

## Lab1. Petalinux Linux System Build

ZynqMP PS 로만 구성된 Vivado Project HW를 구성하고, Petalinux 명령어를 사용하여 sdcard로 부팅가능한 Linux System을 생성하고 Ultra96 보드를 가지고 테스트한다. 또한 Linux System에 추가될 Program들을 위한 개발환경을 위한 SDK(Software Development Kit)를 Build하고 마지막으로 Petalinux Project를 배포할 Petalinux BSP를 만든다.

### 1. Source

```
$ mkdir ~/work  
$ cd ~/work  
$ git clone https://github.com/inipro/zynqmp_linux.git  
$ cd zynqmp_linux
```

### 2. Export Vivado Project

Ultra96v1(hw1\_v1.tcl) 또는 Ultra96v2(hw1\_v2.tcl) Vivado Project를 만든다

```
$ vivado -nolog -nojournal -mode batch -source hw1_v1.tcl  
$ cd hw1  
$ vivado hw1.xpr
```

또는

```
$ vivado -nolog -nojournal -mode batch -source hw1_v2.tcl  
$ cd hw1  
$ vivado hw1.xpr
```

Bitstream을 생성한다

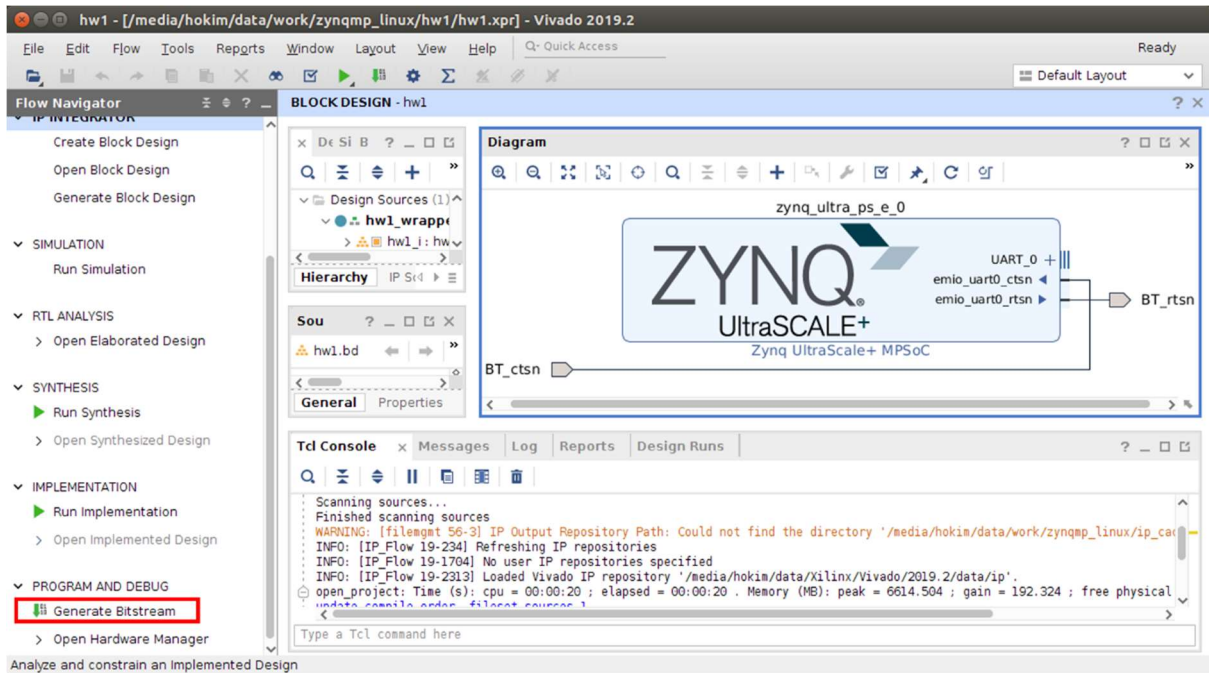


그림 1 Vivado project

HW를 Export 한다.(File → Export → Export Hardware)

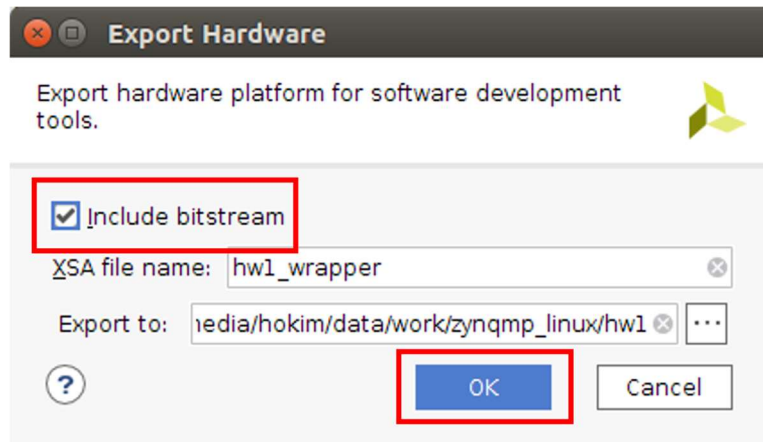


그림 2 Export Hardware

### 3. Petalinux Project 생성 및 구성

hw1/ 폴더의 xsa파일에 기초하여 Petalinux project를 만든다

```
$ cd ~/work/zynqmp_linux/petalinux
$ petalinux-create -t project -n ultra96 --template zynqMP
$ cd ultra96
$ petalinux-config --get-hw-description=../hw1/
```

Petalinux Configuration 화면이 다음처럼 나타난다.

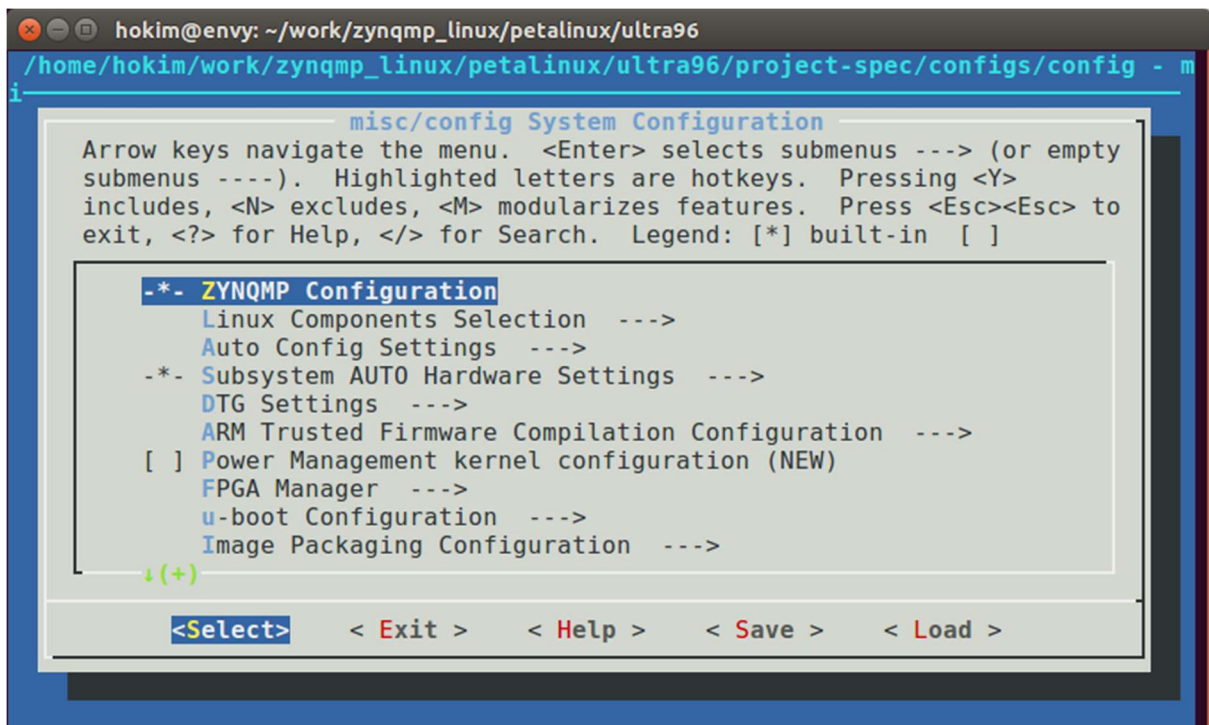


그림 3 Petalinux Configuration

Subsystem AUTO Hardware Setting → Serial Settings → Primary stdin/stdout 에서  
uart1을 선택한다.

