_4 login: macb e000b000.ethernet eth0: link up (1000/Full)
```

Figure 7. Console output 3

3-1-2. "First Look" Lab 에서 했던 것처럼 시스템에 로그인 하고 탐색한다.
Note: 로그인 이름과 암호로 root 를 사용한다.

3-1-3. Ctrl + a 를 누른 다음 <x> 키를 눌러 QEMU 를 종료한다.

3-2. BOOT.BIN 파일을 미리 만들어진 디렉토리에서 MICRO-SD 카드로 복사한다.

3-2-1. BOOT.BIN 파일만 ~/emblnx/labs/lab2/ZYBO_petalinux_v2015_4/pre-built/linux/images 디렉토리에서 MICRO-SD 카드로 복사한다.

3-2-2. 보드가 꺼져 있는지 확인한다.

3-2-3. MICRO-SD 카드를 Target Board 에 삽입한다.

3-2-4. Board 가 MICRO-SD 카드에서 부팅되도록 설정되어 있는지 확인한다.

3-3. HOST 에서 DHCP 서버를 실행한다.

3-3-1. DHCP 서버를 실행한다.

[host]$ sudo service isc-dhcp-server restart

3-4. Board 의 전원을 켜고 Serial 포트 터미널을 설정한다.

3-4-1. Board 의 전원을 켠다.

3-4-2. 다음 명령을 실행하여 /dev/ttyUSB1 이 read/write 접근으로 설정되어 있는지 확인한다.

[host]$ sudo chmod 666 /dev/ttyUSB1

3-4-3. dashboard 의 Search 필드에 Serial 포트를 입력하라.

3-4-4. Serial 포트 터미널 응용 프로그램을 선택하라.

3-5. 새로운 리눅스 이미지를 보드에서 부팅한다.

3-5-1. 보드가 부팅 프로세스를 진행할 때 GtkTerm 콘솔에서 부팅 정보를 보려면 보드(BTN7)를 재설정한다.

3-5-2. GtkTerm 창에 아래와 유사한 메시지가 나타나면 아무 키나 눌러 자동 부팅을 중지한다:

```
U-Boot 2015.07 (Jan 21 2016 - 07:27:49 +0000)

DRAM:  ECC disabled 512 MiB
MMC:   zynq_sdhci: 0
SF: Detected S25FL128S_64K with page size 256 Bytes, erase size 64 KiB, total 16 MiB
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   Gem.e000b000
U-BOOT for ZYBO_petalinux_v2015_4

Hit any key to stop autoboot:  0
U-Boot-PetaLinux>
```

Figure 8. Stopping the autoboot

3-5-3. uboot 부팅 중에 "address 에 DHCP 클라이언트 바인딩" 메시지가 표시되지 않으면 dhcp 를 실행하여 IP 주소를 얻어야 한다.

U-Boot-PetaLinux> dhcp

