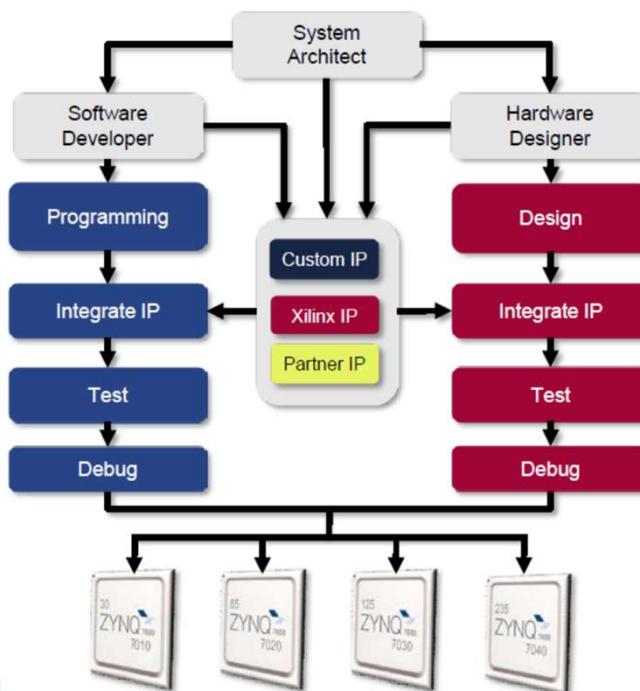


Embedded Linux on Zynq



김현옥 (hokim1972@naver.com)

- Zynq = Dual core ARM (PS: Processing System) + FPGA (PL: Programmable Logic)
- Processor concentrated chip
 - Processor runs with minimal(?) PL programming.
 - Less flexibility.
 - More efficient parallel hardware and software development.



- **Bare-Metal** without OS

Possible to write fast programs.

Difficult to realize complex systems with components like webserver, FTP server, USB support and so on.

SDK Provides ARM Toolchain.

- **Real-Time OS**

For applications with strict time constraints.

Most are commercial.

- **Linux**

Greatest flexibility.

Many drivers, libraries and software package.



- **Vivado GUI, CUI(2015.4)**

- Design hardware

- Generate bit/hdf files

- **HSI Hardware Software Interface**

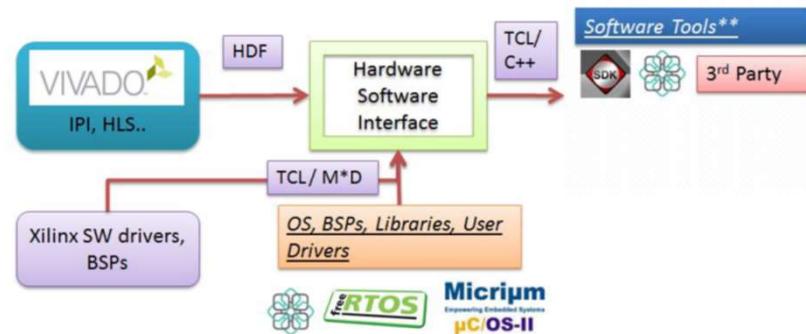
- Scalable framework enabled SW tool integration
with Vivado

- Compile FSBL / Generate DT

- **Ubuntu (16.04 LTS)**

- (Host) Compile u-boot, kernel and DT

- (Target) Compile drivers / applications

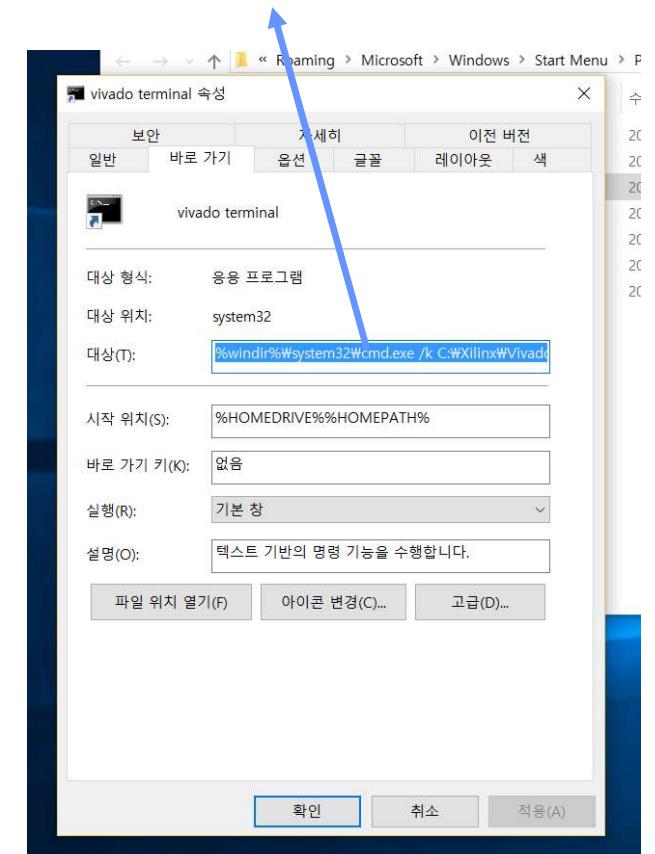
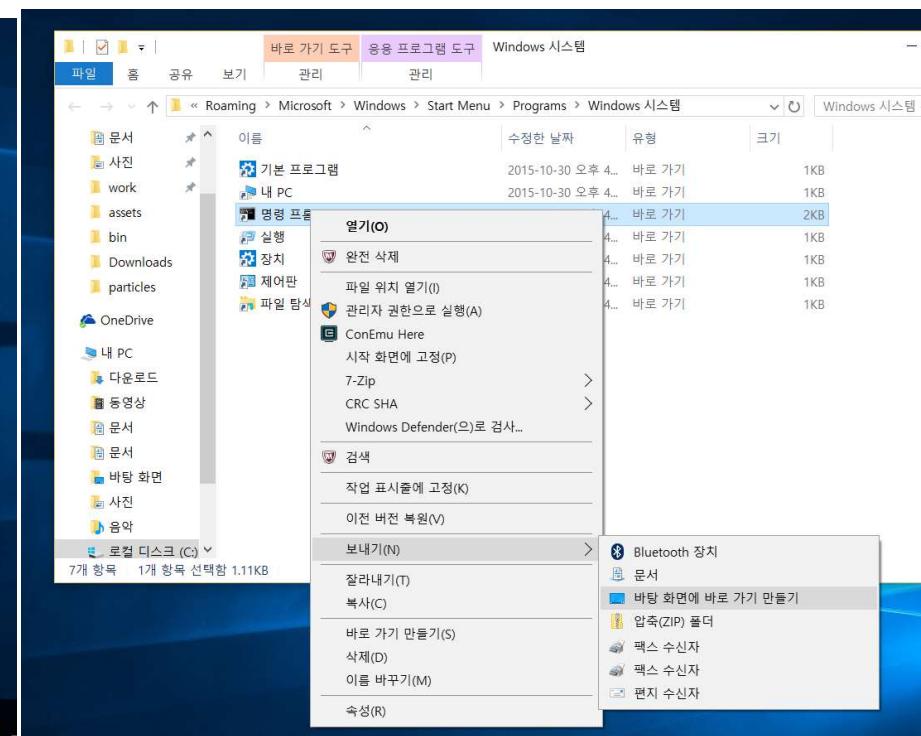
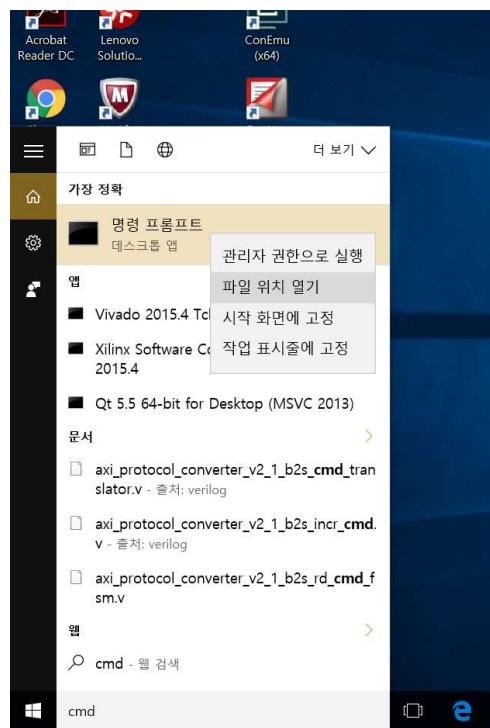