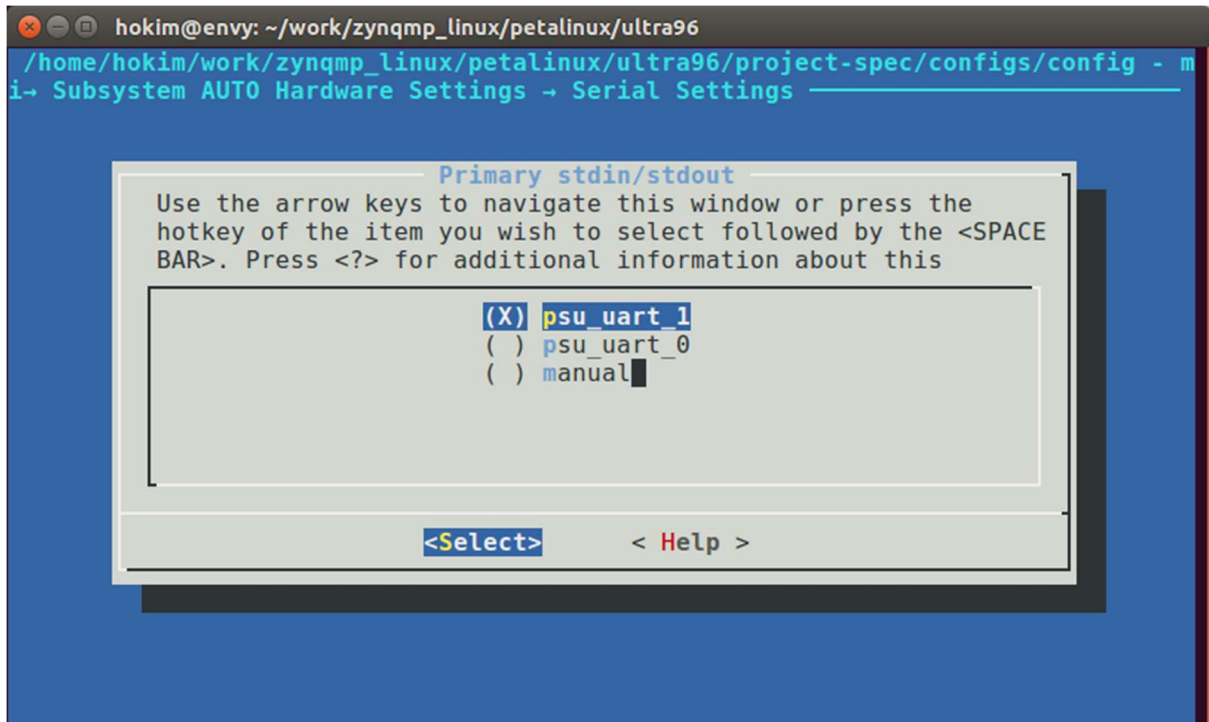


그림 4 Petalinux Configuration(Primary stdin/stdout)

DTG Settings → MACHINE\_NAME에 avnet-ultra96-rev1을 입력한다.

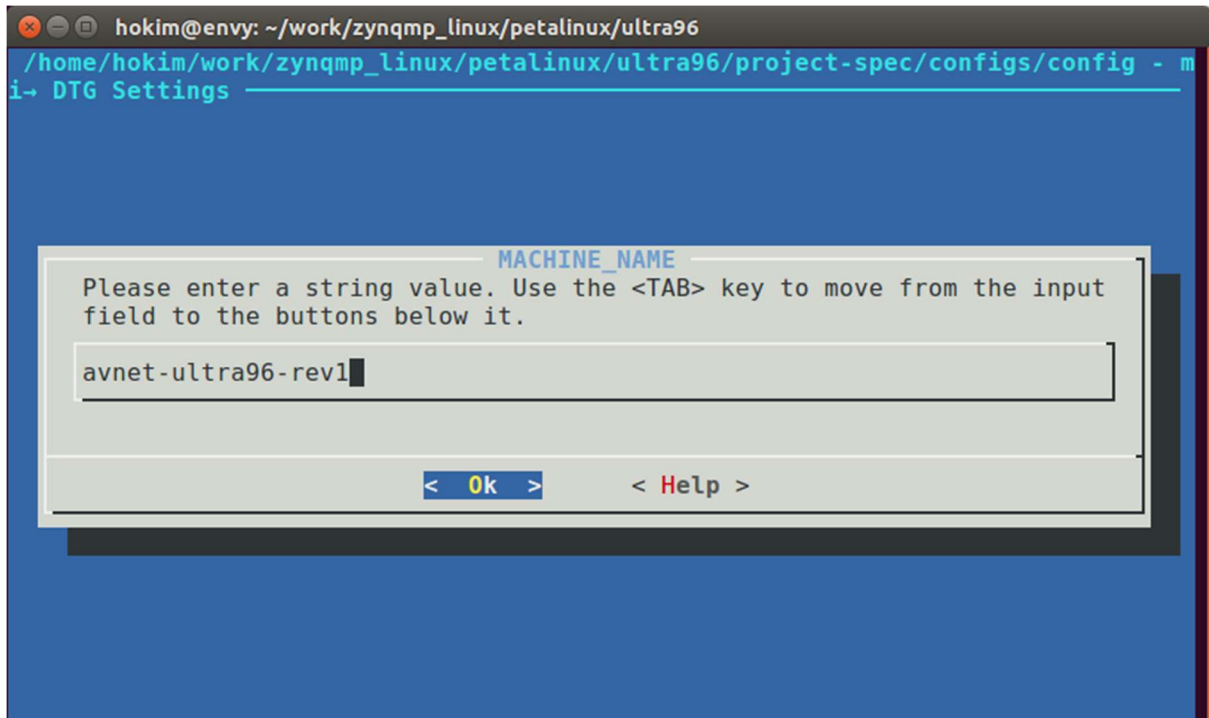


그림 5 Petalinux Configuration(DTG MACHINE\_NAME)

DTG Settings → Kernel Bootargs 에서 generate boot args automatically를 선택해

제하고 user set kernel bootargs에 earlycon console=ttyPS0,115200  
clk\_ignore\_unused root=/dev/mmcblk0p2 rw rootwait  
uio\_pdrv\_genirq.of\_id=xlnx,generic-uio cma=512M를 입력한다.

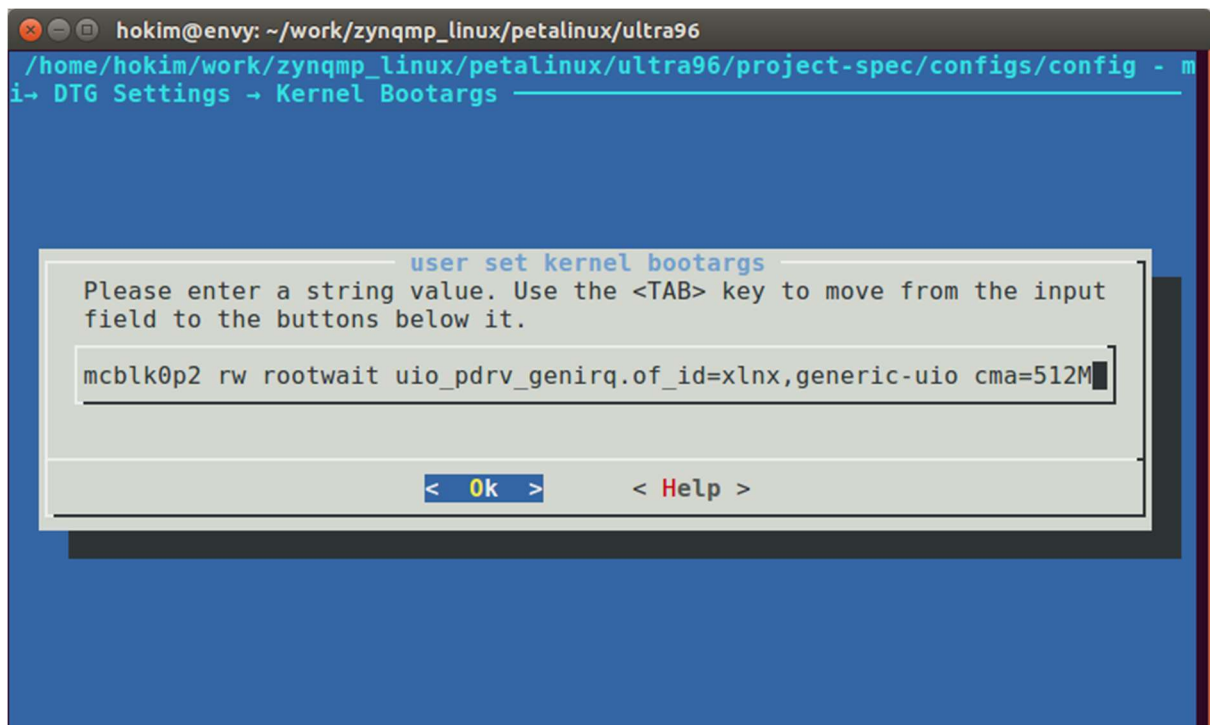


그림 6 Petalinux Configuration(Kernel Bootargs)

u-boot Configuration → u-boot config target에 avnet\_ultra96\_rev1\_defconfig를 입력한다.

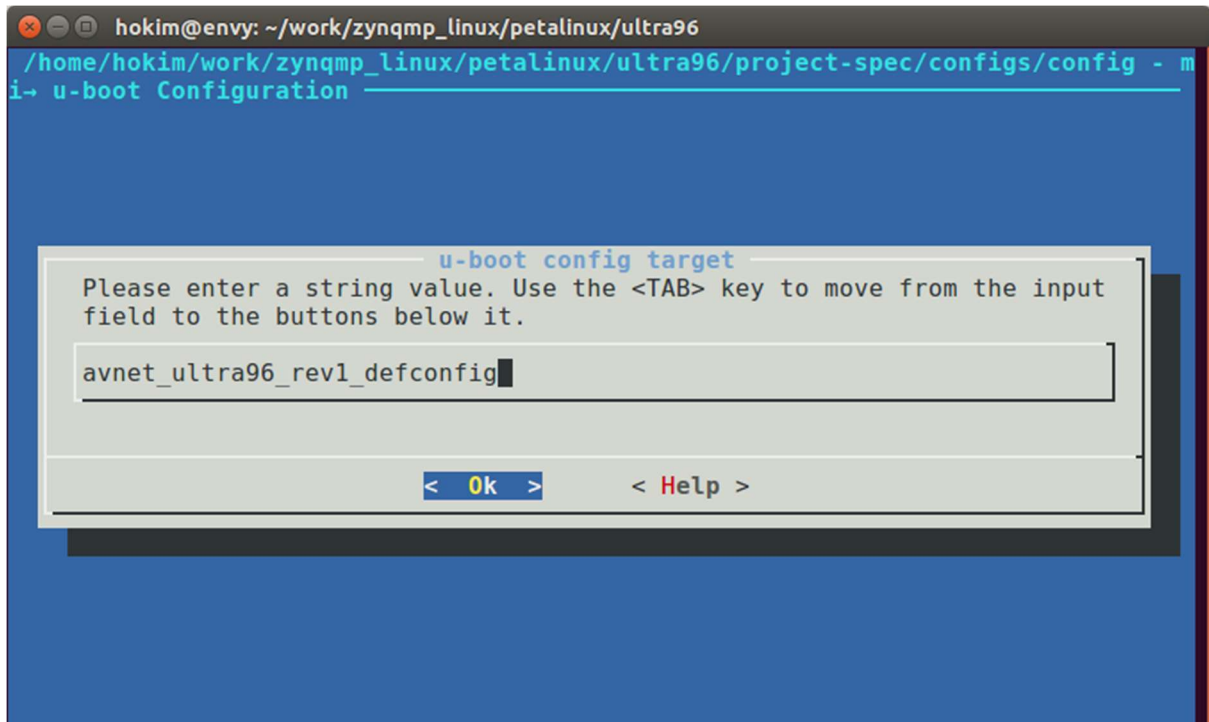


그림 7 Petalinux Configuration(u-boot config target)

Image Packaging Configuration → Root filesystem type에서 EXT (SD/eMMC/QSPI/SATA/USB)을 선택한다.