```
Hit any key to stop autoboot:  0
U-Boot-PetaLinux> dhcp
Gem.e000b000 Waiting for PHY auto negotiation to complete........ done
BOOTP broadcast 1
BOOTP broadcast 2
BOOTP broadcast 3
DHCP client bound to address 192.168.1.2 (1009 ms)
U-Boot-PetaLinux>
```

Figure 9. Running DHCP to obtain the IP address

3-5-4. u-boot 콘솔에서 아래 명령을 실행하여 TFTP 서버 IP 를 Host IP 로 설정한다.

U-Boot-PetaLinux> set serverip 192.168.1.1

3-5-5. u-boot 콘솔에서 다음 명령을 실행하여 TFTP 를 사용하여 새로운 Image 를 다운로드하고 부팅한다.

U-Boot-PetaLinux> run netboot

이 명령은 Host 의 /tftpboot 에서 ARM Cortex-A9 MPcore 시스템의 주 메모리로 image.ub 파일을 다운로드하고 이미지로 시스템을 부팅한다.

3-5-6. GtkTerm 창을 보라.
다음과 유사한 메시지가 Image 다운로드 진행률을 표시한다.

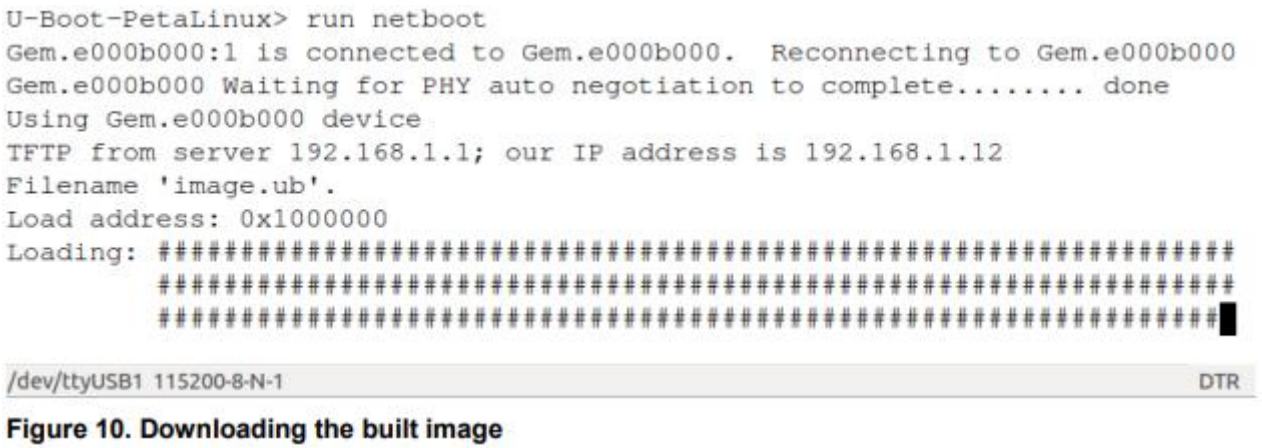

Figure 10. Downloading the built image

netboot 명령은 Image 다운로드가 끝나자마자 자동으로 System 을 부팅한다.

3-5-7. GtkTerm 창에서 부팅 메시지를 본다.
다른 부팅 메시지는 기본 구성을 사용했으므로 Lab1 과 동일하다.

3-6. ping 명령을 사용하여 Network 연결을 테스트한다.

3-6-1. System 이 부팅 된 후 로그인 이름과 암호로 root 를 입력하여 System 에 로그인 한다.

3-6-2. ping 명령을 실행하여 Host System 을 ping 한다.

 # ping 192.168.1.1

Host System 에서 응답을 확인해야 한다.

3-6-3. Host 컴퓨터 터미널 창에서 ping 명령을 실행하여 Target Board 의 응답을 확인한다.

[host]$ ping 192.168.1.2

Board 가 다른 주소에 바인드 된 경우 다른 IP 주소를 사용하라(주소를 찾으려면 그림 8 참조)
Host System 에서 응답을 확인해야 한다.

3-7. 리눅스에서 소프트 재부팅을 수행한다.

3-7-1. Serial 터미널 창 도구에서 reboot 명령을 실행하여 시스템을 재부팅한다.

 # reboot

System 이 재부팅 되어야 한다.

3-7-2. GtkTerm 창을 닫는다.

3-7-3. Board 전원을 끈다.

Conclusion

이 Lab 에서 아래를 수행하는 방법을 배웠다:

- 리눅스 크로스 컴파일
- QEMU 에서 ARM Cortex-A9 MPcore 시스템 전용 리눅스 부팅
- 이더넷을 통해 보드에 새로운 이미지를 다운로드한다.

후속 Lab 에서도 이와 같은 기능을 사용한다.

Completed Solution

솔루션을 실행하려면 labsolution/lab2/SDCard 디렉토리의 BOOT.bin 을 MICRO-SD 카드에 복사한다.
MICRO-SD 카드를 Zybo 에 놓는다.
Zybo 를 MICRO-SD 카드 부팅 모드로 설정한다.
이더넷 케이블을 사용하여 Zybo 를 Host 컴퓨터에 연결한다.

아래 명령을 실행하여 Host 에서 DHCP 서버를 시작한다.