Development Environment Set-up

■ Windows 10

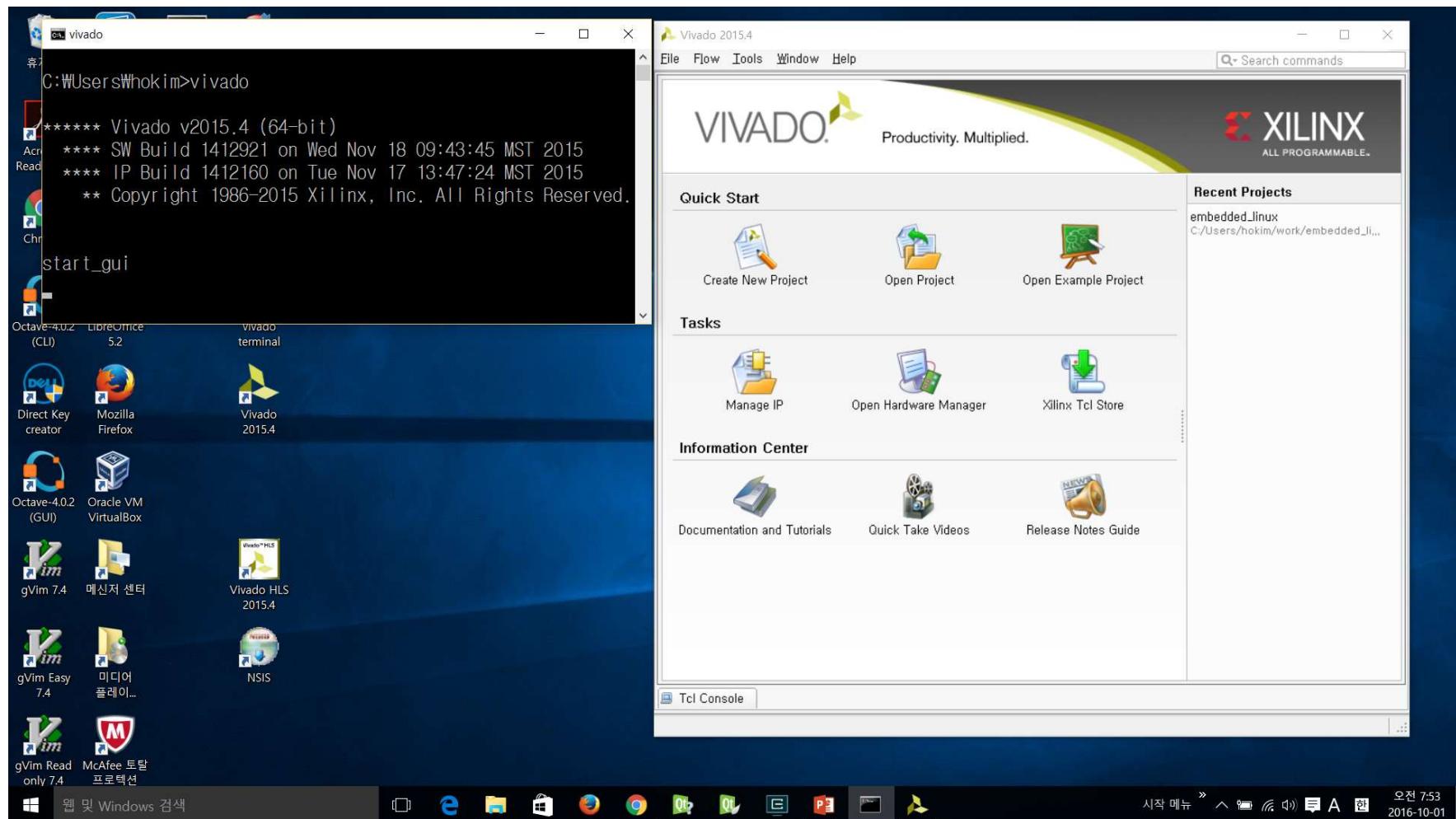
Install Vivado 2015.4 w/ SDK.
cmd terminal for Vivado.

```
%windir%\system32\cmd.exe /k C:\Xilinx\Vivado\2015.4\settings64.bat
```