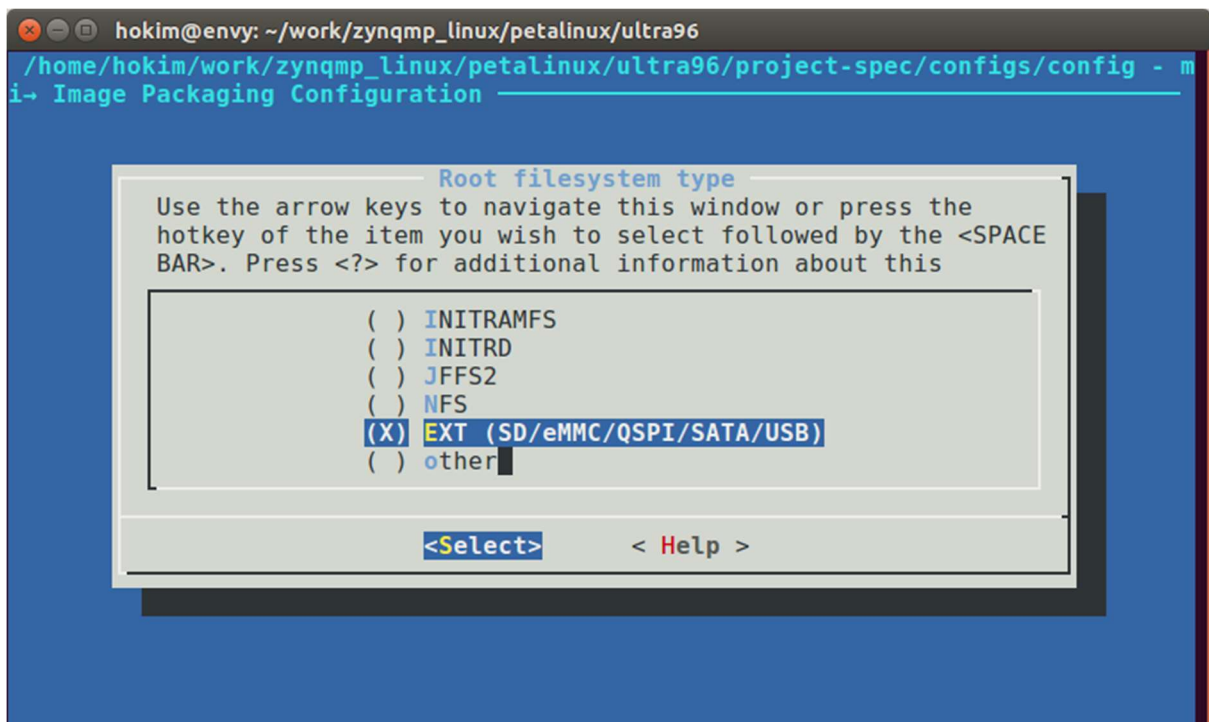


그림 8 Petalinux Configuration(Root filesystem type)

Image Packaging Configuration에서 Copy final images to tftpboot를 선택해제 한다.

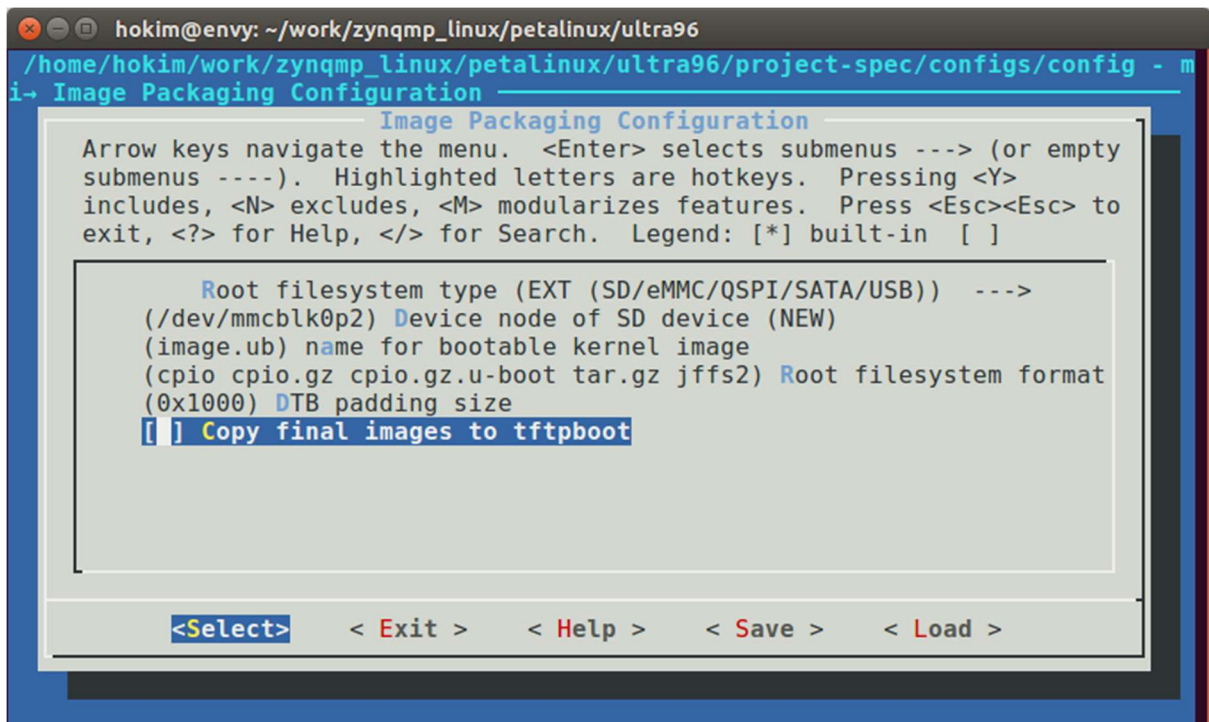


그림 9 Petalinux Configuration(disable tftpboot copy)

Yocto Settings → YOCTO\_MACHINE\_NAME에 ultra96-zynqmp를 입력한다.



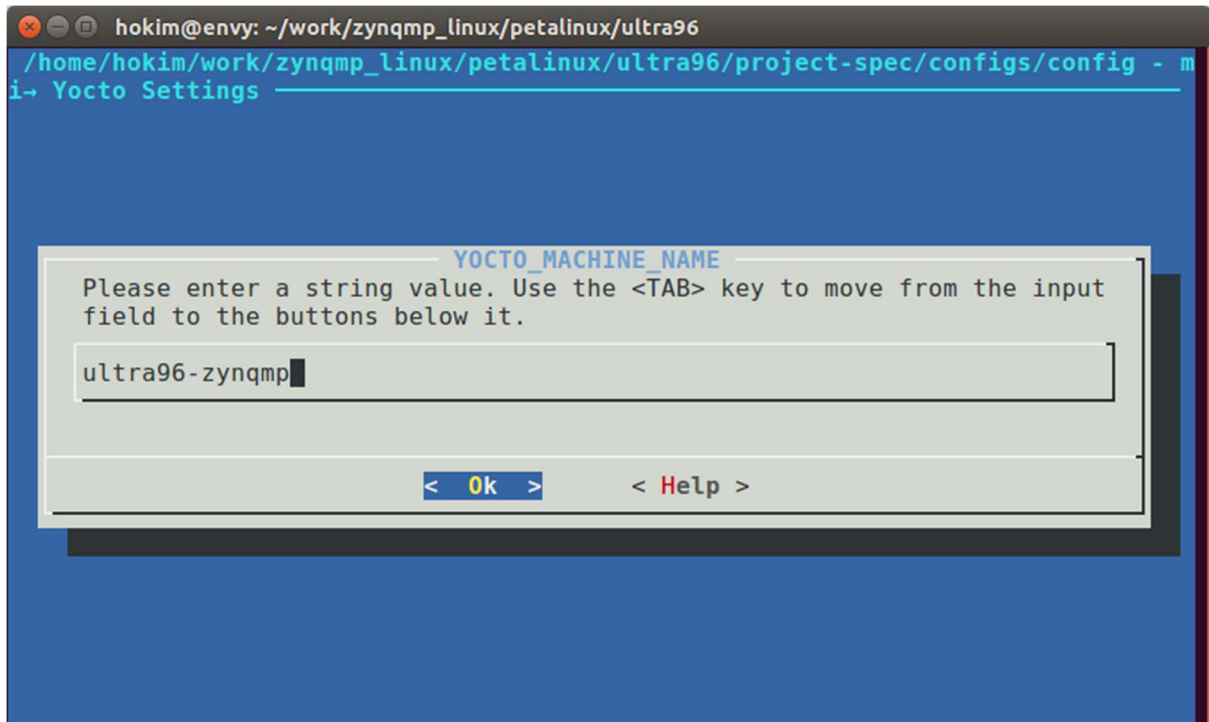


그림 10 Petalinux Configuration(YOCTO\_MACHINE\_NAME)

Yocto Settings → Add pre-mirror url에 Xilinx downloads의 폴더경로를 지정한다.  
입력된 /media/hokim/data/downloads\_2019.2을 사용자의 환경에 맞게 변경한다.

