

# **Embedded Linux on Zynq UltraScale+ MPSoC Labs**

**Tool : Xilinx Vivado & Petalinux 2019.2**

**Kit : Ultra96 Training Kit**

# Labs

- **Lab1. Petalinux Linux System Build**
- Lab2. Linux Application Build Flow
- Lab3. HW Linux Application Build
- Lab4. Custom HW Linux Application Build
- Lab5. Linux Xilinx Video Pipeline

## Lab1. Petalinux Linux System Build Overview

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

## Step 1 Source

- ❖ 아래 명령들을 통해 Github에 있는 Embedded Linux on Zynq UltraScale+ MPSoC 교육 자료를 복사한다.

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

## Step 2 Create Vivado Project

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

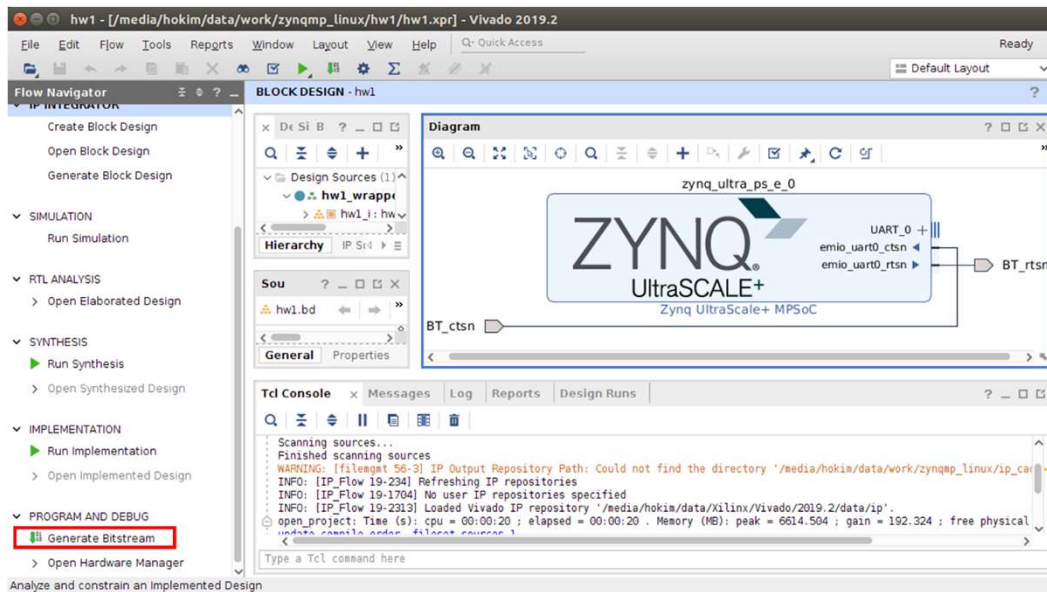
❖ Create Vivado Project for Ultra96 v1 Board

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

❖ Create Vivado Project for Ultra96 v2 Board

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

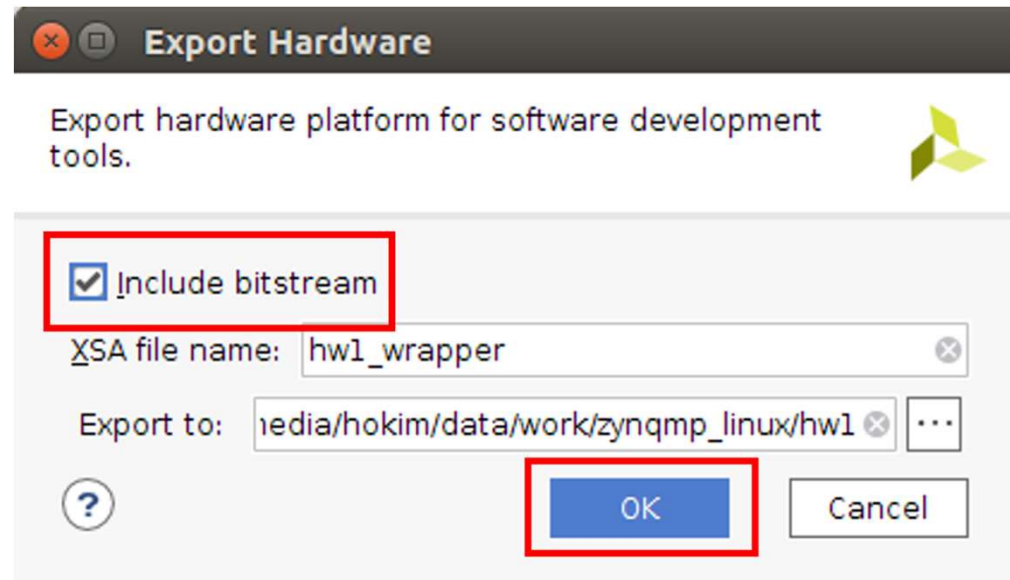
## Step 2 Generate Bitstream



❖ Flow Navigator ⇒ PROGRAM  
AND DEBUG ⇒ Generate  
Bitstream을 클릭하여  
Bitstream을 생성한다.

## Step 2 Export Hardware

❖ File ⇒ Export ⇒ Export Hardware를 클릭한다.



## Step 3 Create Petalinux Project

❖ 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/
```



## Step 3 Petalinux Configuration 1

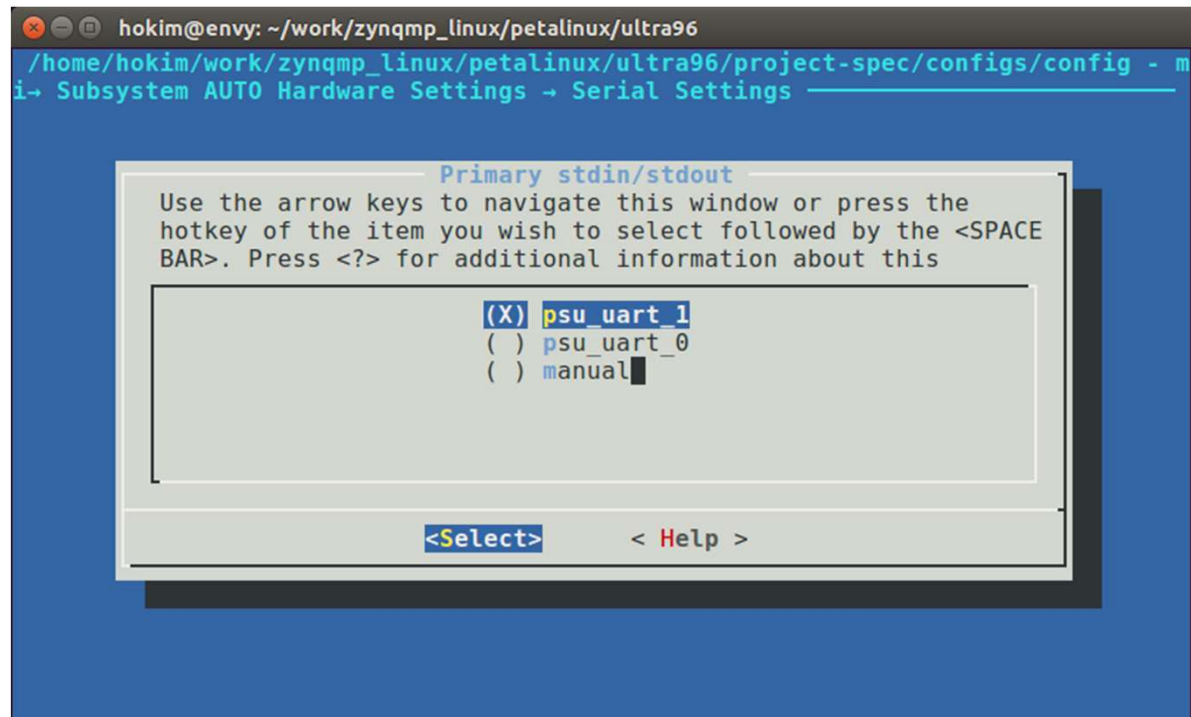
❖ Petalinux Configuration 화면  
이 나타난다.

```
hokim@envy: ~/work/zynqmp_linux/petalinux/ultra96
/home/hokim/work/zynqmp_linux/petalinux/ultra96/project-spec/configs/config - m
i
misc/config System Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]

-* ZYNQMP Configuration
  Linux Components Selection --->
  Auto Config Settings --->
  -* Subsystem AUTO Hardware Settings --->
    DTG Settings --->
    ARM Trusted Firmware Compilation Configuration --->
    [ ] Power Management kernel configuration (NEW)
    FPGA Manager --->
    u-boot Configuration --->
    Image Packaging Configuration --->
  i(+)

<Select> <Exit> <Help> <Save> <Load>
```

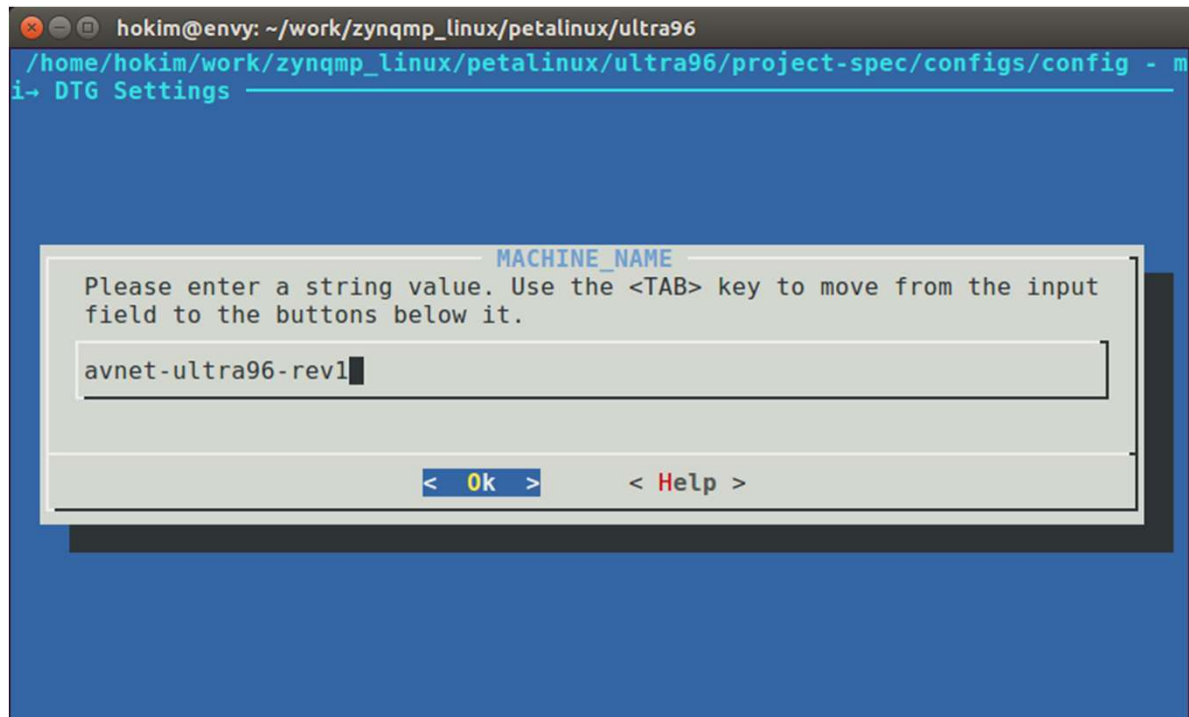
## Step 3 Petalinux Configuration 2 (Primary stdin/stdout)



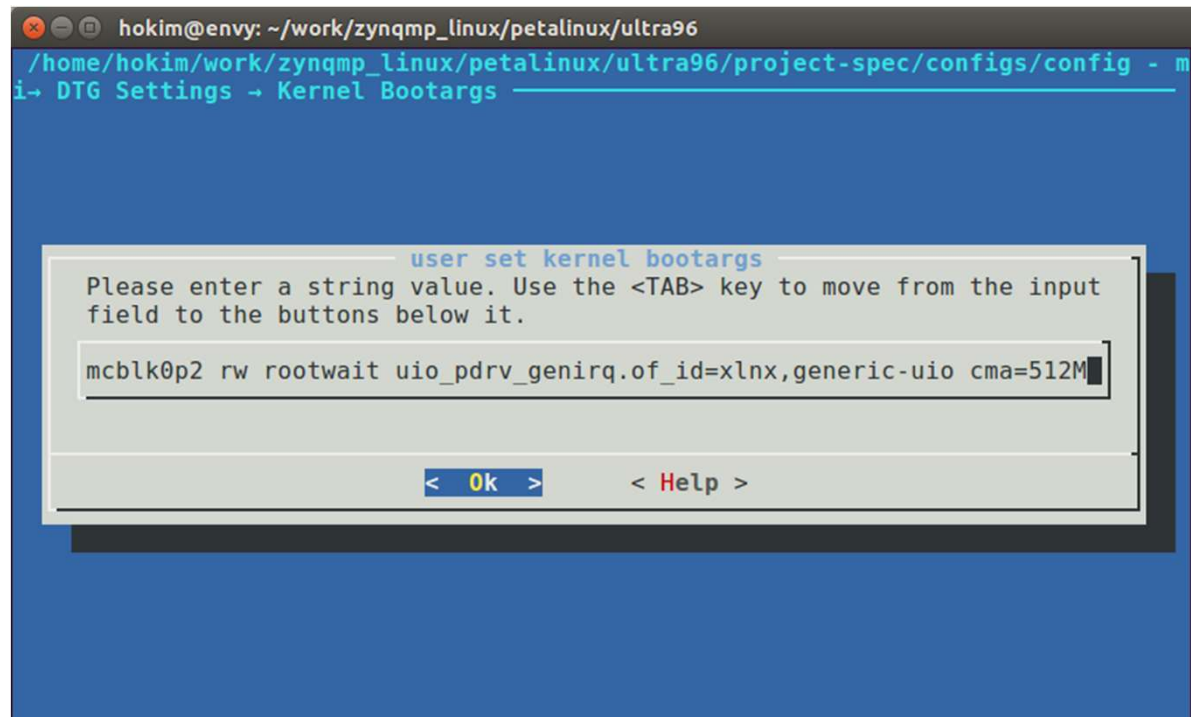
❖ Subsystem AUTO Hardware  
Setting ⇒ Serial Settings ⇒  
Primary stdin/stdout 에서  
uart1을 선택한다.

## Step 3 Petalinux Configuration 3 (DTG MACHINE\_NAME)

- ❖ DTG Settings ⇒  
MACHINE\_NAME에 avnet-ultra96-rev1을 입력한다.



## Step 3 Petalinux Configuration 4 (Kernel Bootargs)



The screenshot shows a terminal window with the Petalinux configuration interface. The title bar indicates the user is hokim@envy and the current directory is ~/work/zynqmp\_linux/petalinux/ultra96. The command prompt shows the user is in the directory /home/hokim/work/zynqmp\_linux/petalinux/ultra96/project-spec/configs/config - m. The user has navigated to 'DTG Settings' and then 'Kernel Bootargs'. A dialog box titled 'user set kernel bootargs' is open, prompting the user to enter a string value. The input field contains the text 'mcbblk0p2 rw rootwait uio\_pdrv\_genirq.of\_id=xlnx,generic-uio cma=512M'. Below the input field are two buttons: '< Ok >' and '< Help >'.