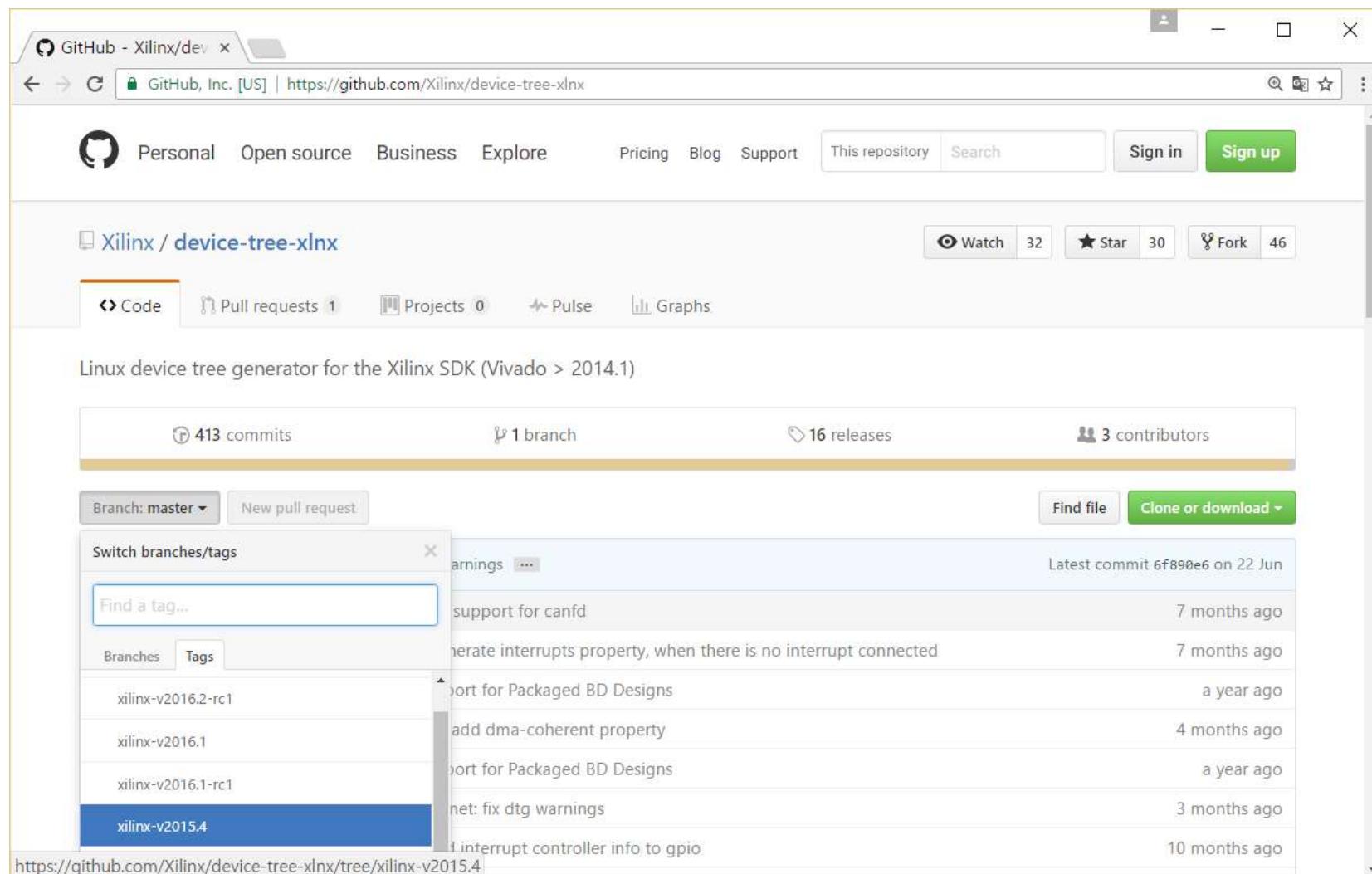


Development Environment Set-up

■ cmd terminal for Vivado.



■ Device Tree Source



Xilinx / device-tree-xlnx

Code Pull requests 1 Projects 0 Pulse Graphs

Linux device tree generator for the Xilinx SDK (Vivado > 2014.1)

413 commits 1 branch 16 releases 3 contributors

Branch: master New pull request Find file Clone or download

Switch branches/tags

Find a tag... Branches Tags

- xilinx-v2016.2-rc1
- xilinx-v2016.1
- xilinx-v2016.1-rc1
- xilinx-v2015.4

warnings ... Latest commit 6f890e6 on 22 Jun

support for canfd 7 months ago

generate interrupts property, when there is no interrupt connected 7 months ago

port for Packaged BD Designs a year ago

add dma-coherent property 4 months ago

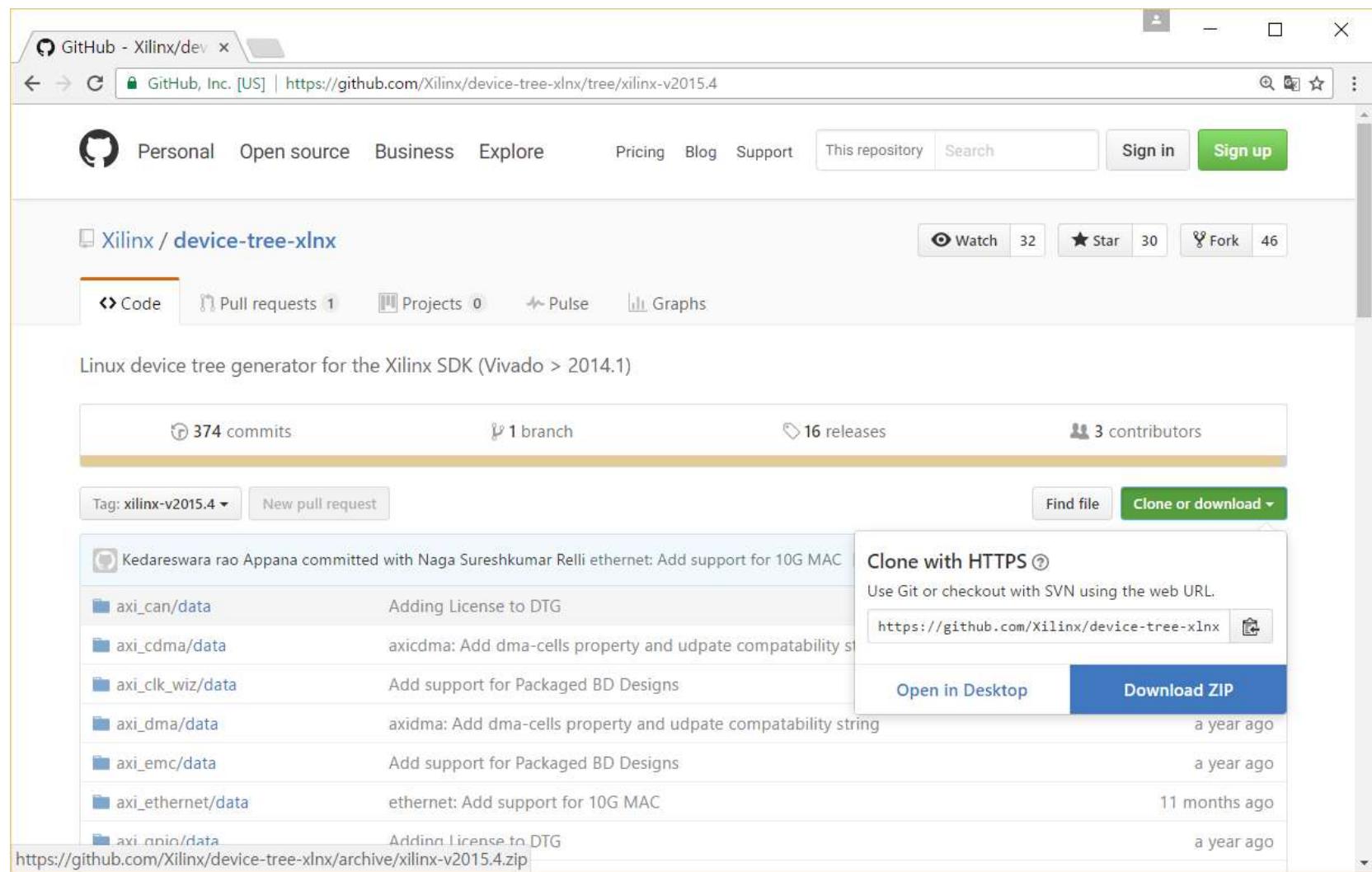
port for Packaged BD Designs a year ago

net: fix dtg warnings 3 months ago

int interrupt controller info to gpio 10 months ago

<https://github.com/Xilinx/device-tree-xlnx/tree/xilinx-v2015.4>

■ Device Tree Source



GitHub - Xilinx/dev

GitHub, Inc. [US] | https://github.com/Xilinx/device-tree-xlnx/tree/xilinx-v2015.4