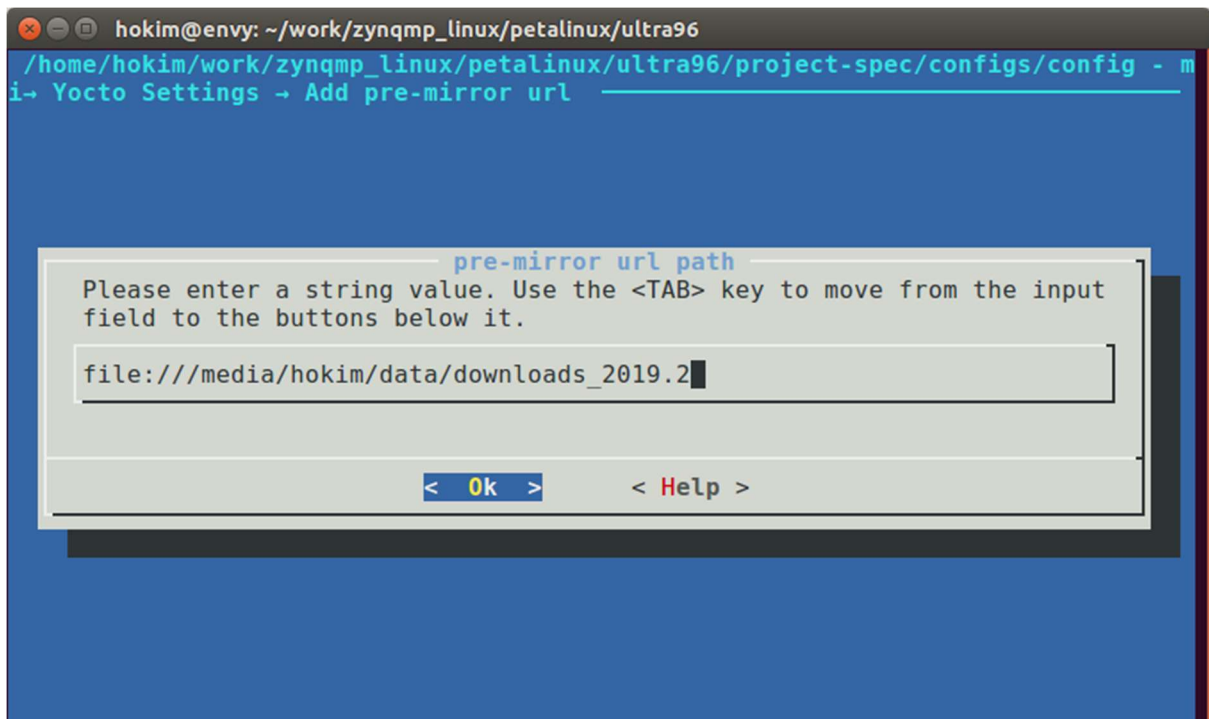


그림 11 Petalinux Configuration(pre-mirror url path)



Yocto Settings → Local sstate feeds settings에 Xilinx sstate의 폴더경로를 지정한다. 입력된 /media/hokim/data/sstate\_aarch64\_2019.2를 사용자의 환경에 맞게 변경한다.

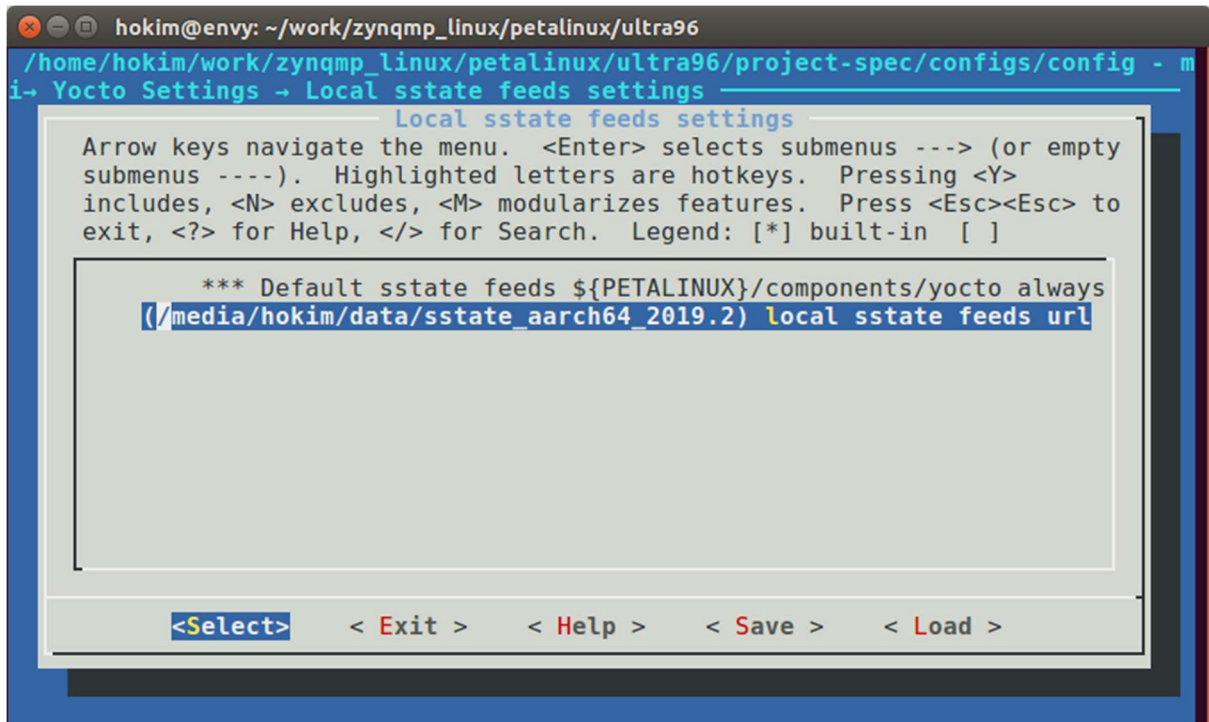


그림 12 Petalinux Configuration(Local sstate feeds)

Yocto Settings → User Layers에 \${PROOT}/../meta-inipro를 입력한다.

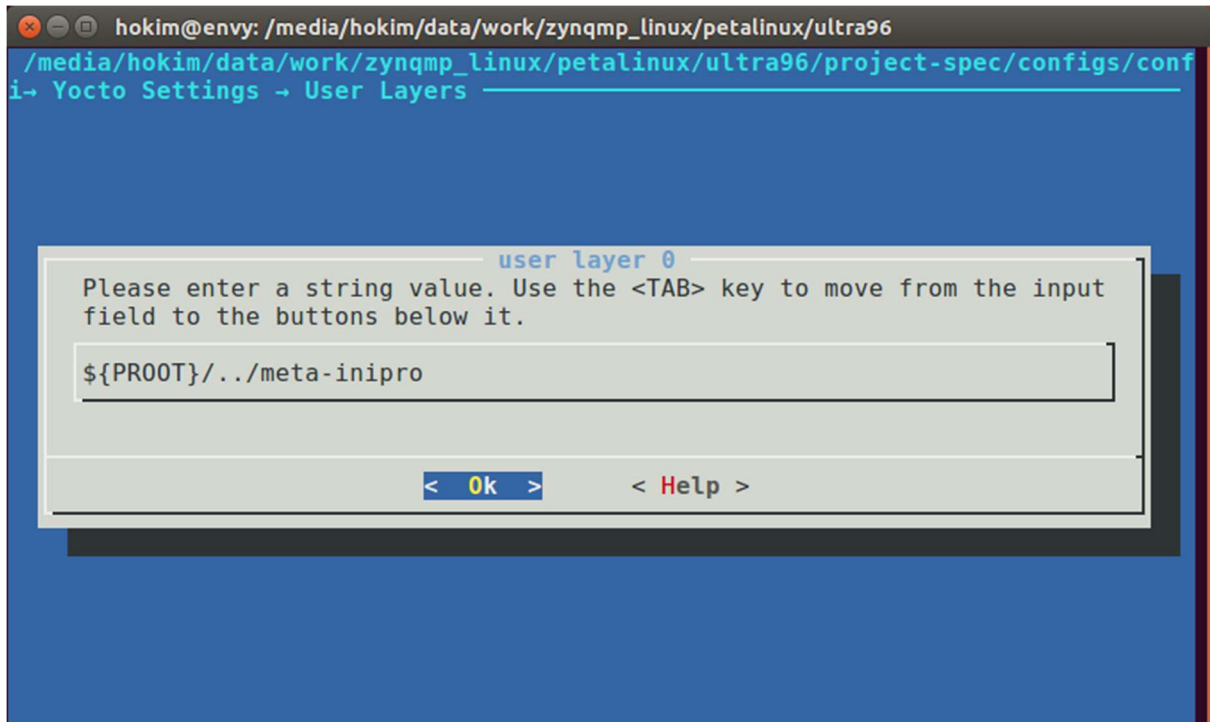


그림 13 Petalinux Configuration(User Layers)

설정을 마치고 종료한다.

#### 4. Petalinux bsp Configuration

편집기(vi, gedit, ...)를 사용하여 Petalinux Project(ultra96) 폴더아래의 project-spec/meta-user/conf/petalinuxbsp.conf에 다음처럼 line17부터의 내용을 추가한다.