```
hokim@envy: ~/work/zynqmp_linux/petalinux/ultra96
/home/hokim/work/zynqmp_linux/petalinux/ultra96/project-spec/configs/config - m
i→ DTG Settings → Kernel Bootargs

user set kernel bootargs
Please enter a string value. Use the <TAB> key to move from the input
field to the buttons below it.
mcbblk0p2 rw rootwait uio_pdrv_genirq.of_id=xlnx,generic-uio cma=512M
< Ok > < Help >
```

❖ DTG Settings ⇒ Kernel

Bootargs 에서 generate boot args automatically를 선택해제 하고 user set kernel bootargs 에 "earlycon

console=ttyPS0,115200

clk\_ignore\_unused

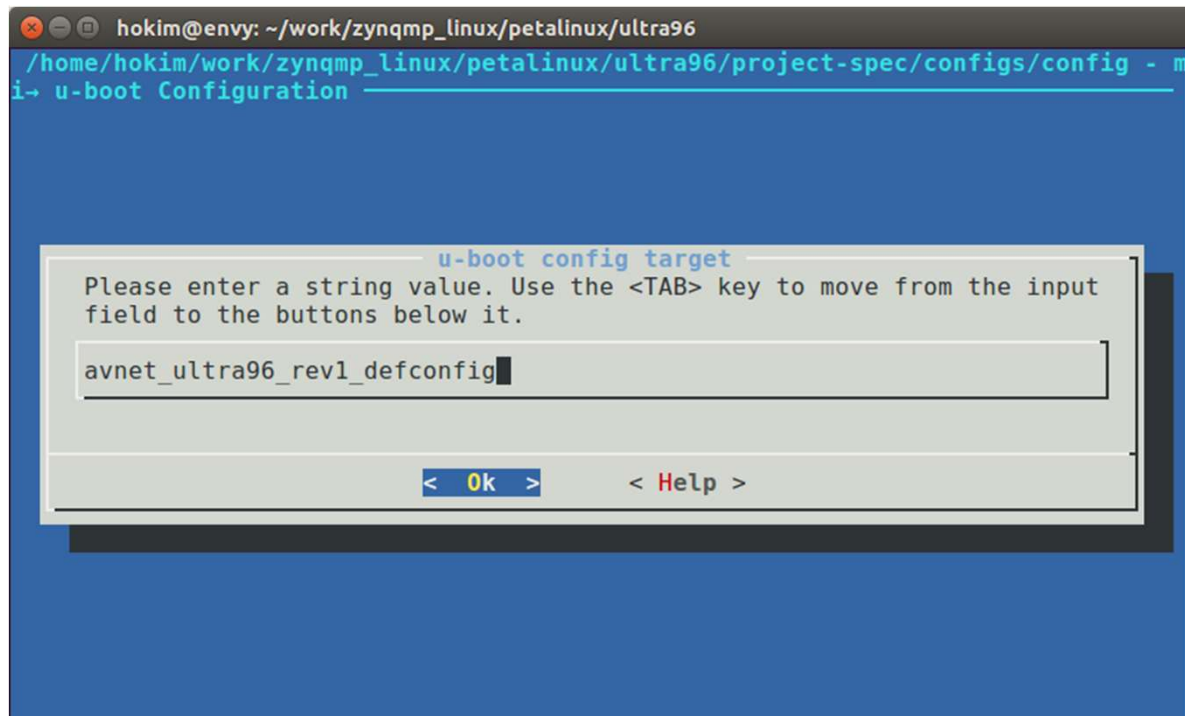
root=/dev/mmcblk0p2 rw

rootwait

uio\_pdrv\_genirq.of\_id=xlnx,generic-uio cma=512M"를 입력한다.

## Step 3 Petalinux Configuration 5 (u-boot config target)

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



```
hokim@envy: ~/work/zynqmp_linux/petalinux/ultra96
/home/hokim/work/zynqmp_linux/petalinux/ultra96/project-spec/configs/config - m
i→ u-boot Configuration
```

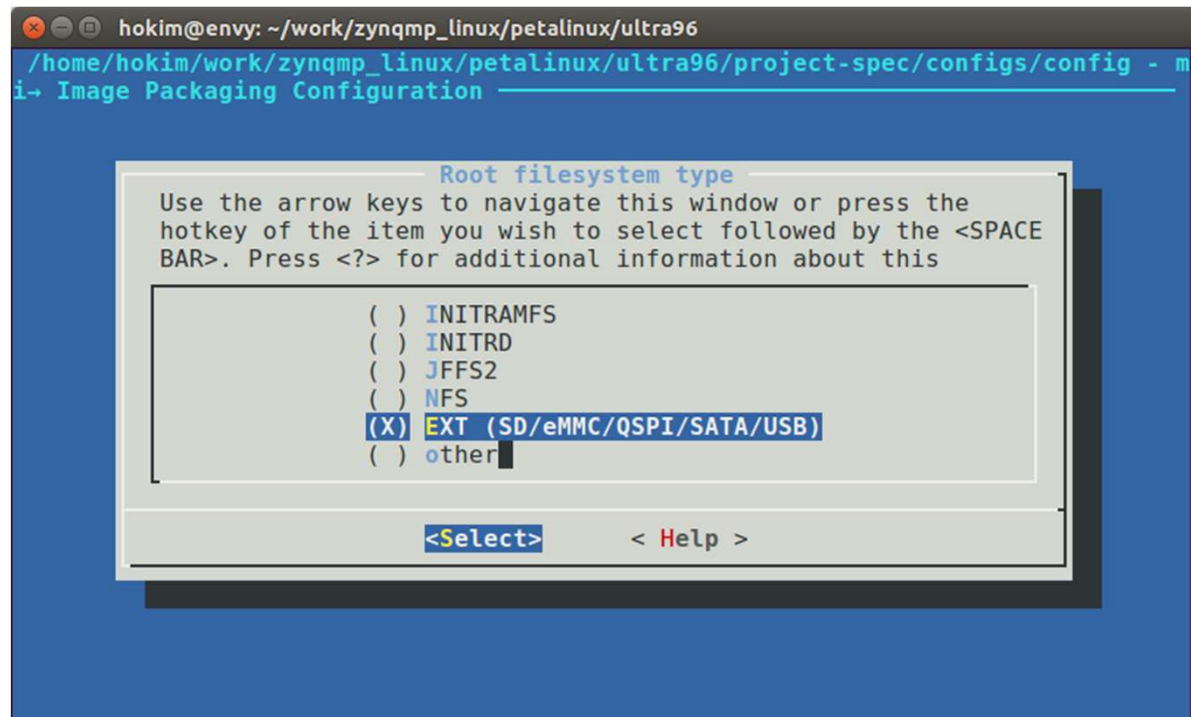
u-boot config target

Please enter a string value. Use the <TAB> key to move from the input field to the buttons below it.

avnet\_ultra96\_rev1\_defconfig

< Ok > < Help >

## Step 3 Petalinux Configuration 6 (Root filesystem type)



### ❖ Image Packaging

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

## Step 3 Petalinux Configuration 7 (disable tftpboot copy)

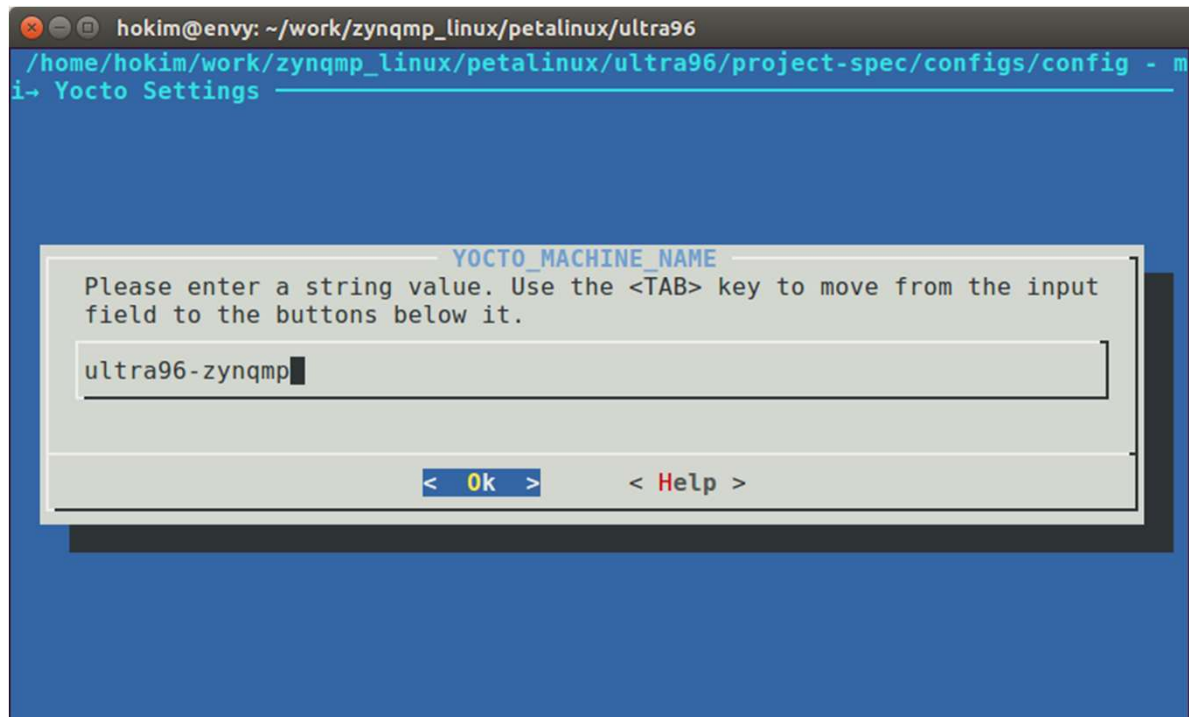
```
hokim@envy: ~/work/zynqmp_linux/petalinux/ultra96
/home/hokim/work/zynqmp_linux/petalinux/ultra96/project-spec/configs/config - m
i- Image Packaging Configuration
Image Packaging Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]
Root filesystem type (EXT (SD/eMMC/QSPI/SATA/USB)) --->
(/dev/mmcblk0p2) Device node of SD device (NEW)
(image.ub) name for bootable kernel image
(cpio cpio.gz cpio.gz.u-boot tar.gz jffs2) Root filesystem format
(0x1000) DTB padding size
[ ] Copy final images to tftpboot
<Select> < Exit > < Help > < Save > < Load >
```

### ❖ Image Packaging

Configuration에서 Copy final images to tftpboot를 선택해제 한다.

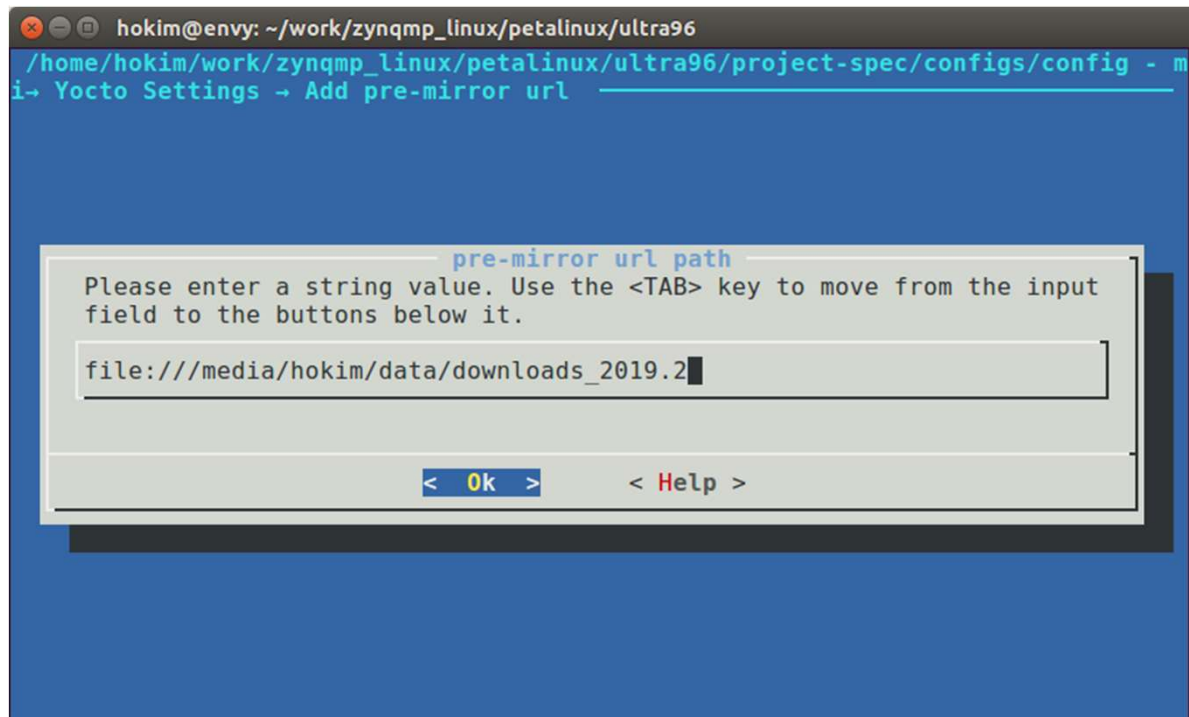
## Step 3 Petalinux Configuration 8(YOCTO\_MACHINE\_NAME)

- ❖ Yocto Settings ⇒  
YOCTO\_MACHINE\_NAME에  
ultra96-zynqmp를 입력한다.



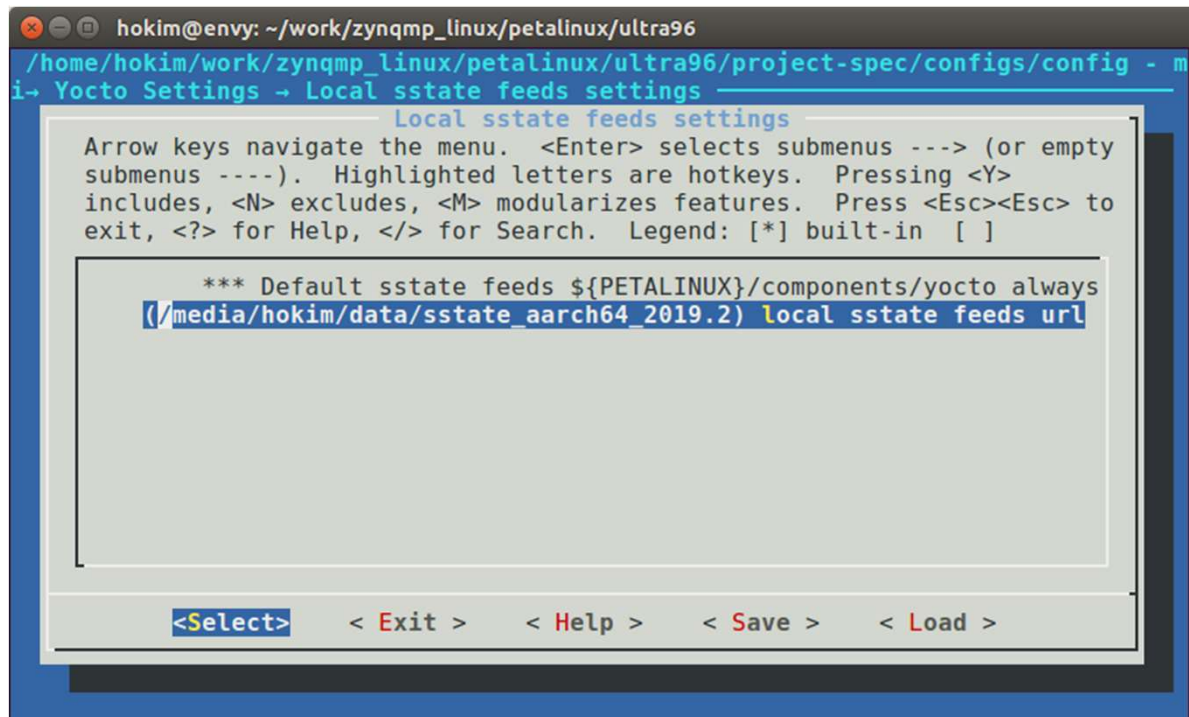


## Step 3 Petalinux Configuration 9 (pre-mirror url path)



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

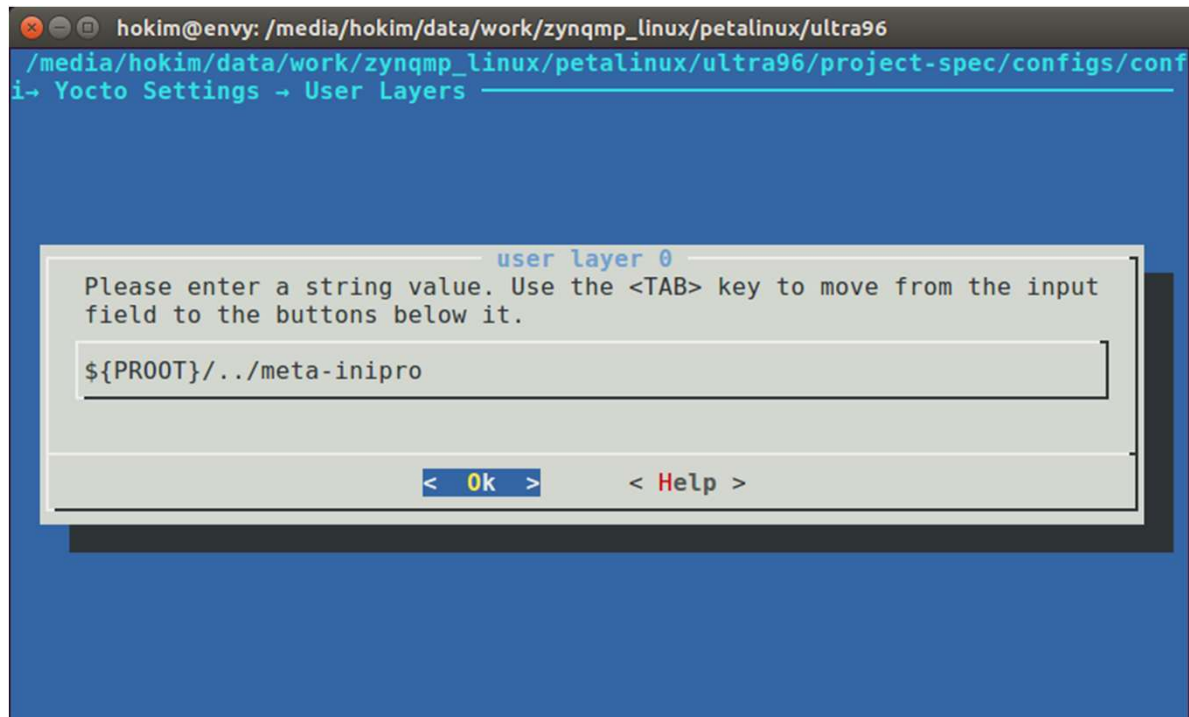
## Step 3 Petalinux Configuration 10 (Local sstate feeds)