Personal Open source Business Explore Pricing Blog Support This repository Search Sign in Sign up

Xilinx / device-tree-xlnx

Watch 32 Star 30 Fork 46

Code Pull requests 1 Projects 0 Pulse Graphs

Linux device tree generator for the Xilinx SDK (Vivado > 2014.1)

374 commits 1 branch 16 releases 3 contributors

Tag: xilinx-v2015.4 ▾ New pull request Find file Clone or download ▾

Kedareswara rao Appana committed with Naga Sureshkumar Relli ethernet: Add support for 10G MAC

axi_can/data Adding License to DTG

axi_cdma/data axicdma: Add dma-cells property and update compatibility string

axi_clk_wiz/data Add support for Packaged BD Designs

axi_dma/data axidma: Add dma-cells property and update compatibility string

axi_emc/data Add support for Packaged BD Designs

axi_ethernet/data ethernet: Add support for 10G MAC

axi_gpio/data Adding License to DTG

Clone with HTTPS ⓘ

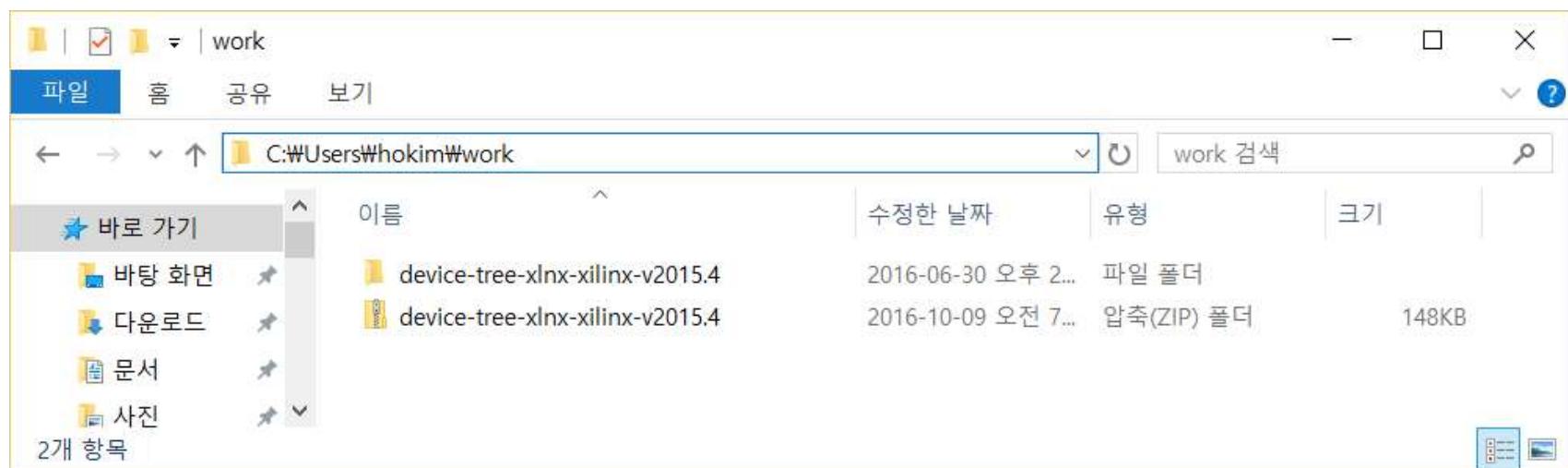
Use Git or checkout with SVN using the web URL.