SSTATE\_MIRRORS\_append의 폴더경로는 사용자의 환경에 맞게 file:///media/hokim/data/sstate\_aarch64\_2019.2\_2부분을 수정한다.

```
hokim@envy: /media/hokim/data/work/zynqmp_linux/petalinux/ultra96
10
11 #Remove all qemu contents
12 IMAGE_CLASSES_remove = "image-types-xilinx-qemu qemuboot-xilinx"
13 IMAGE_FSTYPES_remove = "wic.qemu-sd"
14
15 EXTRA_IMAGEDEPENDS_remove = "qemu-helper-native virtual/boot-bin"
16
17 MACHINE_FEATURES_remove = "mipi"
18
19 DISTRO_FEATURES_append = " bluez5 dbus"
20
21 EXTRA_IMAGE_FEATURES += "package-management"
22
23 PACKAGE_FEED_URI = "http://192.168.2.50:5678"
24
25 IMAGE_ROOTFS_EXTRA_SPACE = "102400"
26
27 SIGGEN_UNLOCKED_RECIPES += "tzdata dnf-native dropbear dtc-native cmake-native"
28
29 SSTATE_MIRRORS_append = " \
30 file://.* file:///media/hokim/data/sstate_aarch64_2019.2_2/PATH \n \
31 "
10,0-1 Bot
```

그림 14 petalinuxbsp.conf(Ultra96v1)

```
hokim@envy: /media/hokim/data/work/zynqmp_linux/petalinux/ultra96
14
15 EXTRA_IMAGEDEPENDS_remove = "qemu-helper-native virtual/boot-bin"
16
17 MACHINE_FEATURES_remove = "mipi"
18
19 DISTRO_FEATURES_append = " bluez5 dbus"
20
21 EXTRA_IMAGE_FEATURES += "package-management"
22
23 PACKAGE_FEED_URI = "http://192.168.2.50:5678"
24
25 IMAGE_ROOTFS_EXTRA_SPACE = "102400"
26
27 SIGGEN_UNLOCKED_RECIPES += "tzdata dnf-native dropbear dtc-native cmake-native"
28
29 PREFERRED_VERSION_wilc-firmware = "15.2"
30
31 ULTRA96_VERSION_ultra96-zynqmp = "2"
32
33 SSTATE_MIRRORS_append = " \
34 file://.* file:///media/hokim/data/sstate_aarch64_2019.2_2/PATH \n \
35 "
```

그림 15 petalinuxbsp.con(Ultra96v2)

## 5. Device Tree Configuration

다음의 명령을 수행하여 xsa 파일과 그림5의 DTG MACHINE NAME에 기초한

Device Tree를 Generation 한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96  
$ petalinux-build -c device-tree -x configure
```

Petalinux Project(ultra96) 폴더 아래의 components/plnx\_workspace/device-tree/device-tree/에서 \*.dtsi, system-top.dts 들이 생성되었음을 확인한다. 사용자의 요구에 맞게 Device Tree의 내용을 변경하기 위해서는 project-spec/meta-user/recipes-bsp/device-tree/system-user.dtsi를 편집기로 열어서 Device Tree를 수정한다. Ultra96v1 보드는 avnet-ultra96-rev1.dtsi에 보드에 맞는 Device Tree정보가 들어있어서 수정할 내용은 없고 Ultra96v2 보드를 위해서는 다음과 같이 수정해야 한다.

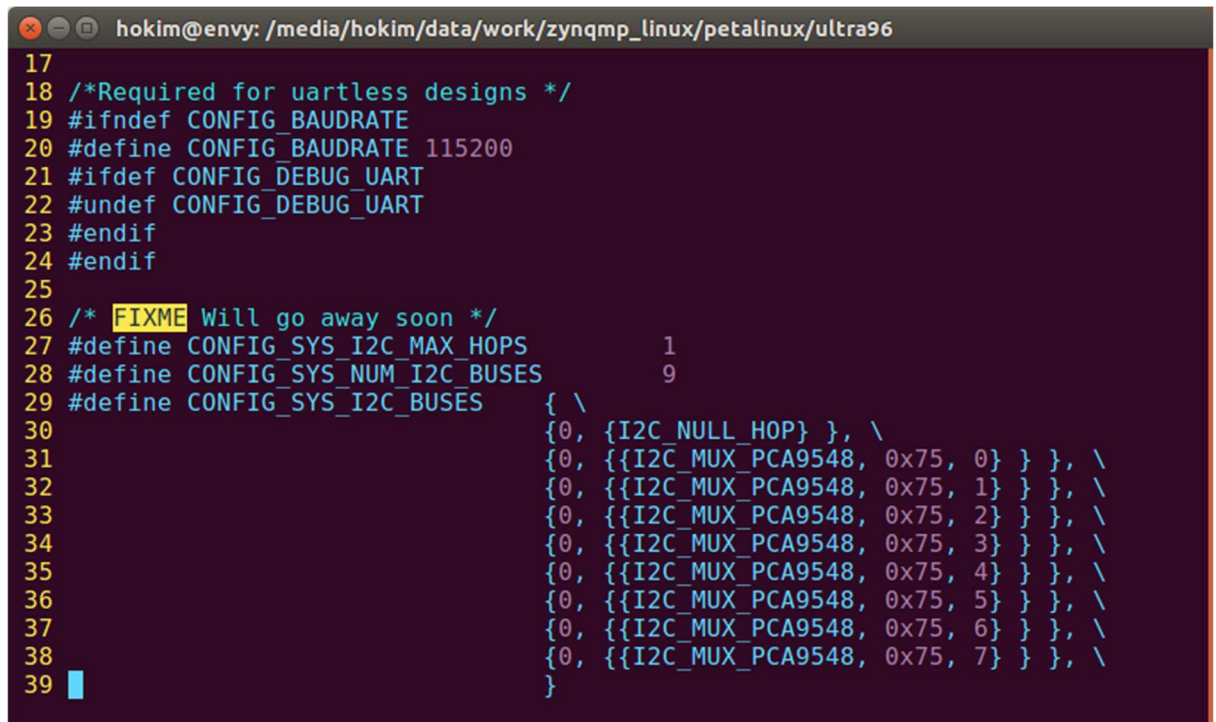
```
hokim@envy: /media/hokim/data/work/zynqmp_linux/petalinux/ultra96
1 /include/ "system-conf.dtsi"
2 / {
3     /delete-node/ ltc2954;
4 };
5
6 &sdio_pwrseq {
7     chip_en-gpios = <&gpio 8 1>; // requires a patched pwrseq_simple.c for W
    ILC3000
8 };
9
10 &gpio {
11     /delete-property/gpio-line-names;
12 };
13
14 &i2csw 4 {
15     /delete-node/ pmic@5e;
16     irps5401_13: irps5401@13 {
17         compatible = "infineon,irps5401";
18         reg = <0x13>;
19     };
20     irps5401_14: irps5401@14 {
21         compatible = "infineon,irps5401";
22         reg = <0x14>;
23     };
24     ir38060_15: ir38060@15 {
25         compatible = "infineon,ir38060";
26         reg = <0x15>;
27     };
28 };
29
30 &i2csw 5 {
31     /delete-node/ ina226@40;
32 };
33
34 &sdhci1 {
35     max-frequency = <50000000>;
36     /delete-property/cap-power-off-card;
37     /delete-node/ wifi@2;
38     wilc_sdio@1 {
39         compatible = "microchip,wilc3000";
40         reg = <0>;
41         bus-width = <0x4>;
42     };
43 };
44
45 &uart0 {
46     /delete-node/ bluetooth;
47 };
```

그림 16 system-user.dtsi(Ultra96v2)

## 6. u-boot Configuration

그림7의 u-boot config target config의 설정에 의해 u-boot는 configuration되고 설정에 관한 기본정보는 project-spec/meta-plnx-generated/recipes-bsp/u-boot/configs/config.cfg, platform-auto.h와

project-spec/meta-user/recipes-bsp/u-boot/files/platform-top.h에서 확인할 수 있다. 변경사항을 위해서는 platform-top.h를 수정한다. 여기서는 i2c mux를 u-boot에 추가하기 위해 다음과 같이 line26부터의 내용을 추가한다.

A terminal window with a dark background and light-colored text. The window title is 'hokim@envy: /media/hokim/data/work/zynqmp\_linux/petalinux/ultra96'. The terminal shows a C preprocessor file with line numbers 17 through 39. Lines 18-25 contain comments and definitions for CONFIG\_BAUDRATE and CONFIG\_DEBUG\_UART. Line 26 has a comment '/\* FIXME Will go away soon \*/'. Lines 27-39 define CONFIG\_SYS\_I2C\_MAX\_HOPS as 1, CONFIG\_SYS\_NUM\_I2C\_BUSES as 9, and CONFIG\_SYS\_I2C\_BUSES as an array of 8 I2C mux configurations for PCA9548 chips, each with a unique address (0x75 to 0x7B).

```
17
18 /*Required for uartless designs */
19 #ifndef CONFIG_BAUDRATE
20 #define CONFIG_BAUDRATE 115200
21 #ifdef CONFIG_DEBUG_UART
22 #undef CONFIG_DEBUG_UART
23 #endif
24 #endif
25
26 /* FIXME Will go away soon */
27 #define CONFIG_SYS_I2C_MAX_HOPS 1
28 #define CONFIG_SYS_NUM_I2C_BUSES 9
29 #define CONFIG_SYS_I2C_BUSES { \
30     {0, {I2C_NULL_HOP} }, \
31     {0, {{I2C_MUX_PCA9548, 0x75, 0} } }, \
32     {0, {{I2C_MUX_PCA9548, 0x75, 1} } }, \
33     {0, {{I2C_MUX_PCA9548, 0x75, 2} } }, \
34     {0, {{I2C_MUX_PCA9548, 0x75, 3} } }, \
35     {0, {{I2C_MUX_PCA9548, 0x75, 4} } }, \
36     {0, {{I2C_MUX_PCA9548, 0x75, 5} } }, \
37     {0, {{I2C_MUX_PCA9548, 0x75, 6} } }, \
38     {0, {{I2C_MUX_PCA9548, 0x75, 7} } }, \
39 }
```

그림 17 platform-top.h

## 7. Kernel Configuration

Kernel Configuration을 위해 다음의 명령을 수행한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96
$ petalinux-config -c kernel
```

Kernel Configuration메뉴에서 Virtual Video Test Driver를 다음처럼 활성화시킨다.