```
hokim@envy: ~/work/zynqmp_linux/petalinux/ultra96
/home/hokim/work/zynqmp_linux/petalinux/ultra96/project-spec/configs/config - m
i→ Yocto Settings → Local sstate feeds settings
Local sstate feeds settings
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]
*** Default sstate feeds ${PETALINUX}/components/yocto always
(/media/hokim/data/sstate_aarch64_2019.2) local sstate feeds url
<Select> < Exit > < Help > < Save > < Load >
```

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

## Step 3 Petalinux Configuration 11 (User Layers)



- ❖ Yocto Settings ⇒ User Layers  
에 \${PROOT}/../meta-inipro를  
입력한다.
- ❖ 설정을 마치고 종료한다.

## Step 4 Petalinux BSP Configuration for Ultra96v1

```
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 "
```

- ❖ 편집기(vi, gedit, ...)를 사용하여 Petalinux Project(ultra96) 폴더 아래의 project-spec/meta-user/conf/petalinuxbsp.conf에 그림처럼 line17부터의 내용을 추가한다.
- ❖ SSTATE\_MIRRORS\_append의 폴더경로는 사용자의 환경에 맞게 file:///media/hokim/data/sstate\_aarch64\_2019.2\_2부분을 수정한다.

## Step 4 Petalinux BSP Configuration for Ultra96v2

```
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 "
```

- ❖ 편집기(vi, gedit, ...)를 사용하여 Petalinux Project(ultra96) 폴더 아래의 project-spec/meta-user/conf/petalinuxbsp.conf에 그림처럼 line17부터의 내용을 추가한다.
- ❖ SSTATE\_MIRRORS\_append의 폴더경로는 사용자의 환경에 맞게 file:///media/hokim/data/sstate\_aarch64\_2019.2\_2부분을 수정한다.

## Step 5 Device Tree Generation

❖ 다음의 명령을 수행하여 xsa 파일과 Step 3 Petalinux Configuration 3 (DTG MACHINE NAME)에 기초한 Device Tree를 Generation 한다.

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

## Step 5 Device Tree Configuration

```
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 };
```

- ❖ 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 열어 Device Tree를 수정한다.
- ❖ Ultra96v1 보드는 avnet-ultra96-rev1.dtsi에 보드에 맞는 Device Tree 정보가 들어있어서 수정할 내용은 없고 Ultra96v2 보드를 위해서는 그림과 같이 수정해야 한다.



## Step 6 U-Boot Configuration

```
hokim@envy: /media/hokim/data/work/zynqmp_linux/petalinux/ultra96
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     {{I2C_MUX_PCA9548, 0x75, 0} }, \
32     {{I2C_MUX_PCA9548, 0x75, 1} }, \
33     {{I2C_MUX_PCA9548, 0x75, 2} }, \
34     {{I2C_MUX_PCA9548, 0x75, 3} }, \
35     {{I2C_MUX_PCA9548, 0x75, 4} }, \
36     {{I2C_MUX_PCA9548, 0x75, 5} }, \
37     {{I2C_MUX_PCA9548, 0x75, 6} }, \
38     {{I2C_MUX_PCA9548, 0x75, 7} }, \
39 }
```

- ❖ Petalinux Configuration 5 (u-boot config target)의 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 그림과 같이 line26부터의 내용을 추가한다.

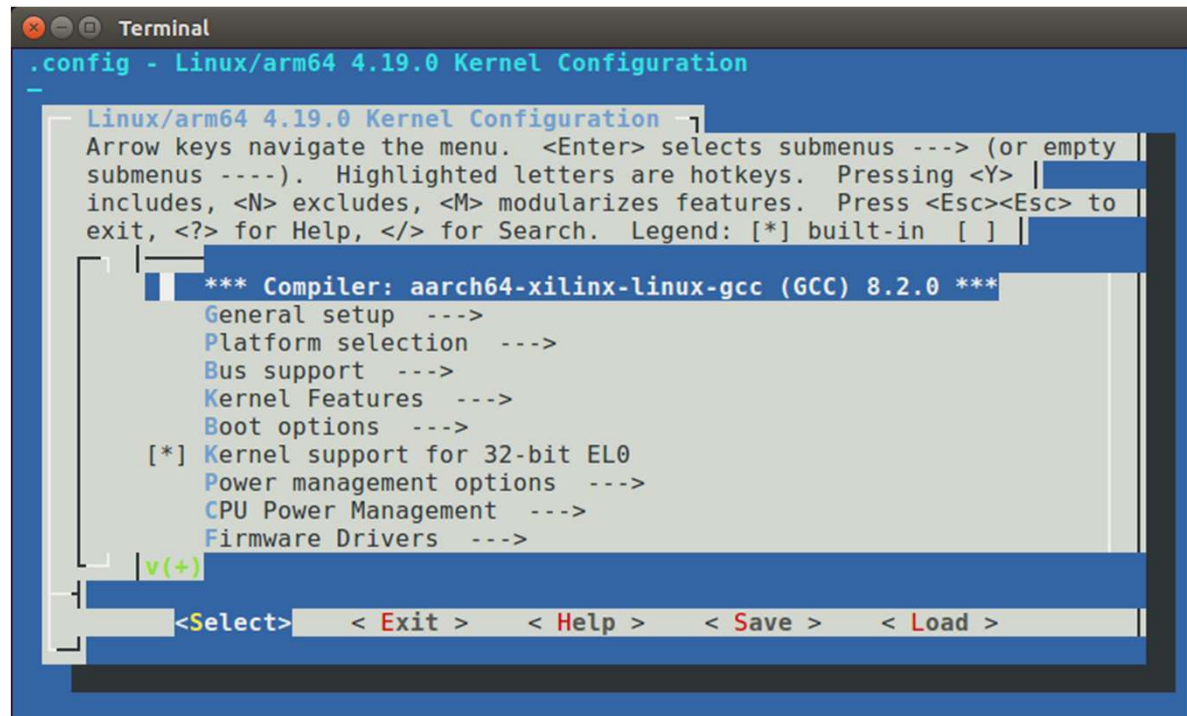


## Step 7 Kernel Configuration 1

❖ Kernel Configuration을 위해 다음의 명령을 수행한다.

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

## Step 7 Kernel Configuration 2



```
Terminal
.config - Linux/arm64 4.19.0 Kernel Configuration

Linux/arm64 4.19.0 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]

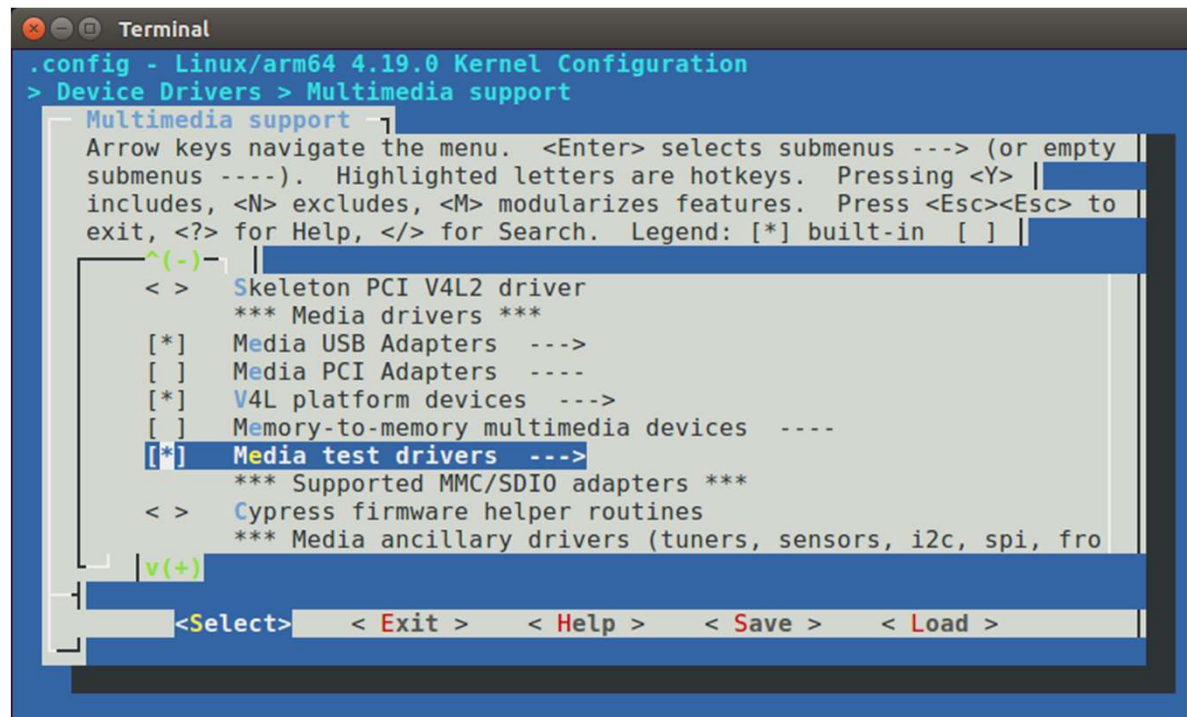
*** Compiler: aarch64-xilinx-linux-gcc (GCC) 8.2.0 ***
General setup --->
Platform selection --->
Bus support --->
Kernel Features --->
Boot options --->
[*] Kernel support for 32-bit EL0
Power management options --->
CPU Power Management --->
Firmware Drivers --->

v(+)
<Select> <Exit> <Help> <Save> <Load>
```

❖그림처럼 Kernel

Configuration 화면이 나오면  
Device Drivers를 선택한다.

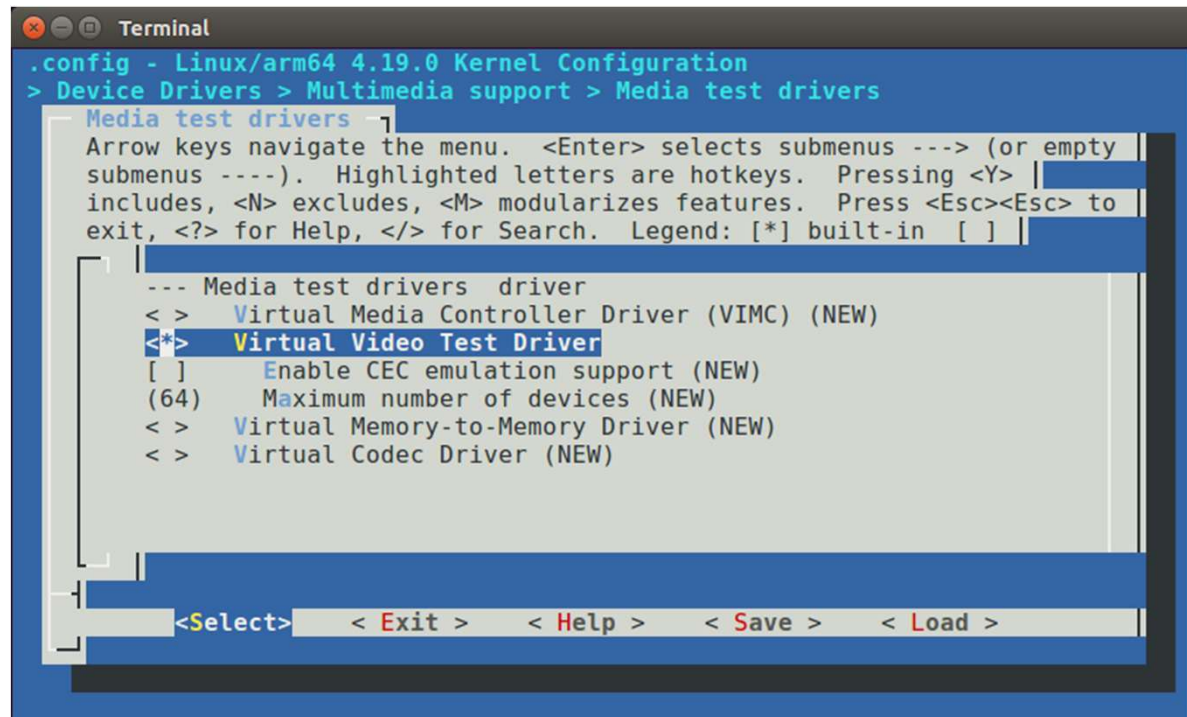
## Step 7 Kernel Configuration 3



```
Terminal
.config - Linux/arm64 4.19.0 Kernel Configuration
> Device Drivers > Multimedia support
Multimedia support
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]
^(-)
< > Skeleton PCI V4L2 driver
*** Media drivers ***
[*] Media USB Adapters --->
[ ] Media PCI Adapters ----
[*] V4L platform devices --->
[ ] Memory-to-memory multimedia devices ----
[*] Media test drivers --->
*** Supported MMC/SDIO adapters ***
< > Cypress firmware helper routines
*** Media ancillary drivers (tuners, sensors, i2c, spi, fro
v(+)
<Select> < Exit > < Help > < Save > < Load >
```

- ❖그림처럼 space bar를 사용하여 Device Drivers ⇒ Multimedia support ⇒ Media test drivers 를 선택한다.

## Step 7 Kernel Configuration 4



```
Terminal
.config - Linux/arm64 4.19.0 Kernel Configuration
> Device Drivers > Multimedia support > Media test drivers
Media test drivers
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]
--- Media test drivers driver
< > Virtual Media Controller Driver (VIMC) (NEW)
[*] Virtual Video Test Driver
[ ] Enable CEC emulation support (NEW)
(64) Maximum number of devices (NEW)
< > Virtual Memory-to-Memory Driver (NEW)
< > Virtual Codec Driver (NEW)