<https://github.com/Xilinx/device-tree-xlnx>

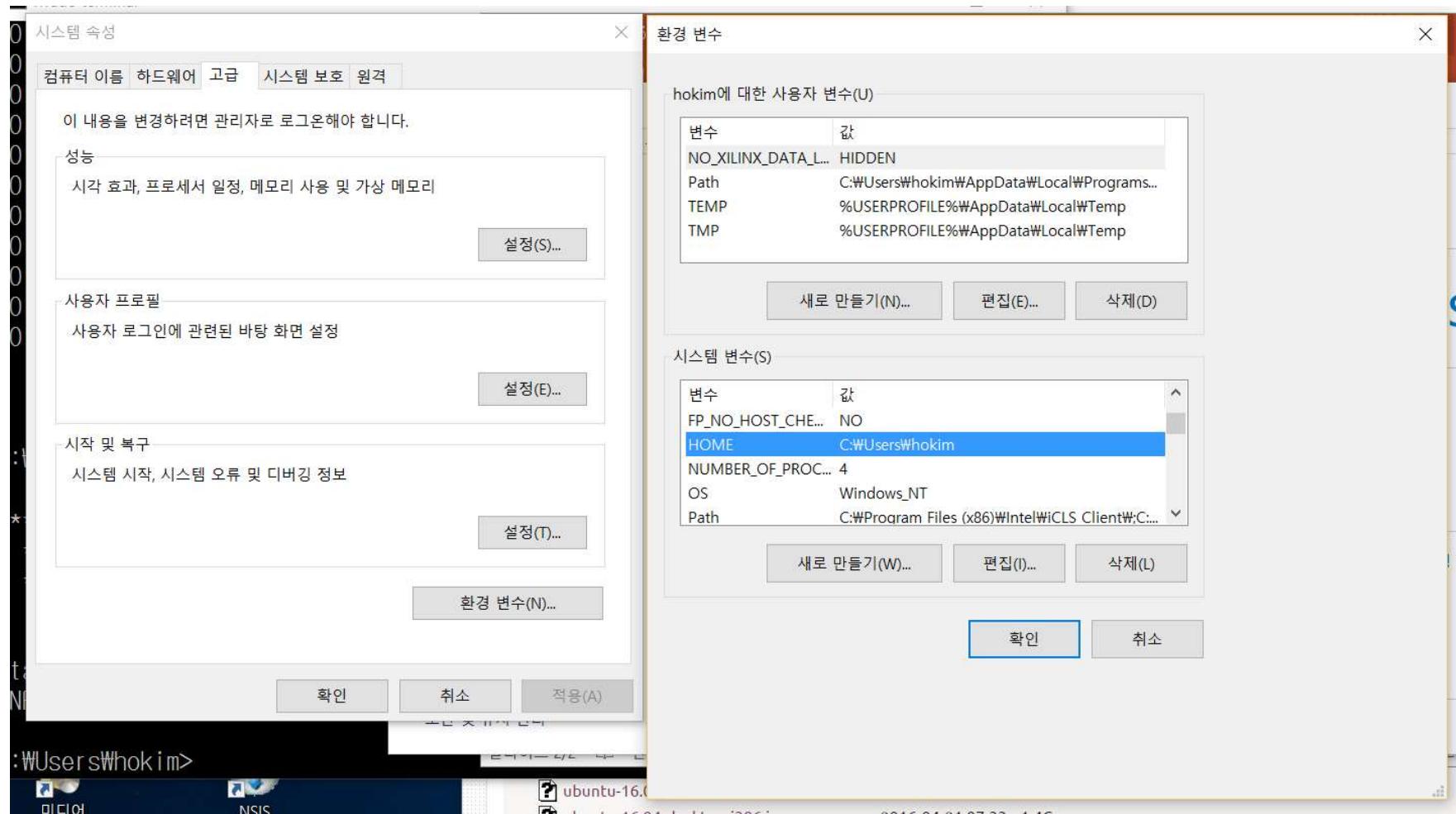
Open in Desktop Download ZIP

https://github.com/Xilinx/device-tree-xlnx/archive/xilinx-v2015.4.zip

■ Device Tree Source



■ Environment Variable(HOME)



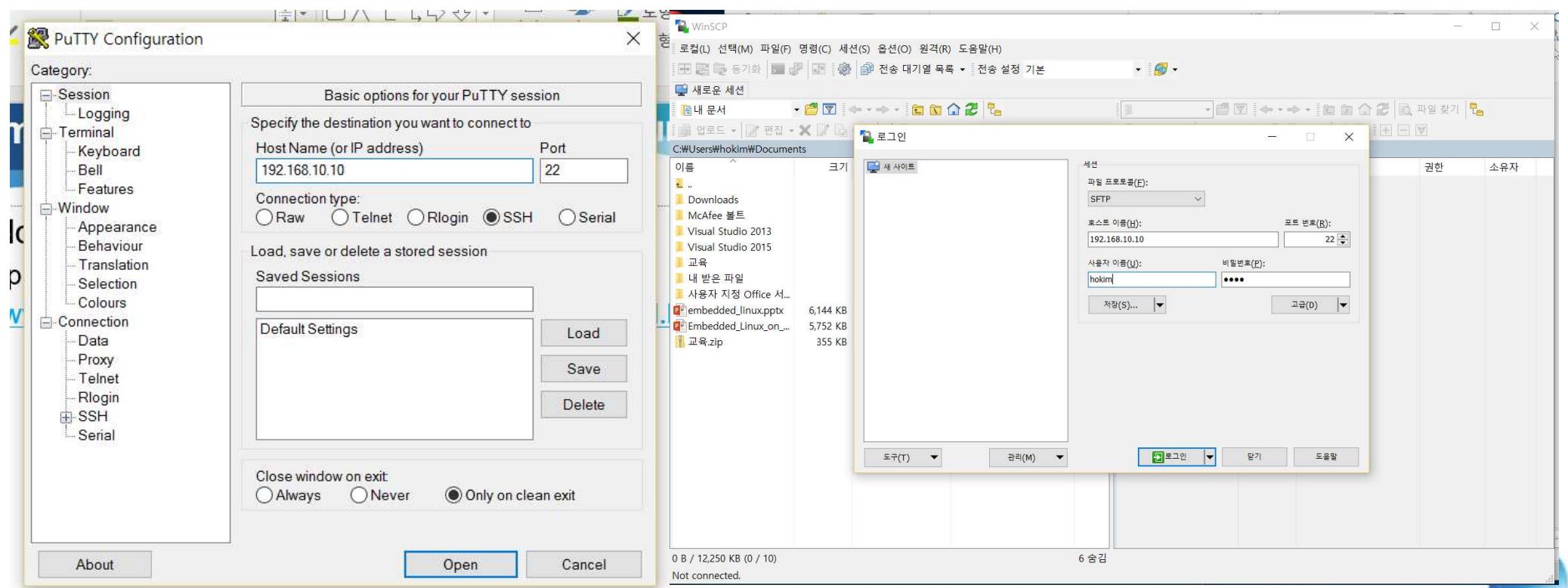
■ Windows ↔ Ubuntu

Install putty.exe for remote login shell.

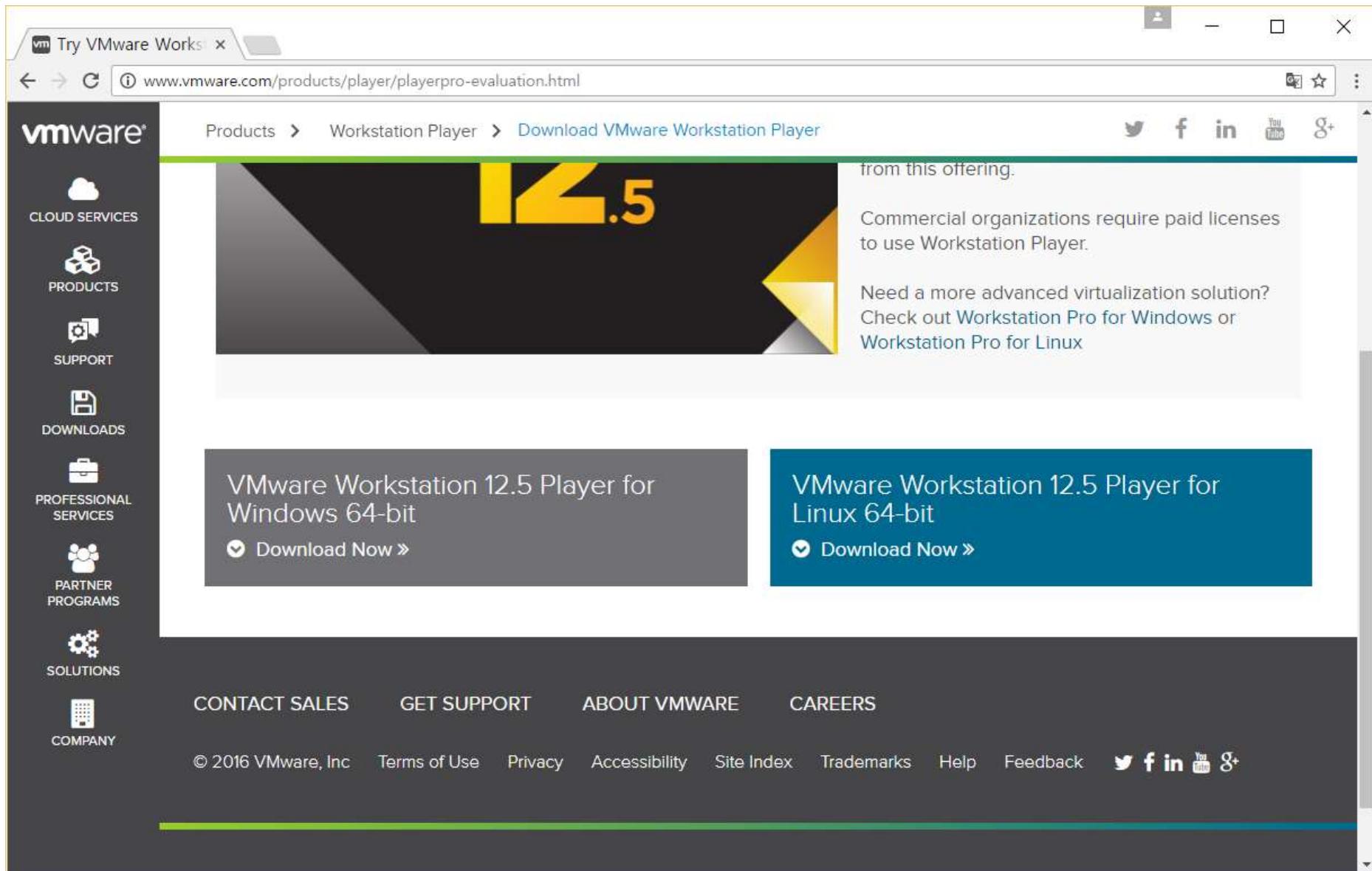
(<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>)

Install WinSCP for file transfer.