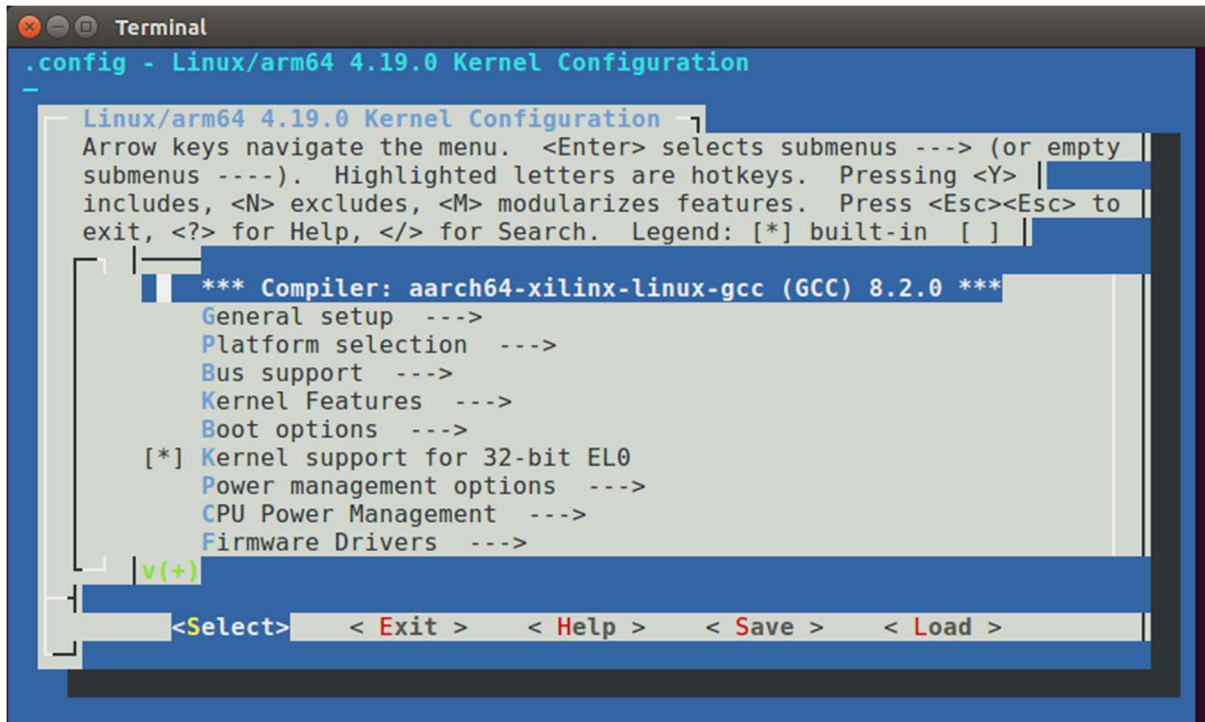


그림 18 Kernel Configuration 1

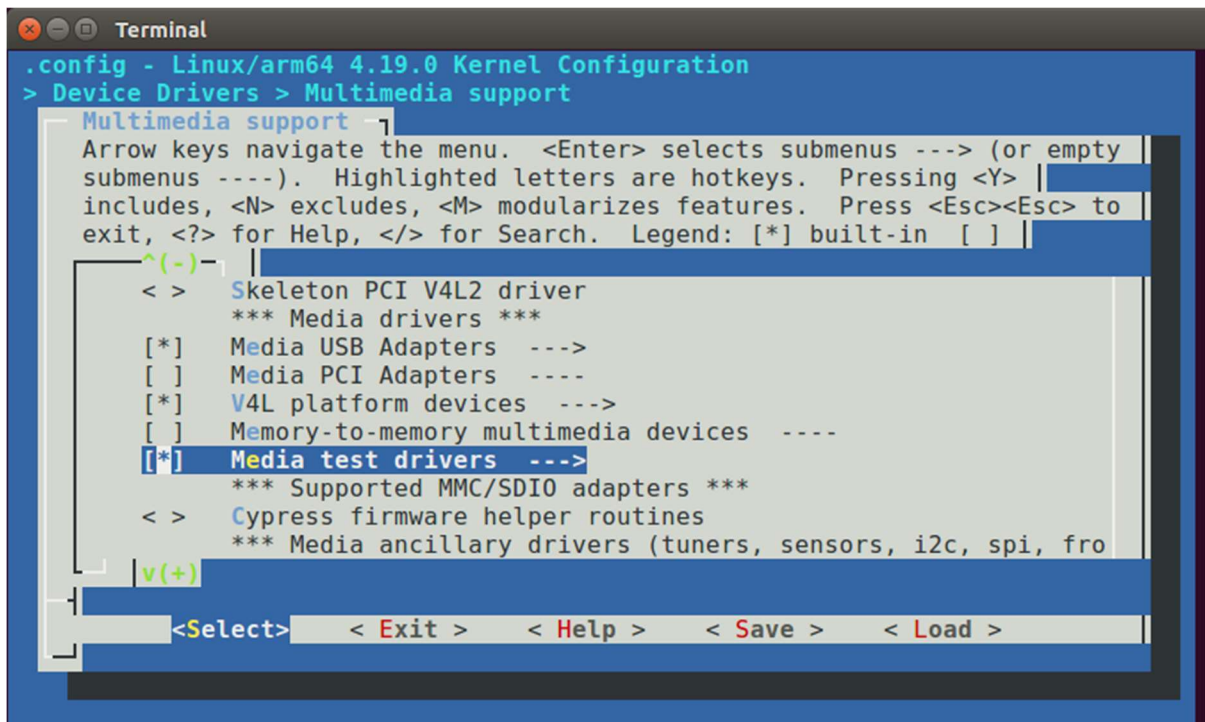


그림 19 Kernel Configuration 2



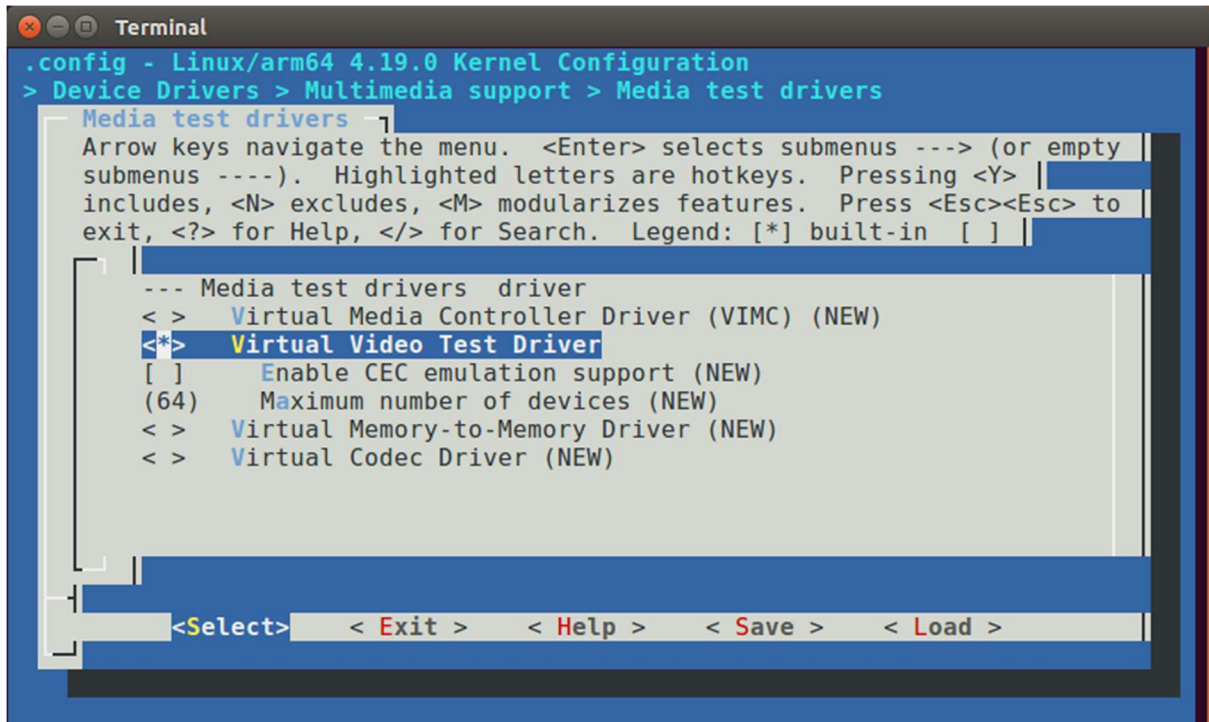


그림 20 Kernel Configuration 3

Petalinux Project(ultra96) 폴더 아래에 components/plnx\_workspace/sources/linux-xlnx/.config.new의 추가된 설정을 갖는 파일이 생성된다. 이 파일은 다음의 Kernel Configuration에서 다시 default로 재설정되기 때문에 일시적이다. 추가된 설정내용을 보존하기 위해 다음의 과정을 수행한다.

```
$ petalinux-config
```

Petalinux Configuration에서 Yocto Settings → Build tool에서 다음을 선택하고 종료한다.