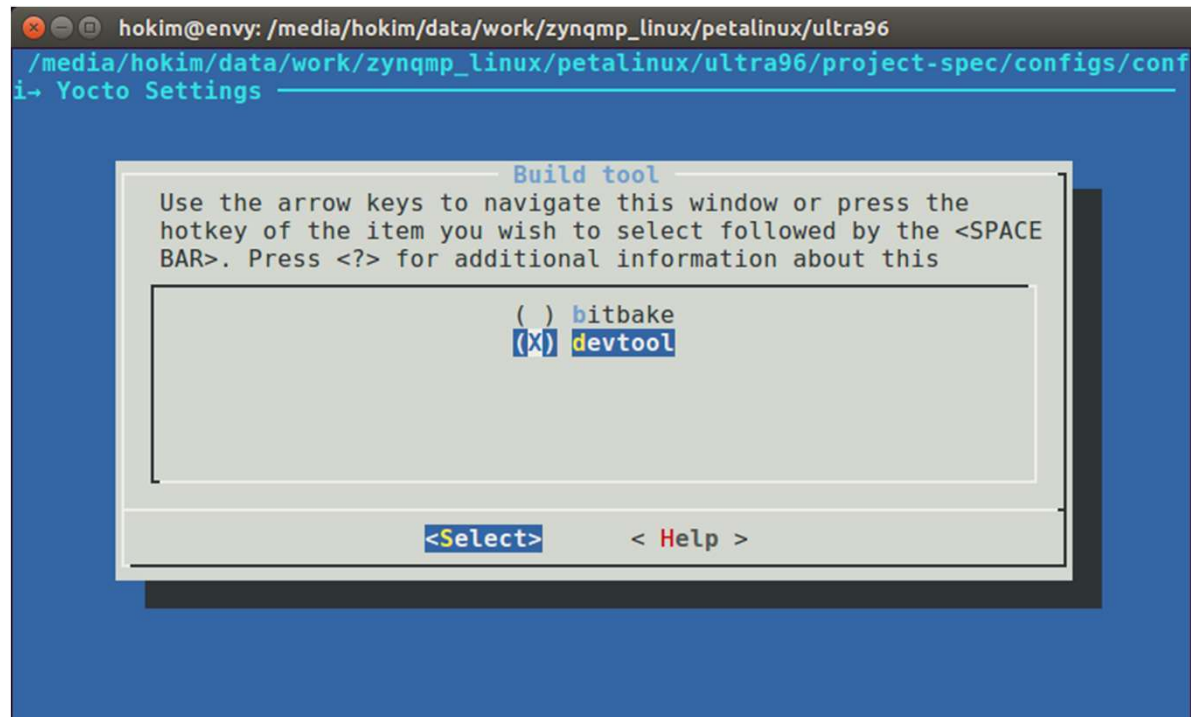
<Select> < Exit > < Help > < Save > < Load >
```

- ❖그림처럼 space bar를 사용하여 Device Drivers ⇒ Multimedia support ⇒ Media test drivers ⇒ Virtual Video Test Driver 를 선택한다.

## Step 7 Kernel Configuration 5

- ❖ 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 수행해야한다.

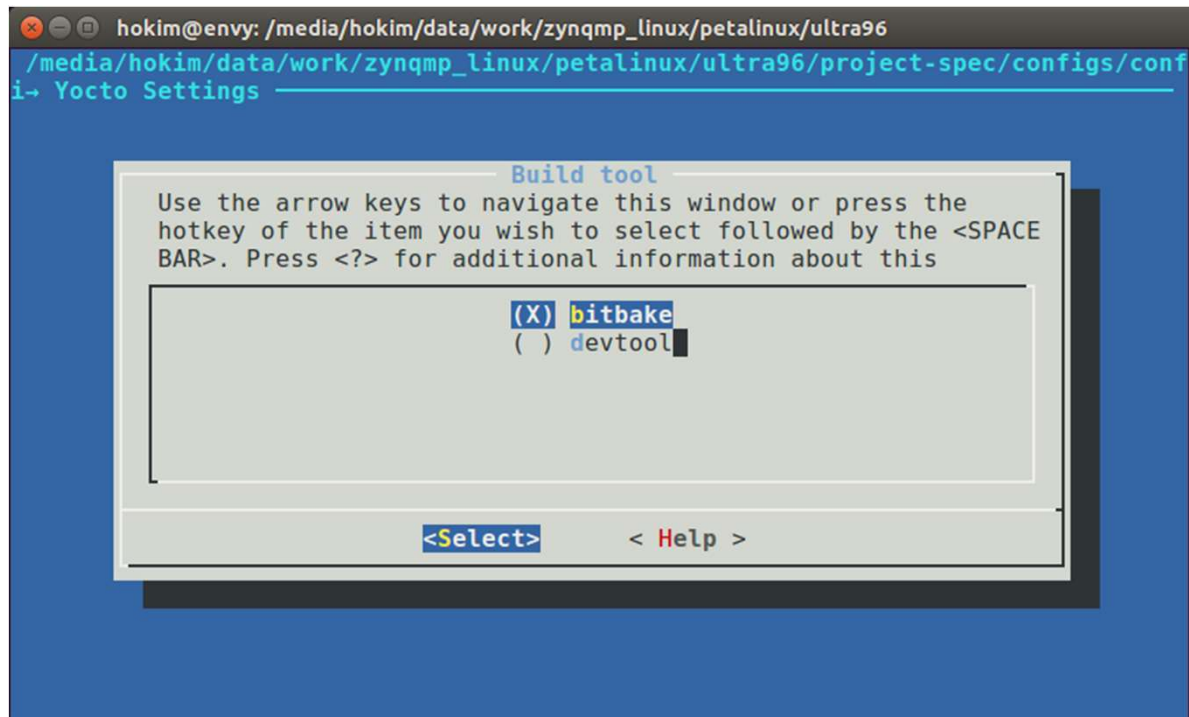
## Step 7 Kernel Configuration 6



❖아래와 같은 명령를 사용하여  
Petalinux Configuration화면이  
나오면 Yocto Settings ⇒  
Build tool에서 devtool을 선택  
하고 종료한다.

```
$ petalinux-config
```

## Step 7 Kernel Configuration 7



- ❖ 아래 명령에 의해 생성된 project-spec/meta-user/recipes-kernel/linux/linux-xlnx 폴더 아래의 linux-xlnx\_2019.2.bbappend와 linux-xlnx/devtool-fragment.cfg 파일들을 확인한다.

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

- ❖ 아래 명령을 사용하여 Petalinux Configuration 화면을 다시 연 후 그림처럼 Yocto Settings ⇒ Build tool 에서 devtool을 선택하고 종료한다.

```
$ petalinux-config
```

## Step 7 Kernel Configuration 8

❖ 다음의 명령은 Configuration을 위해 사용했던 Kernel Source를 cleanup 한다.

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



## Step 8 Image Configuration

```
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-supPLICANT \
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"
```

❖ Petalinux Project(ultra96)폴더  
아래에서 다음의 명령을 사용  
하여 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
```

## Step 9 Petalinux Image Build

- ❖ 다음의 명령을 통해 지금까지 Configuration한 Device Tree, u-boot, Kernel등의 BSP 및 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\_fsbl.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.

## Step 10 Sdcard Preparation 1

```
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.
```

- ❖ sdcard를 host machine의 sdcard 슬롯에 꽂고 다음의 명령을 수행한다.

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

- ❖ (parted) 그림과 같은 명령어들을 입력하여 sdcard에 2개의 partition(boot partition, linux root filesystem partition)을 만든다.

## Step 10 Sdcard Preparation 2

❖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 Petalinux Image Build에서 생성된 결과물들을 Write한다.

```
$ 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
```

## Step 11 Test 1

❖ 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=W$HOME/bin:W$PATH" >> ~/.bashrc
$ sudo usermod -a -G dialout hokim
$ sudo apt install python-serial
```

❖ 위의 hokim은 사용자의 id를 사용하며, 명령어들을 실행한 후 host를 재부팅한다. 다음의 명령으로 sdcard의 각 partition들에 Petalinux Image Build에서 생성된 결과물들을 Write 한다.

❖ Sdcard Preparation에서 만든 sdcard를 보드에 꽂고 USB-to-Uart를 보드에 결합하고 usb 선으로 host와 연결한다.

## Step 11 Test 2

```
hokim@envy: ~  
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCyjN5emylP3eXRms6sC0/TYq3W8Qtx7TxJqB2dZPRM  
ThxftT3JJpZwJvZHKq089CpUSmA609g0TtBee2xubDmU5TetSZPAwy0JRW81thB6gdCZrTkjNEoD+/bN9  
Sekdl4kiSLkpc3x0bE5mXEolHUvBrQTYAQcaIK69Yxd2gIFcMP066tcJfWo6FnH+2SUJ6b10vH3NfjWq  
3Tv00q40FtipjtZC/HKWSfkyavTiXNYEsf5dUKJW0CC80vXFplfL9Yaxt4VZSlmfJmxoKhCVom4U/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: █
```

❖ host에서 다음의 명령으로  
uart연결을 시도한다.

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

❖ 보드의 전원을 공급하고  
power switch를 누르면 다음  
과 같은 화면이 나와야 한다.



## Step 11 Test 3

```
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:~$
```

❖ Image Configuration에서 설정한 root 비밀번호를 사용하여 root로 login한 후 그림처럼 보드의 ip를 알아낸다.

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

```
$ ssh root@172.30.1.39
```

## Step 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 ..
```



## Step 13 Create 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 Board

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

❖ Ultra96v2 Board

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

## Step 13 Create Petalinux Project using Petalinux BSP

### ❖ Ultra96v1 Board

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

### ❖ Ultra96v2 Board

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

❖ ultra96 폴더가 생성되었음을 확인한다.

# Labs

- Lab1. Petalinux Linux System Build
- **Lab2. Linux Application Build Flow**
- Lab3. HW Linux Application Build
- Lab4. Custom HW Linux Application Build
- Lab5. Linux Xilinx Video Pipeline

## Lab2. Linux Application Build Flow Overview

- ❖ Lab1에서 만든 SDK를 사용하여 Linux Application을 host에서 cross compile하고 보드에 올려서 테스트한다.
- ❖ Lab1의 Image Configuration에서 IMAGE\_INSTALL\_append에 추가될 수 있는 Linux Application을 packaging 하는 recipe들을 만들고 package management system을 사용하여 보드에 설치 테스트한다.

## Step 1 HW Preparation

- ❖ 보드에 Usb-to-Uart와 Mini DP to HDMI Adapter를 연결하고 각각 host의 usb 포트와 Monitor에 연결한다.

## Step 2 SDK Installation

❖ SDK script을 실행하여 SDK를 설치한다.

```
$ cd ~/work/zynqmp_linux/petalinux  
$ ./sdk.sh -y -d ~/sdk
```

## Step 3 SDK Application Build

❖ SDK를 이용하여 2개의 application들(hello\_world, hello\_qt)을 Build하고 scp를 사용하여 보드로 전송한다.

```
$ unset LD_LIBRARY_PATH
$ source ~/sdk/environment-setup-aarch64-xilinx-linux
$ cd ~/work/zynqmp_linux/petalinux/workspaces/hello_world
$ mkdir build
$ cd build
$ cmake ..
$ make
$ scp hello_world root@172.30.1.39:.
$ cd ../../hello_qt
$ mkdir build
$ cd build
$ cmake ..
$ make
$ scp hello_qt root@172.30.1.39:.
```

## Step 4 SDK Application Test

Welcome to inipro.net

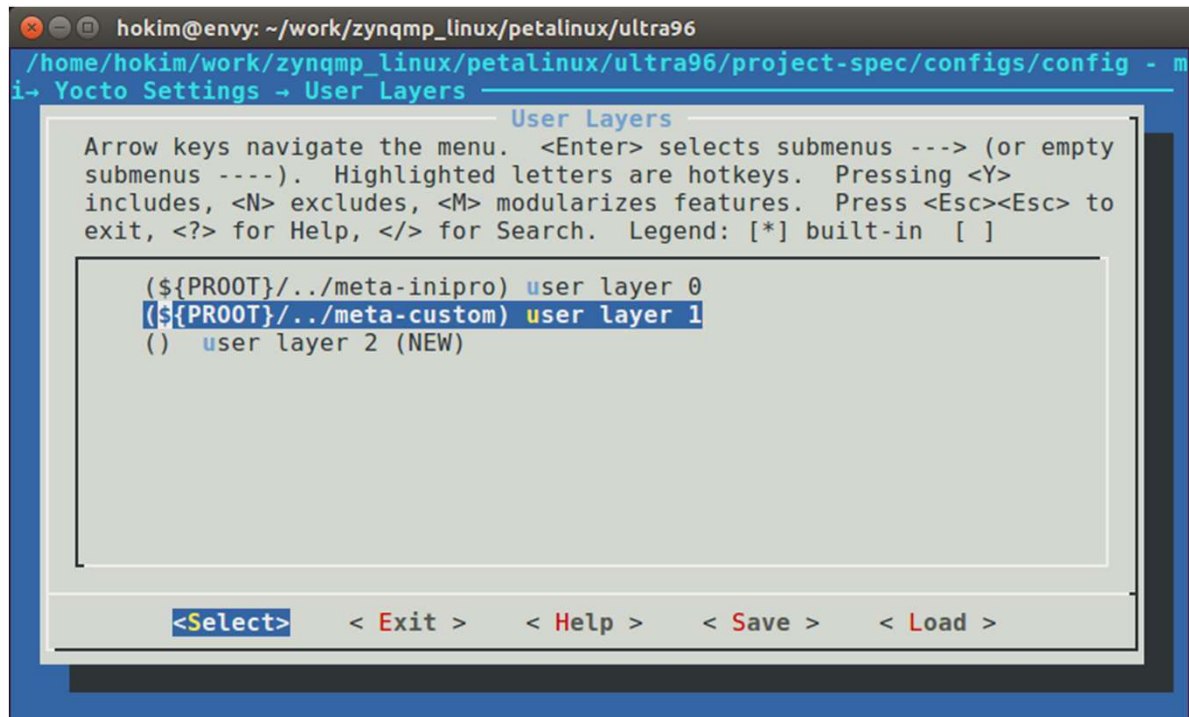
❖ 보드에 ssh로 접속하여 다음처럼 테스트한다.

```
ultra96$ cd ~  
ultra96$ ./hello_world  
Hello World!!!  
ultra96$ ./hello_qt -platform linuxfb
```

❖ 붉은색은 terminal로 출력되는 실행결과이며 hello\_qt의 실행 결과는 Monitor에서 왼쪽 그림처럼 출력되어야 한다.



## Step 5 SDK Application Test



```
hokim@envy: ~/work/zynqmp_linux/petalinux/ultra96
/home/hokim/work/zynqmp_linux/petalinux/ultra96/project-spec/configs/config - m
i→ Yocto Settings → User Layers

User Layers
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]

({${PROOT}}/../meta-inipro) user layer 0
({${PROOT}}/../meta-custom) user layer 1
() user layer 2 (NEW)

<Select> < Exit > < Help > < Save > < Load >
```

❖ Application들을 위한 recipe들  
을 가지고 있는  
~/work/zynqmp\_linux/petalin  
ux 아래에 있는 meta-custom  
layer를 그림과 같이 추가한다.

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

❖ Yocto Settings ⇒ User Layers  
에 그림과 같이  
\${PROOT}/../meta-custom을  
추가한다.

## Step 6 Application Recipes Build 1

- ❖ ~/work/zynqmp\_linux/petalinux/meta-custom/recipes-apps 아래에 있는 helloworld 와 helloqt는 각각 hello\_world와 hello\_qt Application을 위한 recip들이고 autostart는 hello\_qt를 Linux가 boot 하면서 자동으로 실행되도록 하는 recip이다.
- ❖ 다음의 과정을 통해 이전의 recipe들에 대응하는 rpm package들을 생성한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96  
$ petalinux-build -c helloworld  
$ petalinux-build -c helloqt  
$ petalinux-build -c autostart
```

## Step 6 Application Recipes Build 2

❖ Build된 rpm들을 다음과 같이 확인할 수 있다. (※ 붉은색은 화면출력이다.)

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96/build/tmp/deploy/rpm/
$ find . -name 'helloworld*.rpm'
./aarch64/helloworld-lic-1.0-r0.aarch64.rpm
./aarch64/helloworld-dbg-1.0-r0.aarch64.rpm
./aarch64/helloworld-1.0-r0.aarch64.rpm
./aarch64/helloworld-dev-1.0-r0.aarch64.rpm
$ find . -name 'helloqt*.rpm'
./aarch64/helloqt-lic-1.0-r0.aarch64.rpm
./aarch64/helloqt-1.0-r0.aarch64.rpm
./aarch64/helloqt-dev-1.0-r0.aarch64.rpm
./aarch64/helloqt-dbg-1.0-r0.aarch64.rpm
$ find . -name 'autostart*.rpm'
./aarch64/autostart-dev-1.0-r0.aarch64.rpm
./aarch64/autostart-1.0-r0.aarch64.rpm
./aarch64/autostart-dbg-1.0-r0.aarch64.rpm
./aarch64/autostart-lic-1.0-r0.aarch64.rpm
```

## Step 7 Repository Setup for RPMs

- ❖ Application Recipes Build에서 만든 rpm package들을 보드에서 package management tool인 dnf로 설치하도록 하기위해 host에 있는 rpm 폴더를 repository로 다음과 같이 만든다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96  
$ petalinux-build -c package-index
```

- ❖ 위의 과정은 recipe가 추가 또는 변경되는 경우 반복적으로 수행되어야 한다.
- ❖ Repository를 위한 http server가 실행되도록 다음을 수행한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96/build/tmp/deploy/rpm/  
$ python3 -m http.server 5678
```

## Step 8 Package Install & Test 1

```
ultra96$ vi /etc/yum.repos.d/oe-remote-repo.repo
[oe-remote-repo]
name=OE Remote Repo:
baseurl=http://172.30.1.25:5678
gpgcheck=0
ultra96$ dnf -y --refresh install helloworld
ultra96$ rpm -ql helloworld
/usr
/usr/bin
/usr/bin/hello_world
ultra96$ hello_world
Hello World!!!
ultra96$ dnf -y --refresh install helloqt
/usr
/usr/bin
/usr/bin/hello_qt
ultra96$ hello_qt -platform linuxfb
ultra96$ dnf -y --refresh install autostart
ultra96$ rpm -ql autostart
```

- ❖ 보드에 ssh로 접속하여 dnf package management tool을 사용하여 Application recipe들의 rpm package들을 설치하고 rpm들이 어떠한 파일들로 System에 설치되었는지 확인한다.
- ❖ 아래의 oe-remote-repo.repo의 172.30.1.25는 사용자의 host ip로 바꾸어준다.
- ❖ hello\_qt의 실행결과는 SDK Application Test 그림과 같다.