(<https://winscp.net/eng/download.php>)

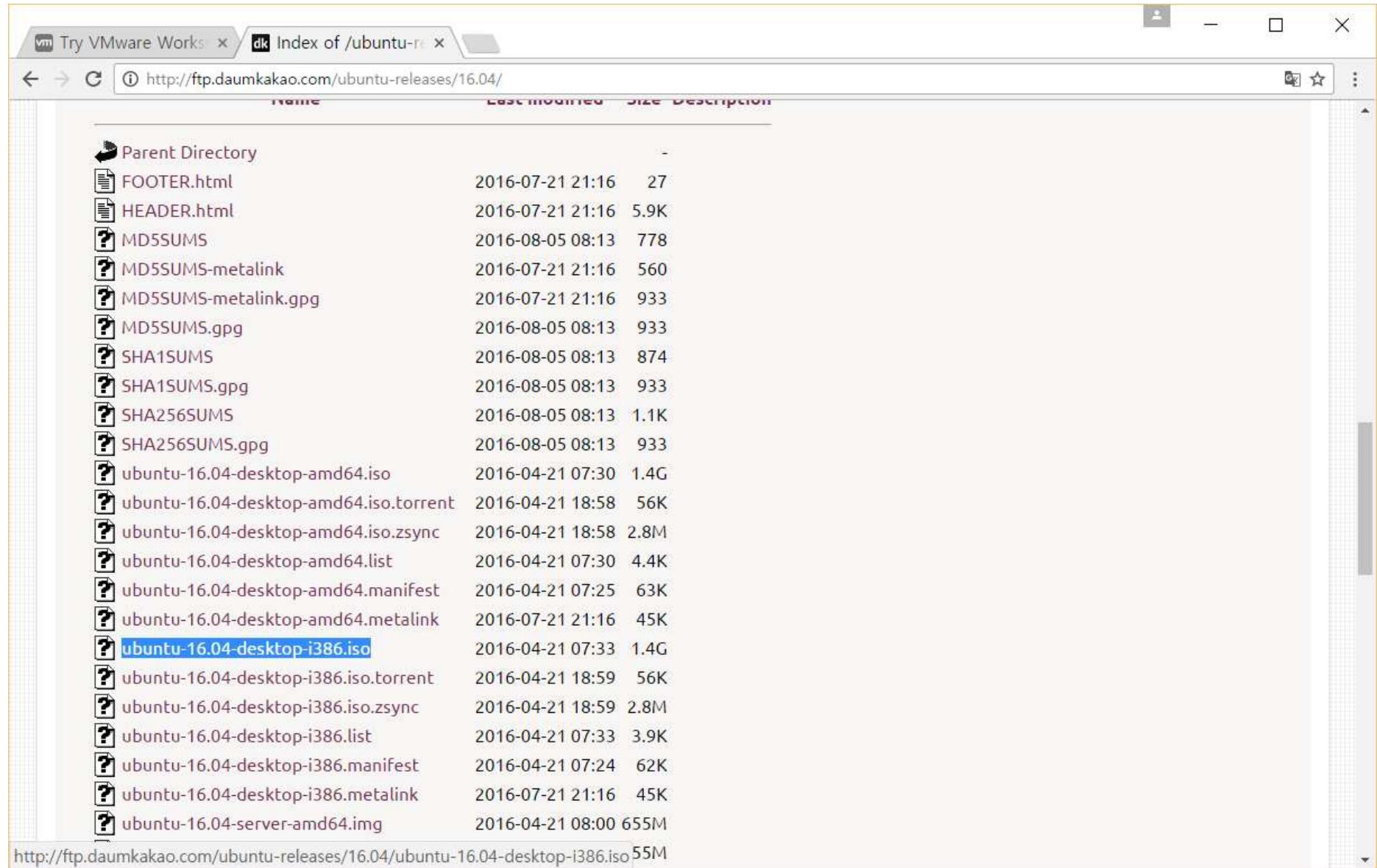


■ Ubuntu on VMware-player



The screenshot shows a web browser window displaying the VMware website at www.vmware.com/products/player/playerpro-evaluation.html. The page is titled "Download VMware Workstation Player". On the left, there's a sidebar with links to Cloud Services, Products, Support, Downloads, Professional Services, Partner Programs, and Solutions. The main content area features a large "12.5" logo and text about the offering. It highlights that commercial organizations require paid licenses. Below this, two download options are shown: "VMware Workstation 12.5 Player for Windows 64-bit" and "VMware Workstation 12.5 Player for Linux 64-bit", each with a "Download Now" button. At the bottom, there are links for Contact Sales, Get Support, About VMware, and Careers, along with standard footer links for Terms of Use, Privacy, Accessibility, Site Index, Trademarks, Help, Feedback, and social media icons for Twitter, Facebook, LinkedIn, YouTube, and Google+.