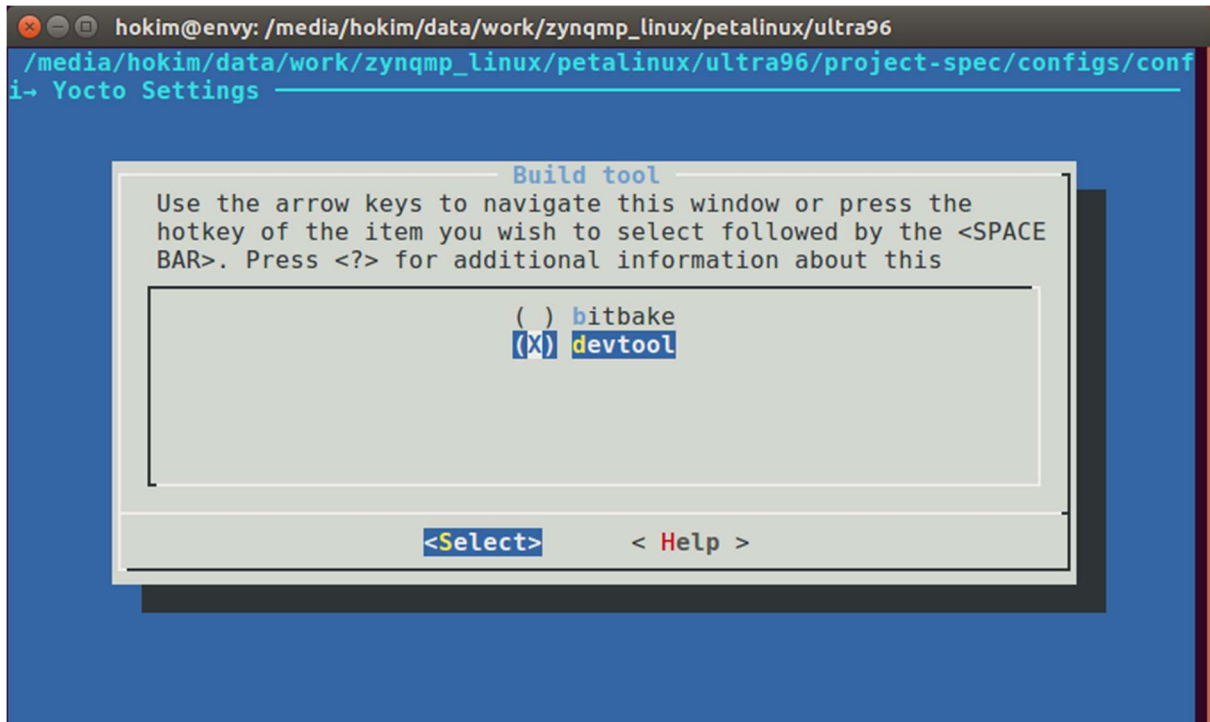


그림 21 Petalinux Configuration(devtool)

```
$ petalinux-build -c kernel -x update-recipe
```

위의 명령에 의해 생성된 project-spec/meta-user/recipes-kernel/linux/linux-xlnx 폴더아래의 linux-xlnx\_2019.2.bbappend와 linux-xlnx/devtool-fragment.cfg 파일들을 확인한다.

다음 명령을 통해 다시 bitbake tool로 돌아온다.

```
$ petalinux-config
```

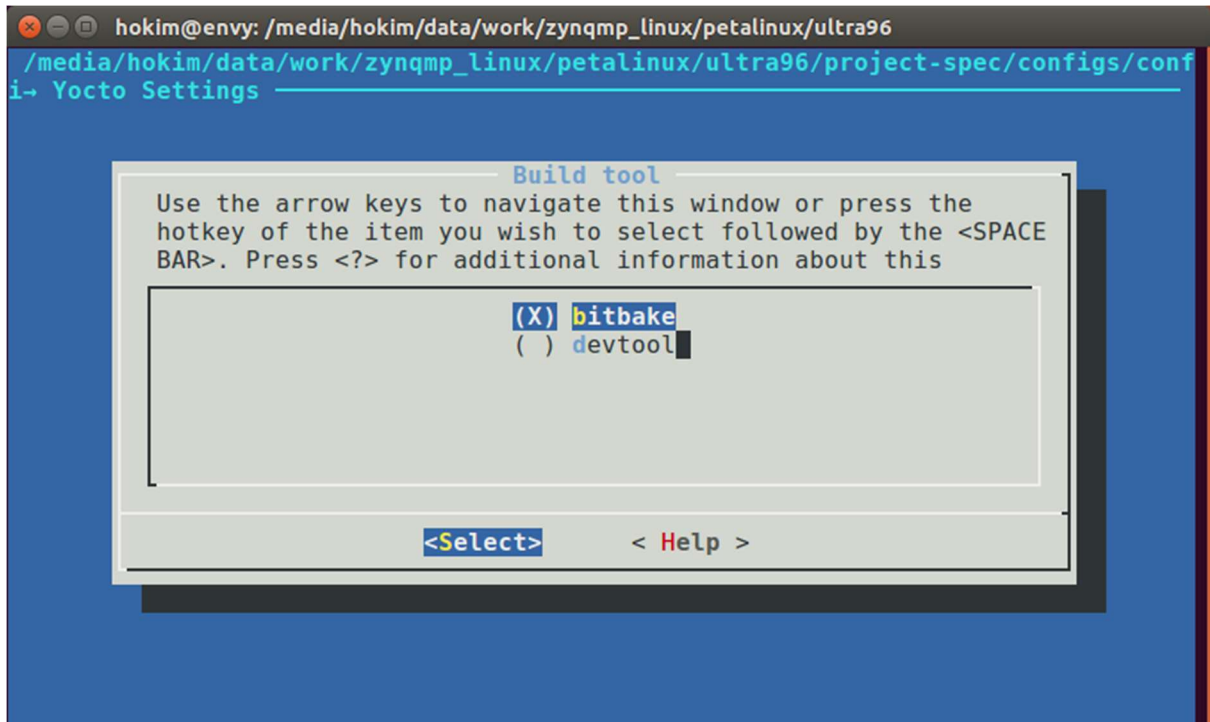


그림 22 Petalinux Configuration(bitbake)

다음의 명령은 Configuration을 위해 사용했던 Kernel Source를 cleanup 한다.

```
$ petalinux-build -c kernel -x reset
```

## 8. Image Configuration

Petalinux Project(ultra96)폴더 아래에서 다음의 명령을 사용하여 root 계정의 비밀번호(line1의 xxxx)의 변경, root filesystem에 설치될 package 목록들 (line2-34), SDK를 위한 설정들(line35-37)을 입력한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96
$ mkdir -p project-spec/meta-user/recipes-core/images
$ vi project-spec/meta-user/recipes-core/images/petalinux-user-image.bbappend
```

```

hokim@envy: /media/hokim/data/work/zynqmp_linux/petalinux/ultra96
1 EXTRA_USERS_PARAMS = "usermod -P xxxx root;"
2 IMAGE_INSTALL_append = " nano \
3                          tzdata \
4                          dtc \
5                          kmod \
6                          e2fsprogs-resize2fs \
7                          i2c-tools \
8                          iw \
9                          wpa-suplicant \
10                         ultra96-power-button \
11                         bluez5 \
12                         ${@bb.utils.contains('ULTRA96_VERSION', '2', 'wilc-fi
rmware-wilc3000', '', d)} \
13                         ${@bb.utils.contains('ULTRA96_VERSION', '2', 'wilc',
'', d)} \
14                         cmake \
15                         packagegroup-petalinux-self-hosted \
16                         packagegroup-petalinux-openamp \
17                         packagegroup-petalinux-v4lutils \
18                         packagegroup-petalinux-display-debug \
19                         packagegroup-petalinux-x11 \
20                         packagegroup-petalinux-opencv \
21                         packagegroup-petalinux-gstreamer \
22                         packagegroup-petalinux-qt \
23                         packagegroup-petalinux-qt-extended \
24                         packagegroup-core-tools-debug \
25                         ffmpeg \
26                         file \
27                         ldd \
28                         xrt \
29                         zocl \
30                         opencl-clhpp-dev \
31                         opencl-headers-dev \
32                         vai-staticdev \
33                         xrtutils \
34                         "
35 inherit populate_sdk_qt5
36 TOOLCHAIN_HOST_TASK += "nativesdk-qtbase-dev"
37 TOOLCHAIN_TARGET_TASK += "kernel-devsrc"

```

그림 23 petalinux-user-image.bbappend

## 9. Petalinux Image Build

다음의 명령을 통해 지금까지 Configuration한 Device Tree, u-boot, Kernel등의 BSP및 root filesystem을 포함한 Linux System을 위한 모든 것을 Build 한다.

```

$ cd ~/work/zynqmp_linux/petalinux/ultra96
$ petalinux-build

```

Petalinux Project(ultra96) 폴더아래의 images/linux에서 zynqmp\_fsble.elf, pmufw.elf, system.bit, bl31.elf, u-boot.elf가 생성되었음을 확인한다. 다음의 명령을 사용하여 이전 파일들로 구성된 BOOT.BIN 파일을 만든다.

```
$ petalinux-package --force --boot --fsbl images/linux/zynqmp_fsbl.elf --u-boot
images/linux/u-boot.elf --pmufw images/linux/pmufw.elf --fpga
images/linux/system.bit
```

BOOT.BIN 파일이 images/linux 아래에 생성되었음을 확인한다.

## 10. Sdcard Preparation

sdcard를 host machine의 sdcard 슬롯에 꽂고 다음의 명령들을 수행한다.

(parted) 다음의 명령어들을 입력하여 sdcard에 2개의 partition들(boot partition, linux root filesystem partition)을 만든다.