## Step 8 Package Install & Test 2

```
/etc
/etc/init.d
/etc/init.d/autostart
ultra96$ rpm -q --scripts autostart
postinstall scriptlet (using /bin/sh):
# autostart - postinst
#!/bin/sh
set -e
if true && type update-rc.d >/dev/null 2>/dev/null; then
    if [ -n "$D" ]; then
        OPT="-f -r $D"
    else
        OPT="-f -s"
    fi
    update-rc.d $OPT autostart start 99 5 .
fi
ultra96$ ls -l /etc/rc5.d/S99autostart
lrwxrwxrwx 1 root root 19 Feb 14 11:18 /etc/rc5.d/S99autostart -> ../init.d/autostart
ultra96$ reboot
```

❖ 마지막 reboot 명령에 의해 보드가 다시 boot되면 그림1과 같은 화면이 자동으로 나타남을 확인할 수 있다.

# Labs

- Lab1. Petalinux Linux System Build
- Lab2. Linux Application Build Flow
- **Lab3. HW Linux Application Build**
- Lab4. Custom HW Linux Application Build
- Lab5. Linux Xilinx Video Pipeline

## Lab3. HW Linux Application Build Overview

- ❖ ZynqMP PL영역에 Pmod8ld, PmodAls, PmoTmp2등을 구동하기 위해 Xilinx HW IP(axi gpio, axi quad spi, axi iic)들을 갖는 Vivado Project를 구성하고, 이들 HW IP들을 위한 Linux Application들을 SDK와 recipe들을 통해 Build 한다.
- ❖ 각 HW IP(GPIO, SPI, I2C Controller)들을 Userspace에서 직접적으로 다루는 방법과 Kernel Module을 만들어 다루는 방법(Out of Tree Build)을 익힌다.
- ❖ Kernel Module중의 하나를 In Tree방식으로 다루어 Upstream Kernel Source을 수정하는 방법을 익힌다.



## Step 1 HW Preparation

- ❖ 보드에 Usb-to-Uart와 Pmod Module들을 연결한다.
- ❖ PMod96보드의 PMOD\_A, PMOD\_B, PMOD\_C에 각각 Pmod8ld, PmodAls, PmodTmp2을 연결한다.
- ❖ PmodAls는 PMOD\_B의 top에 연결한다.
- ❖ Usb-to-Uart와 host의 usb 포트를 연결한다.

## Step 2 Export Vivado Project

❖ Ultra96v1(hw2\_v1.tcl) 또는 Ultra96v2(hw2\_v2.tcl) Vivado Project를 만든다.

```
$ cd ~/work/zynqmp_linux/  
$ vivado -nolog -nojournal -mode batch -source hw2_v1.tcl  
$ cd hw2  
$ vivado hw2.xpr
```

```
$ cd ~/work/zynqmp_linux/  
$ vivado -nolog -nojournal -mode batch -source hw2_v2.tcl  
$ cd hw2  
$ vivado hw2.xpr
```

❖ Bitstream을 생성하고 HW export를 한다.

## Step 3 Petalinux Project Update with new HW

❖ 다음의 명령을 사용하여 hw2/ 의 xsa파일을 기초로 하여 Petalinux Project(ultra96)의 HW를 변경한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96  
$ petalinux-config --silentconfig --get-hw-description=../hw2/
```

## Step 4 New Device Tree Generation

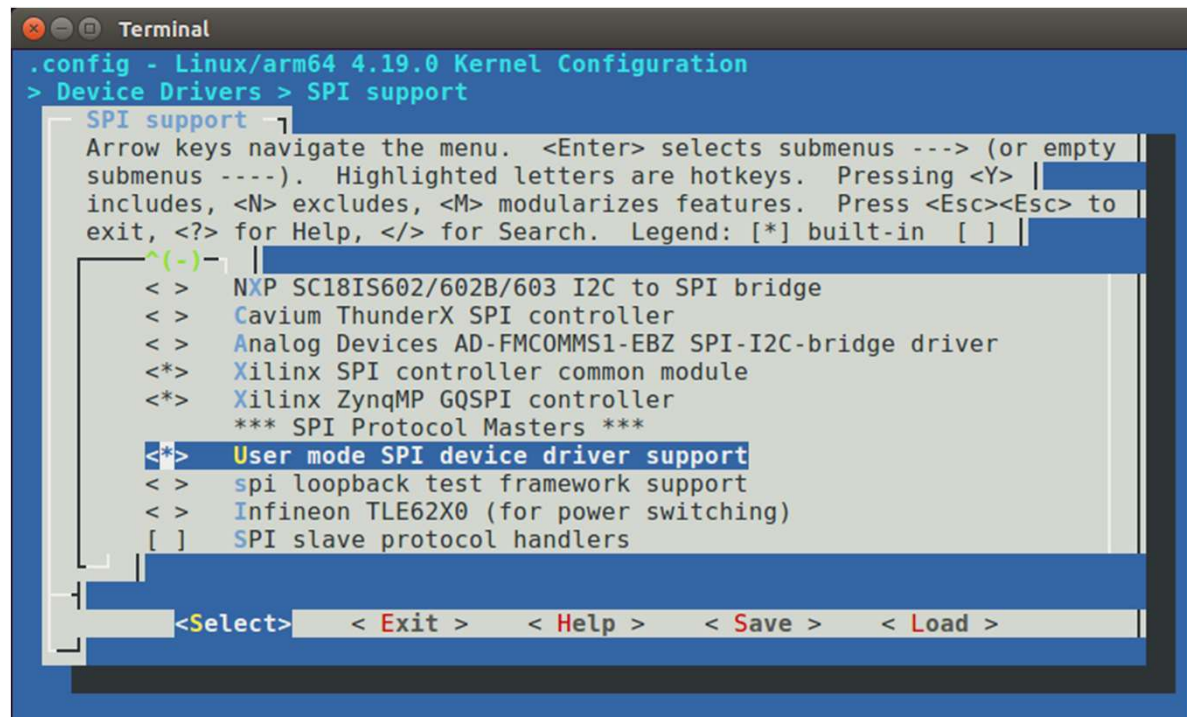
❖ 다음의 명령으로 new HW에 기초한 Device Tree를 Generation 한다.

```
$ 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/pl.dtsi의 내용을 확인한다.

❖ pl.dtsi은 새로 추가된 PL영역의 HW IP들에 대한 Device Tree 정보를 가지고 있다.

## Step 5 Kernel Configuration for SPIDEV 1



```
Terminal
.config - Linux/arm64 4.19.0 Kernel Configuration
> Device Drivers > SPI support
  SPI support
  Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
  submenu ---). Highlighted letters are hotkeys. Pressing <Y>
  includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to
  exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]
  ^(-)-
  < > NXP SC18IS602/602B/603 I2C to SPI bridge
  < > Cavium ThunderX SPI controller
  < > Analog Devices AD-FMCOMMS1-EBZ SPI-I2C-bridge driver
  <*> Xilinx SPI controller common module
  <*> Xilinx ZynqMP GQSPI controller
  *** SPI Protocol Masters ***
  <*> User mode SPI device driver support
  < > spi loopback test framework support
  < > Infineon TLE62X0 (for power switching)
  [ ] SPI slave protocol handlers

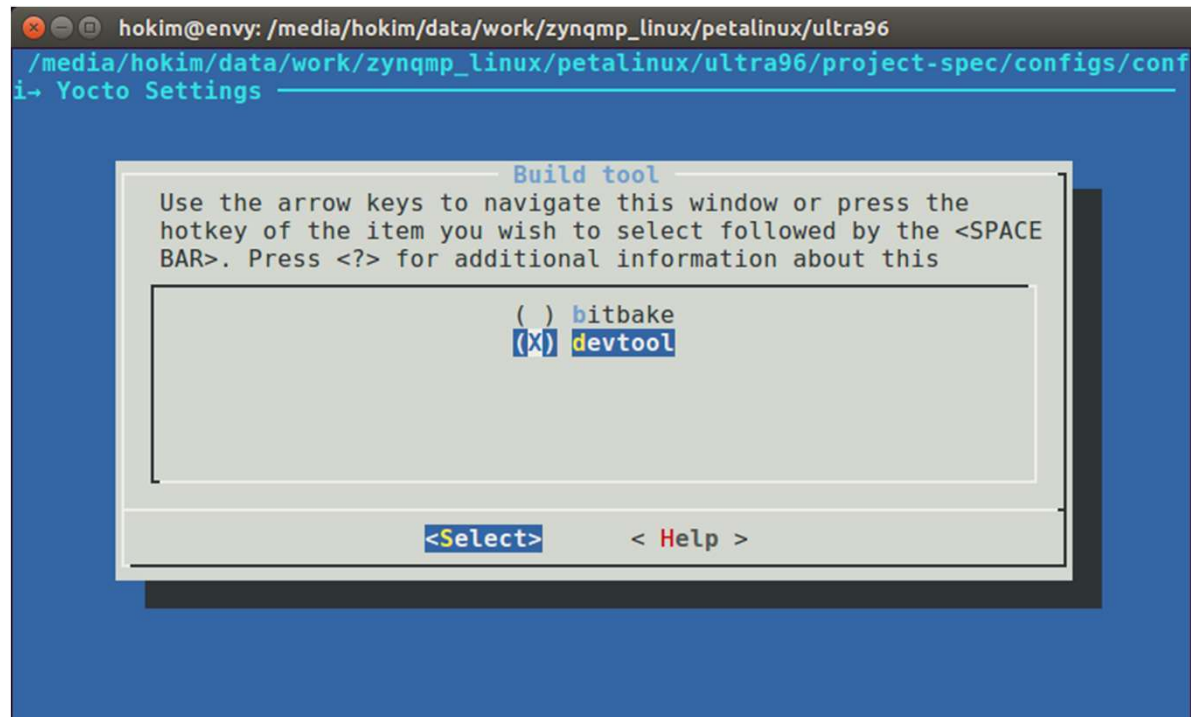
  <Select> <Exit> <Help> <Save> <Load>
```

❖ SPIDEV는 SPI Controller를 Userspace에서 다루기 위해 필요하고 다음의 과정을 거쳐 Kernel에 추가한다.

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

❖ Drivers ⇒ SPI support에서 User mode SPI device support를 활성화시킨다.

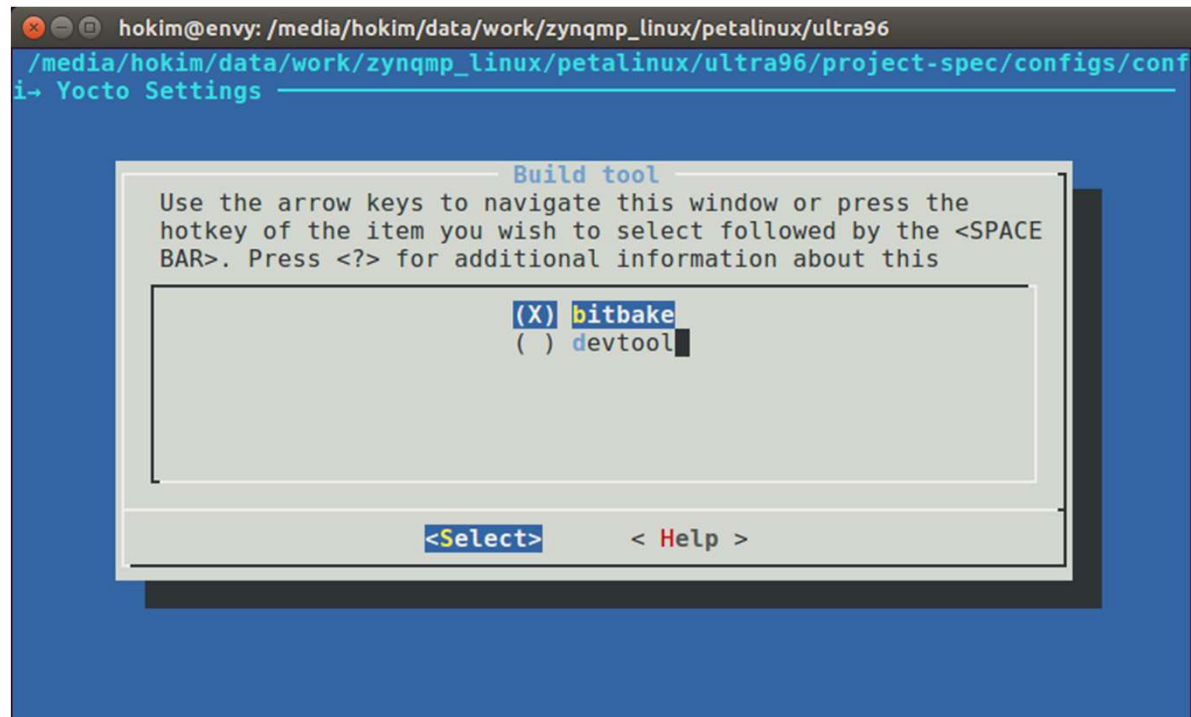
## Step 5 Kernel Configuration for SPIDEV 2



- ❖ 추가된 설정내용을 보존하기 위해 다음의 과정을 수행해야 한다.
- ❖ 아래와 같은 명령를 사용하여 Petalinux Configuration화면이 나오면 Yocto Settings ⇒ Build tool에서 devtool을 선택 하고 종료한다.

```
$ petalinux-config
```

## Step 5 Kernel Configuration for SPIDEV 3



❖ 아래 명령에 의해 생성된 project-spec/meta-user/recipes-kernel/linux/linux-xlnx 폴더 아래의 linux-xlnx\_2019.2.bbappend와 linux-xlnx/devtool-fragment.cfg 파일들을 확인한다.

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

❖ 아래 명령을 사용하여 Petalinux Configuration 화면을 다시 연 후 그림처럼 Yocto Settings ⇒ Build tool 에서 bitbake를 선택하고 종료한다.

```
$ petalinux-config
```

## Step 6 Device Tree Modification for Ultra96v1

❖ 다음 명령으로 Device Tree를 수정한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96
$ vi project-spec/meta-user/recipe-bsp/device-
tree/files/system-user.dtsi
```

- ❖ axi\_quad\_spi\_0 node아래에 spidev node를 그림처럼 추가한다.

❖ Line 5-110이 추가된 부분이다.

```
hokim@envy: /media/hokim/data/work/zynqmp_linux/petalinux/ultra96
1 /include/ "system-conf.dtsi"
2 / {
3 };
4
5 &axi_quad_spi_0{
6     spidev@0 {
7         compatible = "rohm,dh2228fv";
8         reg = <0>;
9         spi-max-frequency = <50000000>;
10    };
11 };
12
```



## Step 6 Device Tree Modification for Ultra96v2

❖ 다음 명령으로 Device Tree를 수정한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96  
$ vi project-spec/meta-user/recipe-bsp/device-tree/files/system-user.dtsi
```

❖ axi\_quad\_spi\_0 node아래에 spidev node를 그림처럼 추가한다.

❖ Line 49-55가 추가된 부분이다.

```
hokim@envy: /media/hokim/data/work/zynqmp_linux/petalinux/ultra96  
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 };  
48  
49 &axi_quad_spi_0 {  
50     spidev@0 {  
51         compatible = "rohm,dh2228fv";  
52         reg = <0>;  
53         spi-max-frequency = <50000000>;  
54     };  
55 };
```

## Step 7 Update BOOT.BIN, image.ub

❖ 새로운 HW를 위한 BOOT.BIN과 image.ub를 다음과 같이 Update 한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96
$ petalinux-build -c virtual/boot-bin
$ 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
$ scp images/linux/{BOOT.BIN,image.ub} root@172.30.1.39:/media/card
```

## Step 8 HW Check after Reboot 1 for Ultra96v1

❖ GPIO, SPI, I2C Controller Linux Driver들에 의해 Userspace로 Export되는 HW 정보를 Check하도록 한다.

```
ultra96$ ls -l /sys/class/gpio/gpio*
lrwxrwxrwx 1 root root 0 Feb 19 19:49 /sys/class/gpio/gpio356 -> ../../devices/platform/amba/ff0a0000.gpio/gpiochip1/gpio/gpio356
lrwxrwxrwx 1 root root 0 Feb 19 19:49 /sys/class/gpio/gpiochip326 -> ../../devices/platform/amba/ff030000.i2c/i2c-1/i2c-7/7-005e/gpio/gpiochip326
lrwxrwxrwx 1 root root 0 Feb 19 19:49 /sys/class/gpio/gpiochip330 -> ../../devices/platform/amba/ff0a0000.gpio/gpio/gpiochip330
lrwxrwxrwx 1 root root 0 Feb 19 19:49 /sys/class/gpio/gpiochip504 -> ../../devices/platform/amba_pl@0/80000000.gpio/gpio/gpiochip504
ultra96$ cat /sys/class/gpio/gpiochip326/label
tps65086-gpio
ultra96$ cat /sys/class/gpio/gpiochip330/label
zynqmp_gpio
ultra96$ cat /sys/class/gpio/gpiochip504/label
/amba_pl@0/gpio@80000000
```

## Step 8 HW Check after Reboot 1 for Ultra96v2

❖ GPIO, SPI, I2C Controller Linux Driver들에 의해 Userspace로 Export되는 HW 정보를 Check하도록 한다.