```
[host]$ sudo service isc-dhcp-server restart
```

labsolution/lab2/tftpboot 디렉토리의 image.ub 파일을 /tftpboo 디렉토리로 복사한다.

보드의 전원을 켠다.
터미널 세션을 설정한다.
자동 부팅 메시지가 표시되면 부팅 프로세스를 중단한다.
Target Board 터미널 창에서 다음 명령을 사용하여 server ip 주소를 설정한다.

```
#set serverip 192.168.1.1
```

netboot 명령을 실행한다.

```
#run netboot
```

시스템에 로그인하고 Lab 을 테스트한다.

Appendix A. General setup of lab network

이 절에서는 PC, QEMU 및 개발 보드 간의 네트워크 구성에 대해 설명한다.

2 개의 클래스 C 서브넷 "192.168.1.*" 및 "10.0.2.*" 이 도입되었다.
서브넷 "192.168.0.*" 은 그림과 같이 물리적 연결(PC 및 보드의 eth0)을 통해
PC 와 보드 간의 네트워크 연결을 설정하는데 사용된다.

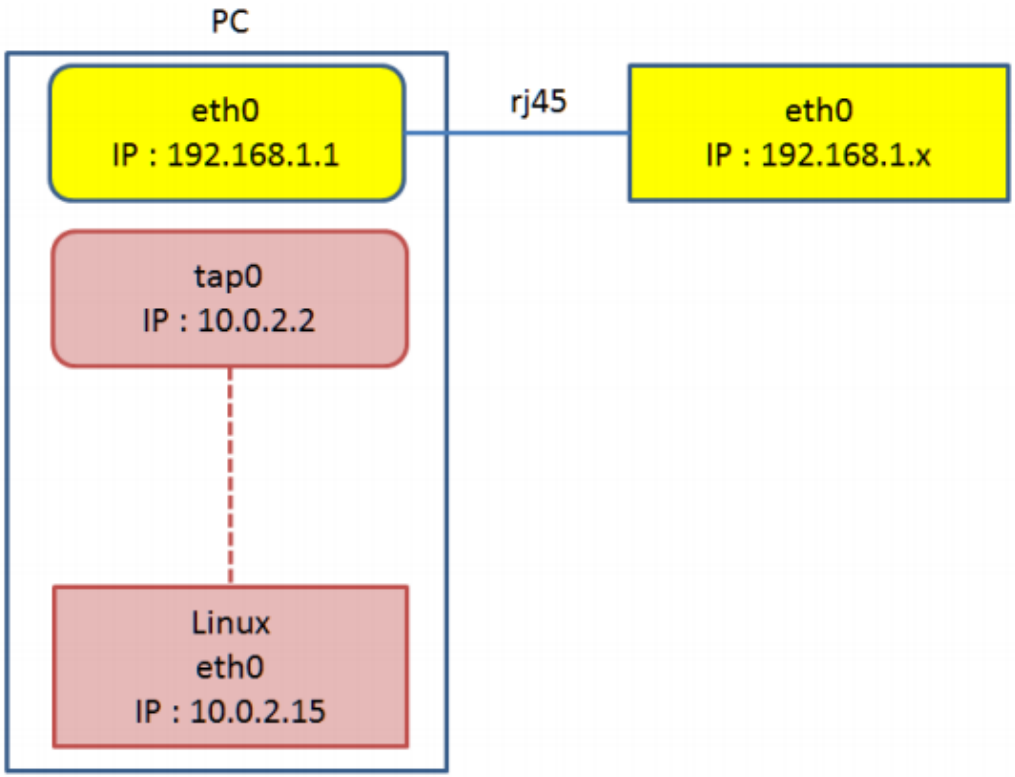

Figure 10. Network Configuration

서브넷 "10.0.2.*" 은 가상 네트워크를 통한 통신 링크를 제공한다.
QEMU 는 TAP 인터페이스(이 경우 "tap0" 인터페이스)를 사용하여
ARM Cortex-A9 MPcore Linux Guest OS 에 전체 네트워크 기능을 제공한다.

이 두 클래스 C 서브넷은 서로 통신 할 수 없다.
예로 개발 보드는 PC 와 통신 할 수 있지만 ARM Cortex-A9 MPcore Linux Guest OS 와는 통신 할 수 없다.

References

https://www.xilinx.com/support/university/vivado/vivado-workshops/Vivado-embedded-linux-zynq.html