■ Ubuntu on VMware-player

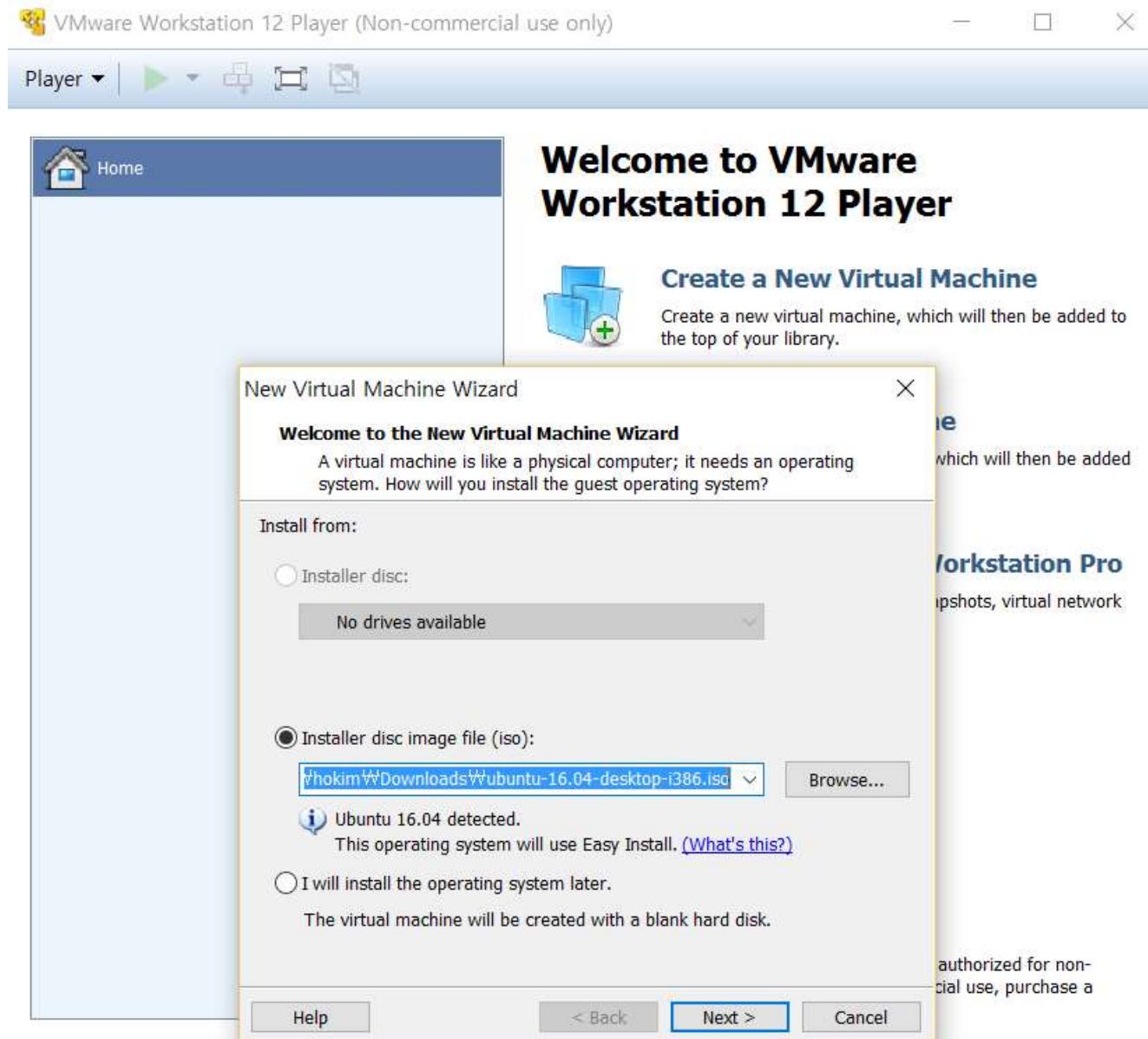


The screenshot shows a web browser window displaying a file listing from an FTP site. The address bar shows the URL: <http://ftp.daumkakao.com/ubuntu-releases/16.04/>. The browser title bar includes tabs for "Try VMware Works" and "Index of /ubuntu-releases/16.04/". The main content area is a table listing files and their details:

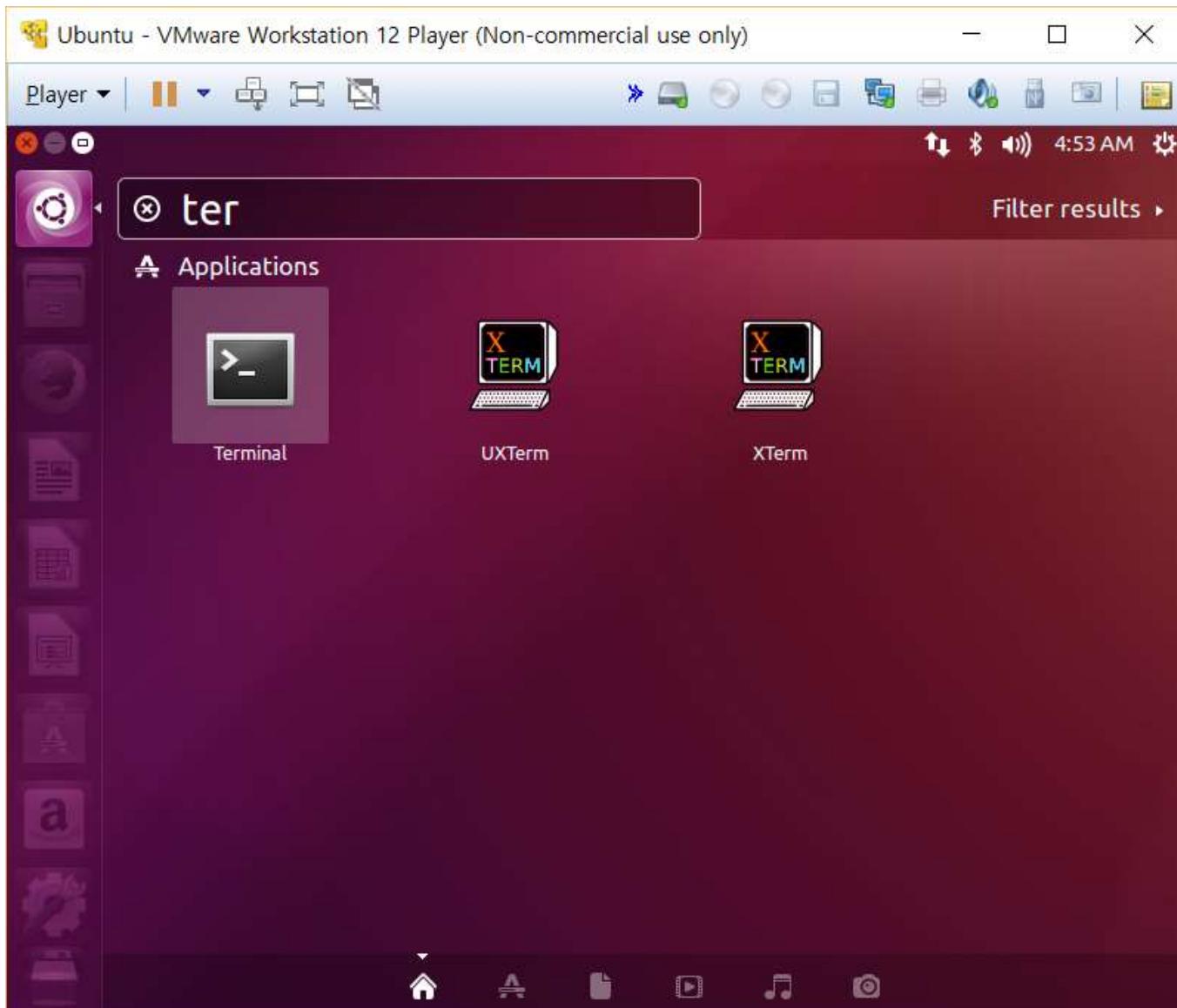
Name	Last Modified	Size	Description
Parent Directory		-	
FOOTER.html	2016-07-21 21:16	27	
HEADER.html	2016-07-21 21:16	5.9K	
MD5SUMS	2016-08-05 08:13	778	
MD5SUMS-metalink	2016-07-21 21:16	560	
MD5SUMS-metalink.gpg	2016-07-21 21:16	933	
MD5SUMS.gpg	2016-08-05 08:13	933	
SHA1SUMS	2016-08-05 08:13	874	
SHA1SUMS.gpg	2016-08-05 08:13	933	
SHA256SUMS	2016-08-05 08:13	1.1K	
SHA256SUMS.gpg	2016-08-05 08:13	933	
ubuntu-16.04-desktop-amd64.iso	2016-04-21 07:30	1.4G	
ubuntu-16.04-desktop-amd64.iso.torrent	2016-04-21 18:58	56K	
ubuntu-16.04-desktop-amd64.iso.zsync	2016-04-21 18:58	2.8M	
ubuntu-16.04-desktop-amd64.list	2016-04-21 07:30	4.4K	
ubuntu-16.04-desktop-amd64.manifest	2016-04-21 07:25	63K	
ubuntu-16.04-desktop-amd64.metalink	2016-07-21 21:16	45K	
ubuntu-16.04-desktop-i386.iso	2016-04-21 07:33	1.4G	
ubuntu-16.04-desktop-i386.iso.torrent	2016-04-21 18:59	56K	
ubuntu-16.04-desktop-i386.iso.zsync	2016-04-21 18:59	2.8M	
ubuntu-16.04-desktop-i386.list	2016-04-21 07:33	3.9K	
ubuntu-16.04-desktop-i386.manifest	2016-04-21 07:24	62K	
ubuntu-16.04-desktop-i386.metalink	2016-07-21 21:16	45K	
ubuntu-16.04-server-amd64.img	2016-04-21 08:00	655M	