```
ultra96$ ls -l /sys/class/gpio/gpio*  
lrwxrwxrwx 1 root root 0 Feb 15 10:56 /sys/class/gpio/gpio356 -> ../../devices/platform/amba/ff0a0000.gpio/gpiochip1/gpio/gpio356  
lrwxrwxrwx 1 root root 0 Jan  1  1970 /sys/class/gpio/gpiochip330 -> ../../devices/platform/amba/ff0a0000.gpio/gpio/gpiochip330  
lrwxrwxrwx 1 root root 0 Jan  1  1970 /sys/class/gpio/gpiochip504 -> ../../devices/platform/amba_pl@0/80000000.gpio/gpio/gpiochip504  
ultra96$ cat /sys/class/gpio/gpiochip330/label  
zynqmp_gpio  
ultra96$ cat /sys/class/gpio/gpiochip504/label  
/amba_pl@0/gpio@80000000
```

## Step 8 HW Check after Reboot 2

```
ultra96$ ls -l /sys/class/spi_master/spi*
lrwxrwxrwx 1 root root 0 Jan  1  1970 /sys/class/spi_master/spi0 -
> ../../devices/platform/amba_pl@0/80020000.axi_quad_spi/spi_master/spi0
lrwxrwxrwx 1 root root 0 Jan  1  1970 /sys/class/spi_master/spi1 -> ../../devices/platform/amba/ff040000.spi/spi_master/spi1
lrwxrwxrwx 1 root root 0 Jan  1  1970 /sys/class/spi_master/spi2 -> ../../devices/platform/amba/ff050000.spi/spi_master/spi2
ultra96$ ls /sys/class/spi_master/spi0/
device of_node power spi0.0 statistics subsystem uevent
ultra96$ i2cdetect -l
i2c-3  i2c      i2c-1-mux (chan_id 0)      I2C adapter
i2c-10 i2c      i2c-1-mux (chan_id 7)      I2C adapter
i2c-1  i2c      Cadence I2C at ff030000    I2C adapter
i2c-8  i2c      i2c-1-mux (chan_id 5)      I2C adapter
i2c-6  i2c      i2c-1-mux (chan_id 3)      I2C adapter
i2c-4  i2c      i2c-1-mux (chan_id 1)      I2C adapter
i2c-2  i2c      ZynqMP DP AUX              I2C adapter
i2c-0  i2c      xiic-i2c                   I2C adapter
i2c-9  i2c      i2c-1-mux (chan_id 6)      I2C adapter
i2c-7  i2c      i2c-1-mux (chan_id 4)      I2C adapter
i2c-5  i2c      i2c-1-mux (chan_id 2)      I2C adapter
```

## Step 8 HW Check after Reboot 3

```
ultra96$ i2cdetect -y -r 0
```

```
 0 1 2 3 4 5 6 7 8 9 a b c d e f
```

```
00:  ---
```

```
10: ---
```

```
20: ---
```

```
30: ---
```

```
40: --- 4b ---
```

```
50: ---
```

```
60: ---
```

```
70: ---
```

## Step 9 SDK HW Application Build & Test 1

❖ SDK를 사용하여 Application을 다음과 같이 Build한다.

```
$ cd ~/work/zynqmp_linux/petalinux/workspaces
$ unset LD_LIBRARY_PATH
$ source ~/sdk/environment-setup-aarch64-xilinx-linux
$ cd gpio_test
$ mkdir build
$ cd build
$ cmake ..
$ make
$ scp gpio_test root@172.30.1.39:.
$ cd ../../spi_test
$ mkdir build
```

```
$ cd build
$ cmake ..
$ make
$ scp spi_test root@172.30.1.39:.
$ cd ../../i2c_test
$ mkdir build
$ cd build
$ cmake ..
$ make
$ scp i2c_test root@172.30.1.39:.
```

## Step 9 SDK HW Application Build & Test 2

- ❖ 보드에서 Test는 다음과 같다.
- ❖ gpio\_test 의 결과는 Pmod8ld의 led 점멸로 확인한다.
- ❖ spi\_test 와 i2c\_test는 1초 간격으로 각각의 adc값을 계속 출력하기 때문에 중지하려면 Ctrl+C를 눌러야 한다.

```
ultra96$ cd ~  
ultra96$ ./gpio_test 3  
ultra96$ ./spi_test  
light = 44  
ultra96$ ./i2c_test  
temp = 3176  
ultra96$ rm gpio_test spi_test i2c_test
```



## Step 10 HW Application Recipe Build & Test

❖ ~/work/zynqmp\_linux/petalinux/meta-custom/recipes-apps 아래에 있는  
gpiotest, spitest, i2ctest recipes들을 다음과 같이 Build하고 보드에서 Test 한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96
$ petalinux-build -c gpiotest
$ petalinux-build -c spitest
$ petalinux-build -c i2ctest
$ petalinux-build -c package-index

$ cd build/tmp/deploy/rpm
$ python3 -m http.server 5678
```

```
ultra96$ dnf -y --refresh install gpiotest spitest i2ctest
ultra96$ gpio_test 5
ultra96$ spi_test
light = 44
ultra96$ i2c_test
temp = 3176
```

## Step 11 SDK HW Kernel Module Build 1 for Ultra96v1

```
hokim@envy: /media/hokim/data/work/zynqmp_linux/petalinux/ultra96
4
5 /{
6     pmod8ld {
7         compatible = "inipro,pmod8ld";
8         led-gpios = <&axi_gpio_0 0 0 0>, <&axi_gpio_0 1 0 0>,
9                     <&axi_gpio_0 2 0 0>, <&axi_gpio_0 3 0 0>,
10                    <&axi_gpio_0 4 0 0>, <&axi_gpio_0 5 0 0>,
11                    <&axi_gpio_0 6 0 0>, <&axi_gpio_0 7 0 0>;
12     };
13 };
14
15 &axi_quad_spi_0{
16     pmodals@0 {
17         compatible = "inipro,pmodals";
18         reg = <0>;
19         spi-max-frequency = <50000000>;
20         spi-cpha;
21         spi-cpol;
22     };
23 };
24
25 &axi_iic_0 {
26     pmodtmp2@4b {
27         compatible = "inipro,pmodtmp2";
28         reg = <0x4b>;
29     };
30 };
31
```

❖ Kernel Module들을 사용하기 위해 Device Tree를 수정한다.

```
$ ~/work/zynqmp_linux/petalinux/ultra96
$ vi project-spec/meta-user/recipes-bsp/device-tree/files/system-user.dtsi
```

❖ Line 5 -30이 수정된 부분이다.

## Step 11 SDK HW Kernel Module Build 1 for Ultra96v2

```
hokim@envy: /media/hokim/data/work/zynqmp_linux/petalinux/ultra96
48
49 {
50     pmod8ld {
51         compatible = "inipro,pmod8ld";
52         led-gpios = <&axi_gpio_0 0 0 0>, <&axi_gpio_0 1 0 0>,
53                   <&axi_gpio_0 2 0 0>, <&axi_gpio_0 3 0 0>,
54                   <&axi_gpio_0 4 0 0>, <&axi_gpio_0 5 0 0>,
55                   <&axi_gpio_0 6 0 0>, <&axi_gpio_0 7 0 0>;
56     };
57 };
58
59 &axi_quad_spi_0{
60     pmodals@0 {
61         compatible = "inipro,pmodals";
62         reg = <0>;
63         spi-max-frequency = <50000000>;
64         spi-cpha;
65         spi-cpol;
66     };
67 };
68
69 &axi_iic_0 {
70     pmodtmp2@4b {
71         compatible = "inipro,pmodtmp2";
72         reg = <0x4b>;
73     };
74 };
75
```

❖ Kernel Module들을 사용하기 위해 Device Tree를 수정한다.

```
$ ~/work/zynqmp_linux/petalinux/ultra96
$ vi project-spec/meta-user/recipes-bsp/device-tree/files/system-user.dtsi
```

❖ Line 49-74가 수정된 부분이다.

## Step 11 SDK HW Kernel Module Build 2

❖ Device Tree를 다시 Build하면 image.ub가 Update되며 이를 보드로 scp한다.

```
$ petalinux-build -c device-tree  
$ scp images/linux/image.ub root@172.30.1.39:/media/card
```

## Step 11 SDK HW Kernel Module Build 3

- ❖ SDK를 사용하여 Kernel Module들을 Build한다.
- ❖ 먼저 SDK가 설치된 폴더의 Kernel Source에 Out of Tree Build를 할 수 있도록 Setup한다. (make modules\_prepare)

```
$ cd ~/sdk
$ unset LD_LIBRARY_PATH
$ source environment-setup-aarch64-xilinx-linux
$ cd sysroots/aarch64-xilinx-linux/usr/src/kernel
$ make modules_prepare
```

```
$ cd ~/work/zynqmp_linux/petalinux/workspaces
$ cd pmod8ld
$ KERNEL_SRC=$SDKTARGETSYSROOT/usr/src/kernel make
$ scp pmod8ld.ko root@172.30.1.39:/lib/modules/4.19.0-xilinx-v2019.2/extra
$ make clean
$ cd ../pmodals
$ KERNEL_SRC=$SDKTARGETSYSROOT/usr/src/kernel make
$ scp pmodals.ko root@172.30.1.39:/lib/modules/4.19.0-xilinx-v2019.2/extra
$ make clean
$ cd ../pmodtmp2
$ KERNEL_SRC=$SDKTARGETSYSROOT/usr/src/kernel make
$ scp pmodtmp2.ko root@172.30.1.39:/lib/modules/4.19.0-xilinx-v2019.2/extra
$ make clean
```

## Step 11 SDK HW Kernel Module Build 4

❖ 보드에서 module을 추가하고 boot시 자동 load되도록 하기위해 다음의 과정을 수행하고 다시 boot한다.

```
ultra96$ depmod -a  
ultra96$ echo pmod8ld > /etc/modules-load.d/pmod8ld.conf  
ultra96$ echo pmodals > /etc/modules-load.d/pmodals.conf  
ultra96$ echo pmodtmp2 > /etc/modules-load.d/pmodtmp2.conf  
ultra96$ reboot
```

## Step 11 SDK HW Kernel Module Build 5

- ❖ 보드에 다시 접속하여 Kernel Module이 load되었는지 확인하고 Test를 수행한다.
- ❖ 다음 Test를 위해 install된 Kernel Module들을 cleanup한다.

```
ultra96$ lsmod
```

Module	Size	Used by
pmod8ld	16384	0
pmodtmp2	16384	0
pmodals	16384	0

```
ultra96$ cd /sys/devices/platform/pmod8ld
```

```
ultra96$ cat bits
```

0

```
ultra96$ echo 10 > bits
```

```
ultra96$ cat bits
```

10

```
ultra96$ cd /sys/class/spi_master/spi0/spi0.0
```

```
ultra96$ cat adc
```

42

```
ultra96$ cd /sys/class/i2c-adapter/i2c-0/0-004b
```

```
ultra96$ cat adc
```

3144

```
ultra96$ rm /etc/modules-load.d/pmod*.conf
```

```
ultra96$ rm /lib/modules/4.19.0-xilinx-v2019.2/extra/pmod*.ko
```

```
ultra96$ depmod -a
```

## Step 12 HW Kernel Module Recipe Build & Test 1

- ❖ ~/work/zynqmp\_linux/petalinux/meta-custom/recipes-modules 아래에 있는 Pmod8ld, PmodAls, PmodTmp2 recipes들을 다음과 같이 Build하고 보드에서 Test 한다.
- ❖ Test를 종료하고 나서 PmodTmp2 Kernel Module을 제거한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96
$ petalinux-build -c pmod8ld
$ petalinux-build -c pmodals
$ petalinux-build -c pmodtmp2
$ petalinux-build -c package-index
```

```
$ cd build/tmp/deploy/rpm
$ python3 -m http.server 5678
```

```
ultra96$ dnf -y --refresh install kernel-module-pmod8ld kernel-
module-pmodals kernel-module-pmodtmp2
ultra96$ rpm -ql kernel-module-pmod8ld-4.19.0-xilinx-v2019.2
/etc
/etc/modules-load.d
/etc/modules-load.d/pmod8ld.conf
/lib
/lib/modules
/lib/modules/4.19.0-xilinx-v2019.2
/lib/modules/4.19.0-xilinx-v2019.2/extra
/lib/modules/4.19.0-xilinx-v2019.2/extra/pmod8ld.ko
ultra96$ rpm -ql kernel-module-pmodals-4.19.0-xilinx-v2019.2
/etc
/etc/modules-load.d
/etc/modules-load.d/pmodals.conf
/lib
/lib/modules
/lib/modules/4.19.0-xilinx-v2019.2
/lib/modules/4.19.0-xilinx-v2019.2/extra
/lib/modules/4.19.0-xilinx-v2019.2/extra/pmodals.ko
```

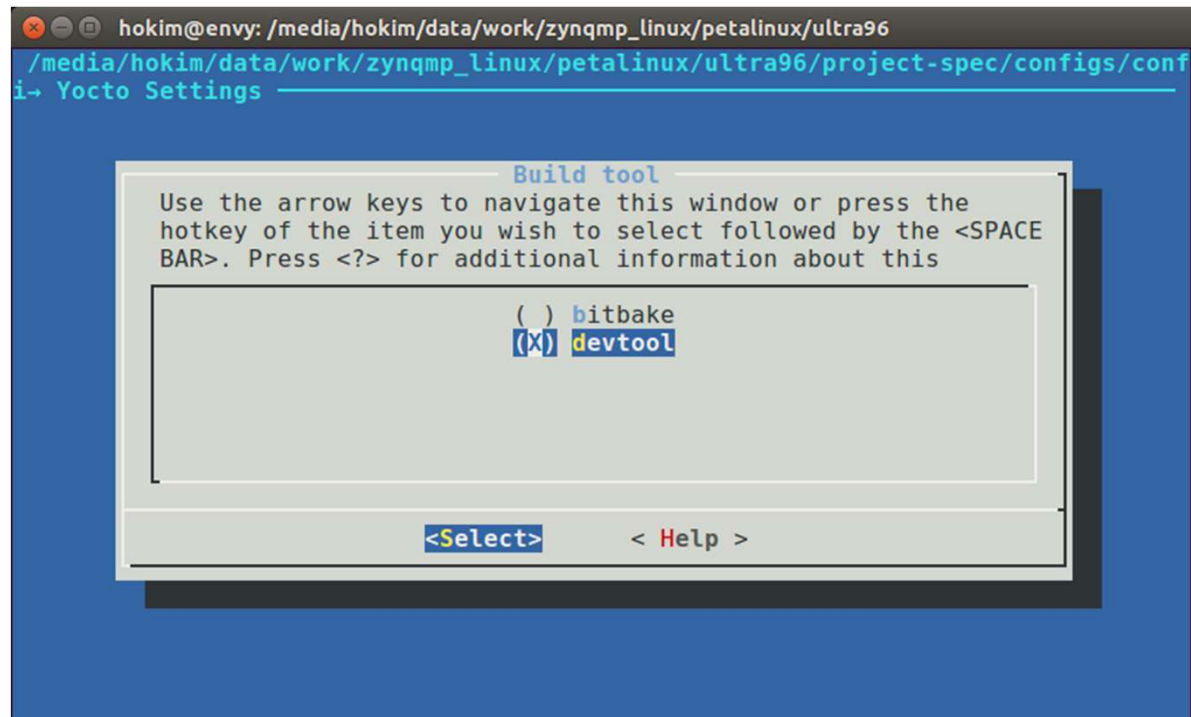


## Step 12 HW Kernel Module Recipe Build & Test 2

```
ultra96$ rpm -ql kernel-module-pmodtmp2-4.19.0-xilinx-v2019.2  
/etc  
/etc/modules-load.d  
/etc/modules-load.d/pmodtmp2.conf  
/lib  
/lib/modules  
/lib/modules/4.19.0-xilinx-v2019.2  
/lib/modules/4.19.0-xilinx-v2019.2/extra  
/lib/modules/4.19.0-xilinx-v2019.2/extra/pmodtmp2.ko
```

```
ultra96$ cd /sys/devices/platform/pmod8ld  
ultra96$ cat bits  
0  
ultra96$ echo 10 > bits  
ultra96$ cat bits  
10  
ultra96$ cd /sys/class/spi_master/spi0/spi0.0  
ultra96$ cat adc  
42  
ultra96$ cd /sys/class/i2c-adapter/i2c-0/0-004b  
ultra96$ cat adc  
3144  
ultra96$ dnf -y --refresh remove kernel-module-pmodtmp2
```

## Step 13 Upstream Kernel Source Modification 1



❖ Upstream Kernel Source를  
unpack, patch, configuration  
하기위해 다음의 과정을 수행  
한다.

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

## Step 13 Upstream Kernel Source Modification 2

```
hokim@envy: ~/work/zynqmp_linux/petalinux/ultra96/components/plnx_workspace/sources/linux-xlnx
1 menuconfig PMODS
2   bool "Pmod Support"
3   depends on HAS_IOMEM
4   help
5       Digilent PMOD Support
6
7 if PMODS
8
9   config PMODS_DEBUG
10    bool "Enable Debug Message"
11
12   config PMODTMP2
13    tristate "pmodtmp2"
14    depends on I2C
15    help
16        This is the Digilent PmodTMP2 driver.
17
18
19 endif # PMODS
~
~
~
```

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

- ❖ Upstream Kernel Source가 components/plnx\_workspace/sources/linux-xlnx에 있음을 확인할 수 있다.
- ❖ PmodTmp2 Kernel Module을 In Tree 하기 위해 Source를 다음과 같이 수정한다.