```
$ sudo parted /dev/mmcblk0
```

```
GNU Parted 3.2
Using /dev/mmcblk0
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Model: SD SL16G (sd/mmc)
Disk /dev/mmcblk0: 15.9GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start  End  Size  Type  File system  Flags
(parted) mkpart primary fat32 0 200MB
Warning: The resulting partition is not properly aligned for best performance.
Ignore/Cancel? I
(parted) mkpart primary ext4 200MB 100%
(parted) print
Model: SD SL16G (sd/mmc)
Disk /dev/mmcblk0: 15.9GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start  End  Size  Type  File system  Flags
1       512B   200MB 200MB  primary  fat32        lba
2       200MB  15.9GB 15.7GB  primary  ext4         lba
(parted) quit
Information: You may need to update /etc/fstab.
```

그림 24 sdcard partitions

sdcard를 슬롯에서 빼서 다시 꽂고 다음의 명령으로 partition들을 format 한다.

```
$ sudo mkfs.vat -n card /dev/mmcblk0p1
$ sudo mkfs.ext4 -L root /dev/mmcblk0p2
```

sdcard를 슬롯에서 빼서 다시 꽂으면 boot partition은 /media/hokim/sdcard, root filesystem partition은 /media/hokim/root 로 mount 된다. 여기서 hokim은 사용자의 id에 따라 다르다.

다음의 명령으로 sdcard의 각 partition들에 Step 10에서 생성된 결과물들을 Write 한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96
$ cp images/linux/{BOOT.BIN,image.ub} /media/hokim/card/
$ sudo tar xvzf images/linux/rootfs.tar.gz -C /media/hokim/root/
$ sync
```

## 11. Test

다음의 명령어들로 host에서 보드의 uart에 연결할 program을 준비한다.

```
$ mkdir ~/bin
$ cp ~/work/zynqmp_linux/utils/miniterm.py ~/bin/
$ chmod +x ~/bin/miniterm.py
$ echo "export PATH=~$HOME/bin:$PATH" >> ~/.bashrc
$ sudo usermod -a -G dialout hokim
$ sudo apt install python-serial
```

위의 hokim은 사용자의 id를 사용하며, 명령어들을 실행한 후 host를 재부팅한다.

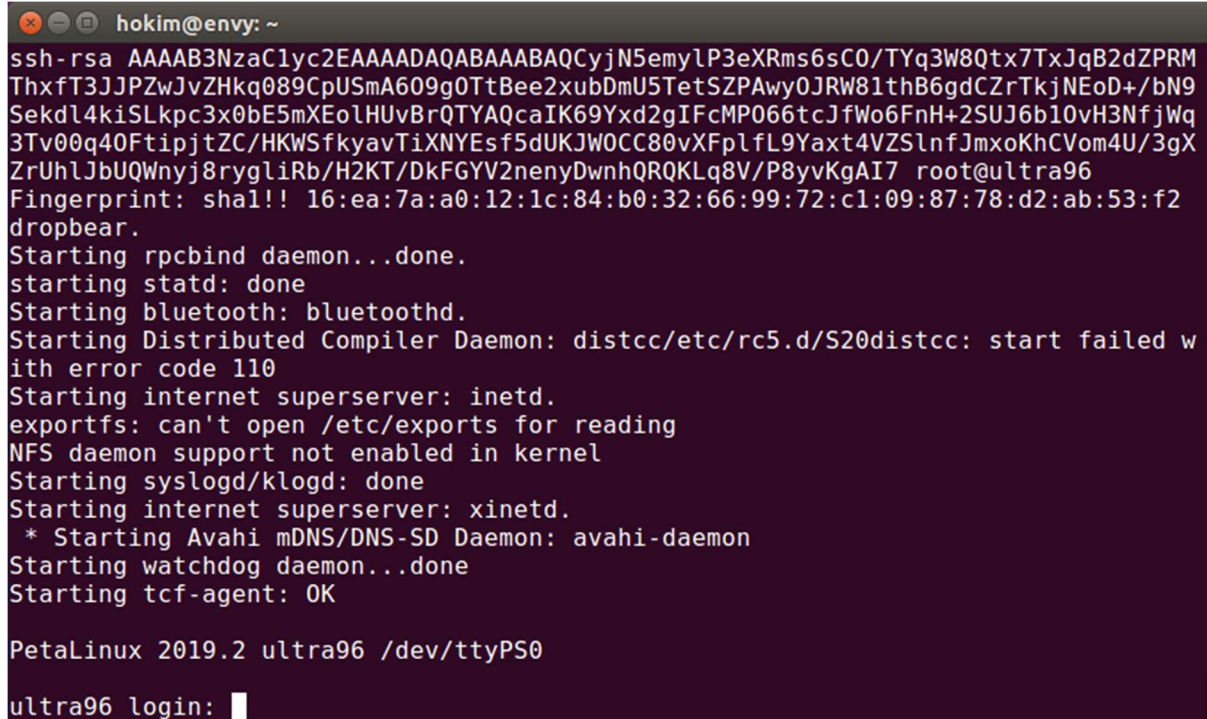
Step 10에서 만든 sdcard를 보드에 꽂고 USB-to-Uart를 보드에 결합하고 usb선으로 host와 연결한다.

host에서 다음의 명령으로 uart연결을 시도한다.



```
$ miniterm.py -p /dev/ttyUSB1
```

보드의 전원을 공급하고 power switch를 누르면 다음과 같은 화면이 나와야 한다.



```
hokim@envy: ~
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCyjN5emylP3eXRms6sC0/TYq3W8QtX7TxJqB2dZPRM
ThxfT3JJpZwJvZHKq089CpUSmA609g0TtBee2xubDmU5TetSZPAwy0JRW81thB6gdCZrTkjNEoD+/bN9
Sekdl4kiSLkpc3x0bE5mXEolHUvBrQTYAQcaIK69Yxd2gIFcMP066tcJfWo6FnH+2SUJ6b10vH3NfjWq
3Tv00q40FtipjtZC/HKWSfkyavTiXNYEs5dUKJW0CC80vXFplfL9Yaxt4VZSlnfJmxoKhCVom4U/3gX
ZrUhlJbUQWnyj8rygliRb/H2KT/DkFGYV2nenyDwnhQRQKLq8V/P8yvKgAI7 root@ultra96
Fingerprint: sha1!! 16:ea:7a:a0:12:1c:84:b0:32:66:99:72:c1:09:87:78:d2:ab:53:f2
dropbear.
Starting rpcbind daemon...done.
starting statd: done
Starting bluetooth: bluetoothd.
Starting Distributed Compiler Daemon: distcc/etc/rc5.d/S20distcc: start failed w
ith error code 110
Starting internet superserver: inetd.
exportfs: can't open /etc/exports for reading
NFS daemon support not enabled in kernel
Starting syslogd/klogd: done
Starting internet superserver: xinetd.
* Starting Avahi mDNS/DNS-SD Daemon: avahi-daemon
Starting watchdog daemon...done
Starting tcf-agent: OK

PetaLinux 2019.2 ultra96 /dev/ttyPS0
ultra96 login: █
```

그림 25 Ultra96 boot screen

그림23의 root 비밀번호를 사용하여 root로 login해서 보드의 ip를 다음과 같이 알아낸다.



```
hokim@envy: ~  
PetaLinux 2019.2 ultra96 /dev/ttyPS0  
ultra96 login: root  
Password:  
root@ultra96:~$ ifconfig  
lo                Link encap:Local Loopback  
                  inet addr:127.0.0.1  Mask:255.0.0.0  
                  inet6 addr: ::1/128 Scope:Host  
                  UP LOOPBACK RUNNING  MTU:65536  Metric:1  
                  RX packets:2 errors:0 dropped:0 overruns:0 frame:0  
                  TX packets:2 errors:0 dropped:0 overruns:0 carrier:0  
                  collisions:0 txqueuelen:1000  
                  RX bytes:140 (140.0 B)  TX bytes:140 (140.0 B)  
  
wlan0             Link encap:Ethernet  HWaddr F8:F0:05:C3:33:96  
                  inet addr:172.30.1.39  Bcast:172.30.1.255  Mask:255.255.255.0  
                  UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
                  RX packets:5 errors:0 dropped:0 overruns:0 frame:0  
                  TX packets:25 errors:0 dropped:0 overruns:0 carrier:0  
                  collisions:0 txqueuelen:1000  
                  RX bytes:1570 (1.5 KiB)  TX bytes:4321 (4.2 KiB)  
  
root@ultra96:~$  
root@ultra96:~$
```

그림 26 Ultra96 board ip address

wlan0 inet addr의 172.30.1.39가 보드의 ip 주소이고 다음의 명령어를 통해 wifi network를 통해 보드로 연결한다.

```
$ ssh root@172.30.1.39
```

## 12. SDK Build

다음의 명령어들을 사용하여 SDK를 Build하고 그 결과를 ~/work/zynqmp\_linux/petalinux로 옮긴다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96  
$ petalinux-build -s  
$ mv images/linux/sdk.sh ..
```

## 13. Petalinux BSP

다음의 명령어들을 사용하여 Petalinux BSP를 만든다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96
$ mkdir pre-built
$ cp images/linux/BOOT.BIN pre-built/
$ cp images/linux/image.ub pre-built/
$ cp images/linux/rootfs.tar.gz pre-built/
$ petalinux-build -x mrproper
$ cd ..
```

Ultra96v1 보드:

```
$ petalinux-package --bsp -p ultra96 --output ultra96v1-2019.2.bsp
```

Ultra96v2 보드:

```
$ petalinux-package --bsp -p ultra96 --output ultra96v2-2019.2.bsp
```

Petalinux BSP를 이용한 Petalinux Project 생성은 다음과 같이 한다.

Ultra96v1 보드:

```
$ rm -fr ultra96
$ petalinux-create -t project -s ultra96v1-2019.2.bsp
```

Ultra96v2 보드:

```
$ rm -fr ultra96
$ petalinux-create -t project -s ultra96v2-2019.2.bsp
```

ultra96 폴더가 생성되었음을 확인한다.