```
$ cd
~/work/zynqmp_linux/petalinux/ultra96/components/plnx_workspace/sources/linux-xlnx
$ mkdir drivers/staging/pmods
$ vi drivers/staging/pmods/Kconfig
```

## Step 13 Upstream Kernel Source Modification 3

```
hokim@envy: ~/work/zynqmp_linux/petalinux/ultra96/components/plnx_workspace/sources/linux-x
1 cflags-$(CONFIG_PMODS_DEBUG) += -DDEBUG
2
3 obj-$(CONFIG_PMODTMP2) += pmodtmp2.o
```

```
$ vi drivers/staging/pmods/Makefile
```

## Step 13 Upstream Kernel Source Modification 4

```
hokim@envy: ~/work/zynqmp_linux/petalinux/ultra96/components/plnx_workspace/sources/linux-xl
125 source "drivers/staging/mt7621-dts/Kconfig"
126
127 source "drivers/staging/gasket/Kconfig"
128
129 source "drivers/staging/axis-fifo/Kconfig"
130
131 source "drivers/staging/erofs/Kconfig"
132
133 source "drivers/staging/xlnx_ctrl_driver/Kconfig"
134
135 source "drivers/staging/xlnx_ernic/Kconfig"
136
137 source "drivers/staging/xroeframer/Kconfig"
138
139 source "drivers/staging/xroetraficgen/Kconfig"
140
141 source "drivers/staging/xlnxsync/Kconfig"
142
143 source "drivers/staging/xlnx_tsmux/Kconfig"
144
145 source "drivers/staging/pmods/Kconfig"
146
147 endif # STAGING
```

145,1 Bot

\$ vi drivers/staging/Kconfig

❖ Line 145를 추가한다.

## Step 13 Upstream Kernel Source Modification 5

```
hokim@envy: ~/work/zynqmp_linux/petalinux/ultra96/components/plnx_workspace/sources/linux-xl
41 obj-$(CONFIG_WILC1000) += wilc1000/
42 obj-$(CONFIG_MOST) += most/
43 obj-$(CONFIG_KS7010) += ks7010/
44 obj-$(CONFIG_GREYBUS) += greybus/
45 obj-$(CONFIG_BCM2835_VCHIQ) += vc04_services/
46 obj-$(CONFIG_DRM_VBOXVIDEO) += vboxvideo/
47 obj-$(CONFIG_PI433) += pi433/
48 obj-$(CONFIG_SOC_MT7621) += mt7621-pci/
49 obj-$(CONFIG_SOC_MT7621) += mt7621-pinctrl/
50 obj-$(CONFIG_SOC_MT7621) += mt7621-spi/
51 obj-$(CONFIG_SOC_MT7621) += mt7621-dma/
52 obj-$(CONFIG_SOC_MT7621) += mt7621-mmc/
53 obj-$(CONFIG_SOC_MT7621) += mt7621-eth/
54 obj-$(CONFIG_SOC_MT7621) += mt7621-dts/
55 obj-$(CONFIG_STAGING_GASKET_FRAMEWORK) += gasket/
56 obj-$(CONFIG_XIL_AXIS_FIFO) += axis-fifo/
57 obj-$(CONFIG_EROFs_FS) += erofs/
58 obj-y += xlnx_ctrl_driver/
59 obj-$(CONFIG_ERNIC) += xlnx_ernic/
60 obj-$(CONFIG_XROE_FRAMER) += xroeframer/
61 obj-$(CONFIG_XLNX_SYNC) += xlnxsync/
62 obj-$(CONFIG_XLNX_TSMUX) += xlnx_tsmux/
63 obj-$(CONFIG_PMODS) += pmods/

63,1 Bot
```

\$ vi drivers/staging/Makefile

❖ Line 63을 추가한다.

## Step 13 Upstream Kernel Source Modification 6

```
Terminal
.config - Linux/arm64 4.19.0 Kernel Configuration
> Device Drivers > Staging drivers - Pmod Support
  Staging drivers
  Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
  submenus ----). Highlighted letters are hotkeys. Pressing <Y> |
  includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to
  exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] |
  ^(-)-
  < > Xilinx AXI-Stream FIFO IP core driver
  < > EROFS filesystem support
  < * > FB Control driver
  < * > VPSS Control driver
  < > Xilinx ERNIC driver
  < > Xilinx Radio over Ethernet Framer driver
  < > Xilinx Radio over Ethernet Traffic Generator driver
  < > Xilinx Synchronizer
  < > Xilinx MPEG2 Transport Stream Muxer
  [*] Pmod Support --->
  <Select> < Exit > < Help > < Save > < Load >
```

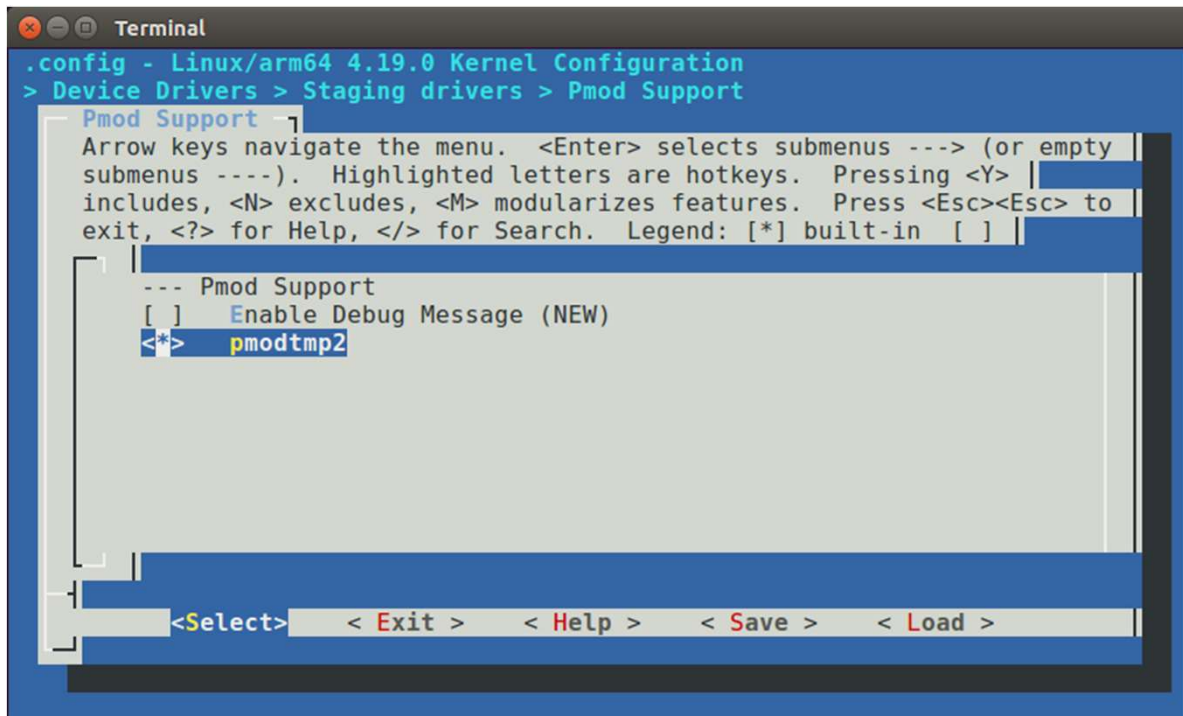
```
$ cp
~/work/zynqmp_linux/petalinux/workspaces/pmo
dtmp2/pmodtmp2.c drivers/staging/pmods/
```

❖ PmodTmp2 Kernel Module을  
Kernel Configuration에서 활  
성화하고 Build 한다.

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



## Step 13 Upstream Kernel Source Modification 7



```
Terminal
.config - Linux/arm64 4.19.0 Kernel Configuration
> Device Drivers > Staging drivers > Pmod Support
Pmod Support
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]
--- Pmod Support
[ ] Enable Debug Message (NEW)
[*] pmodtmp2
<Select> <Exit> <Help> <Save> <Load>
```

```
$ petalinux-build -c kernel
```



## Step 13 Upstream Kernel Source Modification 8

- ❖ Update된 Kernel Image를 가지고 있는 image.ub를 보드로 scp한다.

```
$ scp images/linux/image.ub root@172.30.1.39:/media/card
```

- ❖ 보드를 다시 boot한다.

```
ultra96$ reboot
```

- ❖ Test를 수행한다.

```
ultra96$ cd /sys/class/i2c-adapter/i2c-0/0-004b
ultra96$ cat adc
3144
```

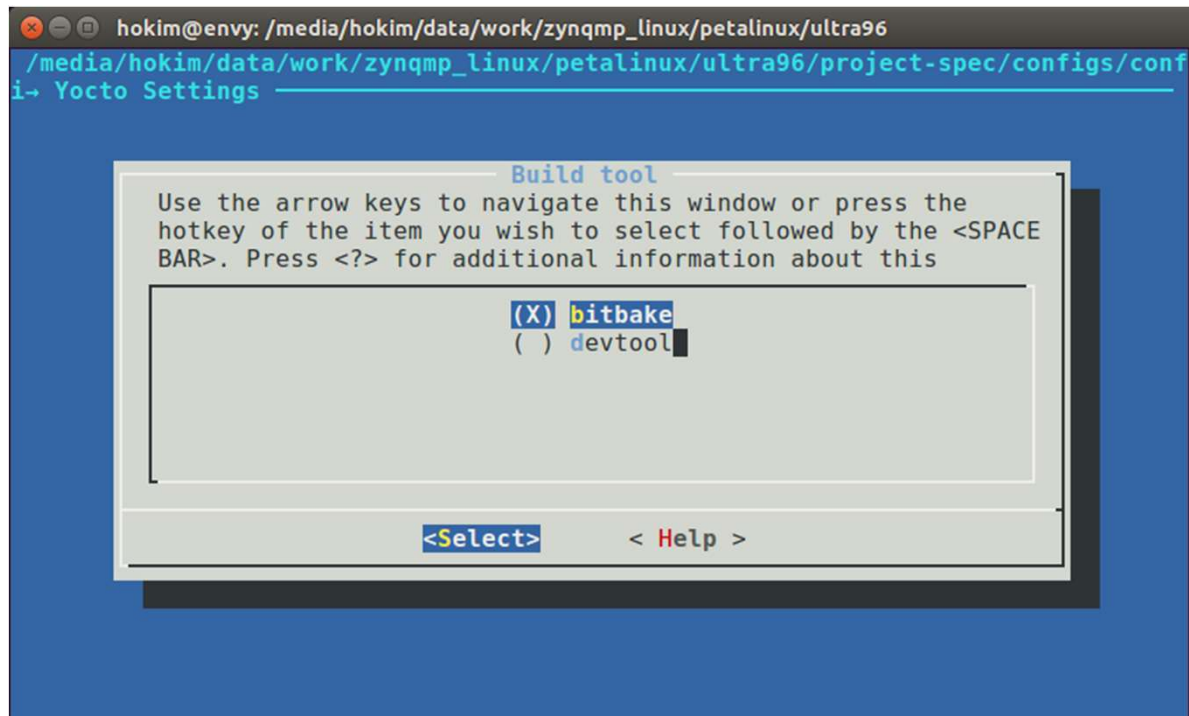
- ❖ Kernel Source의 수정된 코드에 대한 patch 파일을 만든다.

- ❖ git configuration이 되어있지 않으면 git user.email과 user.name을 설정한다.

```
$ cd
~/work/zynqmp_linux/petalinux/ultra96/components/plnx_workspa
ce/sources/linux-xlnx
$ git config --global user.email "hokim@inipro.net"
$ git config --global user.name "Hyunok Kim"
$ git add .
$ git commit -s -m "Add pmodtmp2"
$ git format-patch -1
```

## Step 13 Upstream Kernel Source Modification 9

❖ 수정된 작업에 대해 recipe를 Update하고 cleanup한다.



```
$ cd ~/work/zynqmp_linux/petalinux/ultra96  
$ petalinux-build -c kernel -x update-recipe  
$ petalinux-config
```

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

# Labs

- Lab1. Petalinux Linux System Build
- Lab2. Linux Application Build Flow
- Lab3. HW Linux Application Build
- **Lab4. Custom HW Linux Application Build**
- Lab5. Linux Xilinx Video Pipeline

## Lab4. Custom HW Linux Application Build Overview

- ❖ ZynqMP PL영역에 DMA(Direct Memory Access)기능이 있는 Custom HW IP(vadd)를 갖는 Vivado Project를 구성하고, 이 HW IP를 위한 Linux Application을 SDK와 recipe를 사용하여 Build 한다.
- ❖ Custom HW의 Controller Register에 Access하기 위해 UIO(Userspace IO)를 사용하며 DMA를 위해 물리적으로 연속적인 DDR Memory 할당과 할당된 Memory의 physical address와 virtual address에 대한 정보를 제공하는 ZOCL Driver를 사용한다.

## Step 1 HW preparation

❖ 보드에 Usb-to-Uart를 연결하고 host의 usb 포트에 연결한다.

## Step 2 Export Vivado Project

❖ Ultra96v1(hw3\_v1.tcl) 또는 Ultra96v2(hw3\_v2.tcl) Vivado Project를 만든다.

```
$ cd ~/work/zynqmp_linux/  
$ vivado -nolog -nojournal -mode batch -source hw3_v1.tcl  
$ cd hw3  
$ vivado hw3.xpr
```

```
$ cd ~/work/zynqmp_linux/  
$ vivado -nolog -nojournal -mode batch -source hw3_v2.tcl  
$ cd hw3  
$ vivado hw3.xpr
```

❖ Bitstream을 생성하고 HW export를 한다.

## Step 3 Petalinux Project Update with new HW

❖ 다음의 명령을 사용하여 hw3/ 의 xsa파일을 기초로 하여 Petalinux Project(ultra96)의 HW를 변경한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96  
$ petalinux-config --silentconfig --get-hw-description=../hw3/
```

## Step 4 New Device Tree Generation

❖ 다음의 명령으로 new HW에 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/pl.dtsi의 내용을 확인한다.

❖ pl.dtsi은 새로 추가된 PL영역의 HW IP들에 대한 Device Tree 정보를 가지고 있다.



## Step 5 Device Tree Modification for Ultra96v1

❖ 다음 명령으로 Device Tree를 수정한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96
$ vi project-spec/meta-user/recipe-bsp/device-
tree/files/system-user.dtsi
```

- ❖ ZOC Driver를 위한  
zyxclm\_drm node를 그림처럼  
추가하고 vadd\_0 node의  
compatible을 UIO compatible  
로 수정한다.

❖ Line 5-130| 변경된 부분이다.

```
hokim@envy: ~/work/zynqmp_linux/petalinux/ultra96
1 include/ "system-conf.dtsi"
2 / {
3 };
4
5 / {
6     zyxclm_drm {
7         compatible = "xlnx,zocl";
8     };
9 };
10
11 &vadd_0 {
12     compatible = "xlnx,generic-uis";
13 };
~
~
~
~
~
~
~
~
~
~
```

## Step 5 Device Tree Modification for Ultra96v2

❖ 다음 명령으로 Device Tree를 수정한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96
$ vi project-spec/meta-user/recipe-bsp/device-tree/files/system-user.dtsi
```

❖ ZOCL Driver를 위한  
zyxclm\_drm node를 그림처럼  
추가하고 vadd\_0 node의  
compatible을 UIO compatible  
로 수정한다.

❖ Line 49-57이 변경된 부분이다.

```
hokim@envy: ~/work/zynqmp_linux/petalinux/ultra96
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 };
48
49 / {
50     zyxclm_drm {
51         compatible = "xlnx,zocl";
52     };
53 };
54
55 &vadd_0 {
56     compatible = "xlnx,generic-uio";
57 };
```

## Step 6 Update BOOT.BIN, image.ub

❖ 새로운 HW를 위한 BOOT.BIN과 image.ub를 다음과 같이 Update 한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96
$ petalinux-build -c virtual/boot-bin
$ 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
$ scp images/linux/{BOOT.BIN,image.ub} root@172.30.1.39:/media/card
```

## Step 7 REBOOT

❖ 보드를 다시 boot하기전에 DRM(Direct Rendering Manager)와 UIO Device들의 정보를 확인한다.

```
ultra96$ ls /dev/dri/  
by-path card0  
ultra96$ ls -l /sys/class/drm/card*  
lrwxrwxrwx 1 root root 0 Feb 18 07:33 card0 -> ../../devices/platform/amba/fd4a0000.zynqmp-display/drm/card0  
lrwxrwxrwx 1 root root 0 Feb 18 07:33 card0-DP-1 -> ../../devices/platform/amba/fd4a0000.zynqmp-display/drm/card0/card0-DP-1  
ultra96$ lsmod  
Module                Size Used by  
uio_pdrv_genirq        16384 0  
ultra96$ ls /sys/class/uio/  
uio0 uio1 uio2 uio3  
ultra96$ reboot
```

## Step 8 SDK HW Application Build & Test 1

❖ SDK를 사용하여 Application을 다음과 같이 Build한다.

```
$ cd ~/work/zynqmp_linux/petalinux/workspaces
$ unset LD_LIBRARY_PATH
$ source ~/sdk/environment-setup-aarch64-xilinx-linux
$ cd vadd
$ mkdir build
$ cd build
$ cmake ..
$ make
$ scp vadd root@172.30.1.39:.
```

## Step 8 SDK HW Application Build & Test 2

❖ 보드에서 Test는 다음과 같다.

```
ultra96$ ls /dev/dri/  
by-path card0 card1 renderD128  
ultra96$ ls -l /sys/class/drm/card*  
lrwxrwxrwx 1 root root 0 Feb 18 07:59 /sys/class/drm/card0 -  
> ../../devices/platform/amba/fd4a0000.zynqmp-display/drm/card0  
lrwxrwxrwx 1 root root 0 Feb 18 07:59 /sys/class/drm/card0-DP-1 -  
> ../../devices/platform/amba/fd4a0000.zynqmp-  
display/drm/card0/card0-DP-1  
lrwxrwxrwx 1 root root 0 Feb 18 08:00 /sys/class/drm/card1 -  
> ../../devices/platform/zyxclm_drm/drm/card1  
ultra96$ ls /sys/class/uio/  
uio0 uio1 uio2 uio3 uio4
```

```
ultra96$ cat /sys/class/uio/uio4/name  
vadd  
ultra96$ ls /dev/uio4  
/dev/uio4  
ultra96$ cat /sys/class/uio/uio4/maps/map0/addr  
0x0000000080000000  
ultra96$ cat /sys/class/uio/uio4/maps/map0/size  
0x0000000000010000  
ultra96$ ./vadd  
TEST PASSED  
ultra96$ rm vadd
```

## Step 9 HW Application Recipe Build & Test

❖ ~/work/zynqmp\_linux/petalinux/meta-custom/recipes-apps 아래에 있는 vadd recipe를 다음과 같이 Build하고 보드에서 Test 한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96  
$ petalinux-build -c vadd  
$ petalinux-build -c package-index
```

```
$ cd build/tmp/deploy/rpm  
$ python3 -m http.server 5678
```

```
ultra96$ dnf -y --refresh install vadd  
ultra96$ vadd  
TEST PASSED
```

# Labs

- Lab1. Petalinux Linux System Build
- Lab2. Linux Application Build Flow
- Lab3. HW Linux Application Build
- Lab4. Custom HW Linux Application Build
- **Lab5. Linux Xilinx Video Pipeline**



## Lab5. Linux Xilinx Video Pipeline Overview

- ❖ ZynqMP PL영역에 Xilinx mipi csi2, frame buffer writer IP를 사용하여 Video Pipeline을 구성하고 PCAM 5C Camera Module을 연결하여 Camera 영상을 획득하는 방법을 익힌다.

## Step 1 HW preparation

- ❖ 보드에 Usb-to-Uart, PCAM 5C Camera, Mini DP to HDMI Adapter를 연결한다.
- ❖ Usb-to-Uart를 host의 usb 포트에 연결한다.
- ❖ Mini DP to HDMI Adapter는 Monitor와 연결한다.

## Step 2 Export Vivado Project

❖ Ultra96v1(hw4\_v1.tcl) 또는 Ultra96v2(hw4\_v2.tcl) Vivado Project를 만든다.

```
$ cd ~/work/zynqmp_linux/  
$ vivado -nolog -nojournal -mode batch -source hw4_v1.tcl  
$ cd hw4  
$ vivado hw4.xpr
```

```
$ cd ~/work/zynqmp_linux/  
$ vivado -nolog -nojournal -mode batch -source hw4_v2.tcl  
$ cd hw4  
$ vivado hw4.xpr
```

❖ Bitstream을 생성하고 HW export를 한다.

## Step 3 Petalinux Project Update with new HW

❖ 다음의 명령을 사용하여 hw4/ 의 xsa파일을 기초로 하여 Petalinux Project(ultra96)의 HW를 변경한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96  
$ petalinux-config --silentconfig --get-hw-description=../hw4/
```

## Step 4 New Device Tree Generation

❖ 다음의 명령으로 new HW에 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/pl.dtsi의 내용을 확인한다.

❖ pl.dtsi은 새로 추가된 PL영역의 HW IP들에 대한 Device Tree 정보를 가지고 있다.

## Step 5 Device Tree Modification for Ultra96v1 1

```
hokim@envy: /media/hokim/data/work/zynqmp_linux/petalinux/ultra96
5 &amba_pl {
6     pcam_clk: pcam_clk {
7         compatible = "fixed-clock";
8         #clock-cells = <0>;
9         clock-frequency = <12000000>;
10    };
11 };
12
13 &i2csw_1 {
14     ov5640: camera@3c {
15         compatible = "ovti,ov5640";
16         reg = <0x3c>;
17         clock-names = "xclk";
18         clocks = <&pcam_clk>;
19         powerdown-gpios = <&gpio 36 1>;
20         reset-gpios = <&gpio 39 1>;
21
22         port {
23             ov5640_out: endpoint {
24                 remote-endpoint = <&csiss_in>;
25                 clock-lanes = <0>;
26                 data-lanes = <1 2>;
27             };
28         };
29    };

```

29,1 13%

❖ 다음 명령으로 Device Tree를 수정한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96
$ vi project-spec/meta-user/recipe-bsp/device-
tree/files/system-user.dtsi
```

❖ Video Pipeline을 구성한다.

❖ Line 5-54가 변경된 부분이다.

## Step 5 Device Tree Modification for Ultra96v1 2

```
hokim@envy: /media/hokim/data/work/zynqmp_linux/petalinux/ultra96
30 };
31
32 &mipi_csi2_rx_subsys0 {
33     compatible = "xlnx,mipi-csi2-rx-subsystem-4.0";
34     reset-gpios = <&gpio 78 1>;
35 };
36
37 &csiss_port0 {
38     /delete-property/ xlnx,cfa-pattern;
39     xlnx,video-format = <0>;
40 };
41
42 &csiss_port1 {
43     /delete-property/ xlnx,cfa-pattern;
44     xlnx,video-format = <0>;
45 };
46
47 &csiss_in {
48     data-lanes = <1 2>;
49     remote-endpoint = <&ov5640_out>;
50 };
51
52 &v_frbuf_wr_0 {
53     compatible = "xlnx,axi-frmbuf-wr-v2.1";
54 };
```

30,1 96%

## Step 5 Device Tree Modification for Ultra96v2 1

❖ 다음 명령으로 Device Tree를 수정한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96
$ vi project-spec/meta-user/recipe-bsp/device-tree/files/system-user.dtsi
```

❖ ZOCL Driver를 위한  
zyxclm\_drm node를 그림처럼  
추가하고 vadd\_0 node의  
compatible을 UIO compatible  
로 수정한다.

❖ Line 49-98이 변경된 부분이다.

```
hokim@envy: /media/hokim/data/work/zynqmp_linux/petalinux/ultra96
49 &amba_pl {
50     pcam_clk: pcam_clk {
51         compatible = "fixed-clock";
52         #clock-cells = <0>;
53         clock-frequency = <12000000>;
54     };
55 };
56
57 &i2csw_1 {
58     ov5640: camera@3c {
59         compatible = "ovti,ov5640";
60         reg = <0x3c>;
61         clock-names = "xclk";
62         clocks = <&pcam_clk>;
63         powerdown-gpios = <&gpio 36 1>;
64         reset-gpios = <&gpio 39 1>;
65
66         port {
67             ov5640_out: endpoint {
68                 remote-endpoint = <&csiss_in>;
69                 clock-lanes = <0>;
70                 data-lanes = <1 2>;
71             };
72         };
73     };
74 };
```



## Step 5 Device Tree Modification for Ultra96v2 2

```
hokim@envy: /media/hokim/data/work/zynqmp_linux/petalinux/ultra96
74 };
75
76 &mipi_csi2_rx_subsys0 {
77     compatible = "xlnx,mipi-csi2-rx-subsystem-4.0";
78     reset-gpios = <&gpio 78 1>;
79 };
80
81 &csiss_port0 {
82     /delete-property/ xlnx,cfa-pattern;
83     xlnx,video-format = <0>;
84 };
85
86 &csiss_port1 {
87     /delete-property/ xlnx,cfa-pattern;
88     xlnx,video-format = <0>;
89 };
90
91 &csiss_in {
92     data-lanes = <1 2>;
93     remote-endpoint = <&ov5640_out>;
94 };
95
96 &v_frbbuf_wr_0 {
97     compatible = "xlnx,axi-frmbuf-wr-v2.1";
98 };
```

## Step 6 Update BOOT.BIN, image.ub

❖ 새로운 HW를 위한 BOOT.BIN과 image.ub를 다음과 같이 Update 한다.

```
$ cd ~/work/zynqmp_linux/petalinux/ultra96
$ petalinux-build -c virtual/boot-bin
$ 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
$ scp images/linux/{BOOT.BIN,image.ub} root@172.30.1.39:/media/card
```

## Step 7 Test 1

- ❖ 보드를 다시 boot하고 다음을 Test한다.
- ❖ output.mp4 Camera 동영상파일이고, 마지막 gst-launch-1.0 명령어는 Monitor에 Camera 영상이 보여지게 한다.

```
ultra96$ reboot
ultra96$ i2cdetect -l
i2c-3 i2c i2c-0-mux (chan_id 1) I2C adapter
i2c-1 i2c ZynqMP DP AUX I2C adapter
i2c-8 i2c i2c-0-mux (chan_id 6) I2C adapter
i2c-6 i2c i2c-0-mux (chan_id 4) I2C adapter
i2c-4 i2c i2c-0-mux (chan_id 2) I2C adapter
i2c-2 i2c i2c-0-mux (chan_id 0) I2C adapter
i2c-0 i2c Cadence I2C at ff030000 I2C adapter
i2c-9 i2c i2c-0-mux (chan_id 7) I2C adapter
i2c-7 i2c i2c-0-mux (chan_id 5) I2C adapter
i2c-5 i2c i2c-0-mux (chan_id 3) I2C adapter
```

## Step 7 Test 2

```
ultra96$ i2cdetect -y -r 3
  0 1 2 3 4 5 6 7 8 9 a b c d e f
00:  --- --
10:  --- --
20:  --- --
30:  --- -- UU ---
40:  --- --
50:  --- --
60:  --- --
70:  --- -- UU ---
ultra96$ ls /dev/media*
/dev/media0
ultra96$ media-ctl -d /dev/media0 -p
Device topology
- entity 1: vcap_mipi output 0 (1 pad, 1 link)
    type Node subtype V4L flags 0
    device node name /dev/video2
pad0: Sink
```

## Step 7 Test 3

```
<- "80000000.mipi_csi2_rx_subsystem":0 [ENABLED]

- entity 5: ov5640 3-003c (1 pad, 1 link)
  type V4L2 subdev subtype Sensor flags 0
  device node name /dev/v4l-subdev0
  pad0: Source
    [fmt:JPEG_1X8/640x480@1/30 field:none colorspace:jpeg xfer:srgb ycbr:601 quantization:full-range]
  -> "80000000.mipi_csi2_rx_subsystem":1 [ENABLED]

- entity 7: 80000000.mipi_csi2_rx_subsystem (2 pads, 2 links)
  type V4L2 subdev subtype Unknown flags 0
  device node name /dev/v4l-subdev1
  pad0: Source
    [fmt:UYVY8_1X16/1920x1080 field:none colorspace:srgb]
  -> "vcap_mipi output 0":0 [ENABLED]
  pad1: Sink
    [fmt:UYVY8_1X16/1920x1080 field:none colorspace:srgb]
  <- "ov5640 3-003c":0 [ENABLED]
```

## Step 7 Test 4

```
ultra96$ media-ctl -d /dev/media0 -V '"ov5640 3-003c":0 [fmt:UYVY/1920x1080@1/30 field:none]'
```

```
ultra96$ media-ctl -d /dev/media0 -p
```

```
- entity 5: ov5640 3-003c (1 pad, 1 link)
```

```
    type V4L2 subdev subtype Sensor flags 0
```

```
    device node name /dev/v4l-subdev0
```

```
pad0: Source
```

```
    [fmt:UYVY8_1X16/1920x1080@1/30 field:none colorspace:srgb xfer:srgb ycbcr:601 quantization:full-range]
```

```
-> "80000000.mipi_csi2_rx_subsystem":1 [ENABLED]
```

```
ultra96$ gst-launch-1.0 -v v4l2src device=/dev/video2 num-buffers=15 ! capsfilter caps='video/x-raw,width=1920,height=1080,format=YUY2' ! fpsdisplaysink video-sink='filesink location=/run/out.yuv'
```

```
$ scp root@172.30.1.39:/run/out.yuv .
```

```
$ ffmpeg -f rawvideo -vcodec rawvideo -s 1920x1080 -r 15 -pix_fmt yuyv422 -i out.yuv -c:v libx264 -preset ultrafast -qp 0 output.mp4
```

## Step 7 Test 5

```
ultra96$ modetest -D fd4a0000.zynqmp-display
```

Encoders:

id	crtc	type	possible crtcs	possible clones
38	37	TMDS	0x00000001	0x00000000

Connectors:

id	encoder	status	name	size (mm)	modes	encoders
39	38	connected	DP-1	510x290	27 38	

modes:

name	refresh (Hz)	hdisp	hss	hse	htot	vdisp	vss	vse	vtot	flags
1920x1080	60	1920	2068	2112	2200	1080	1116	1121	1125	148500 flags: phsync, pvsync; type: preferred, driver

CRTCs:

id	fb	pos	size
37	71	(0,0)	(1920x1080)

1920x1080 60 1920 2068 2112 2200 1080 1116 1121 1125 148500 flags: phsync, pvsync; type: preferred, driver

props:

## Step 7 Test 6

Planes:

id crtc fb CRTC x,y x,y gamma size possible crtcs

35 0 0 0,0 0,0 0 0x00000001

formats: VYUY UYVY YUYV YVYU YU16 YV16 YU24 YV24 NV16 NV61 GREY Y10 BG24 RG24 XB24 XR24 XB30 XR30 YU12 YV12 NV12 NV21 XV15 XV20

props:

7 type:

flags: immutable enum

enums: Overlay=0 Primary=1 Cursor=2

value: 0

36 37 71 0,0 0,0 0 0x00000001

formats: AB24 AR24 RA24 BA24 BG24 RG24 RA15 BA15 RA12 BA12 RG16 BG16

props:

7 type:

flags: immutable enum

enums: Overlay=0 Primary=1 Cursor=2



## Step 7 Test 7

value: 1

28 alpha:

flags: range

values: 0 255

value: 255

29 g\_alpha\_en:

flags: range

values: 0 1

value: 1

```
ultra96$ modetest -D fd4a0000.zynqmp-display -w 36:g_alpha_en:0
```

```
ultra96$ gst-launch-1.0 -v v4l2src device=/dev/video2 io-mode=dmabuf ! capsfilter caps=video/x-raw,width=1920,height=1080,format=YUY2 ! fpsdisplaysink fps-update-intervalvideo-sink=1000 signal-fps-measurements=true text-overlay=false sync=false video-sink='kmssink bus-id=fd4a0000.zynqmp-display'
```