The bottom status bar of the browser shows the URL: <http://ftp.daumkakao.com/ubuntu-releases/16.04/ubuntu-16.04-desktop-i386.iso> 55M.

■ Ubuntu on VMware-player

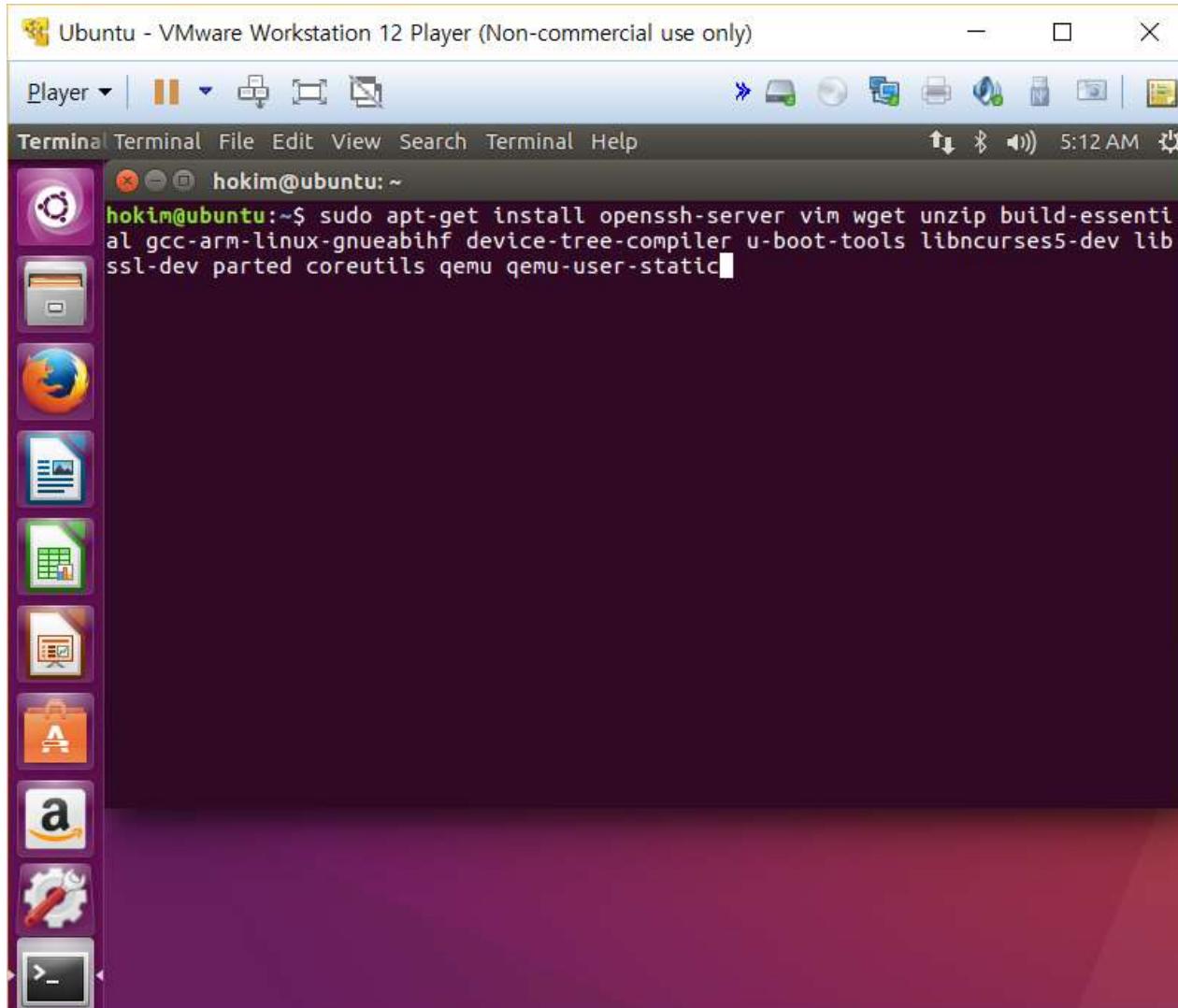


■ Ubuntu on VMware-player



■ Ubuntu on VMware-player

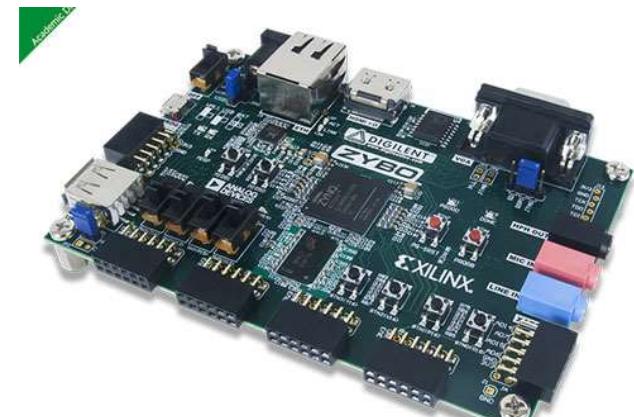
```
$ sudo apt-get install openssh-server vim wget unzip build-essential gcc-arm-linux-gnueabihf device-tree-compiler u-boot-tools libncurses5-dev libssl-dev parted coreutils qemu qemu-user-static
```



■ Target Board Zybo(Zynq-7010)

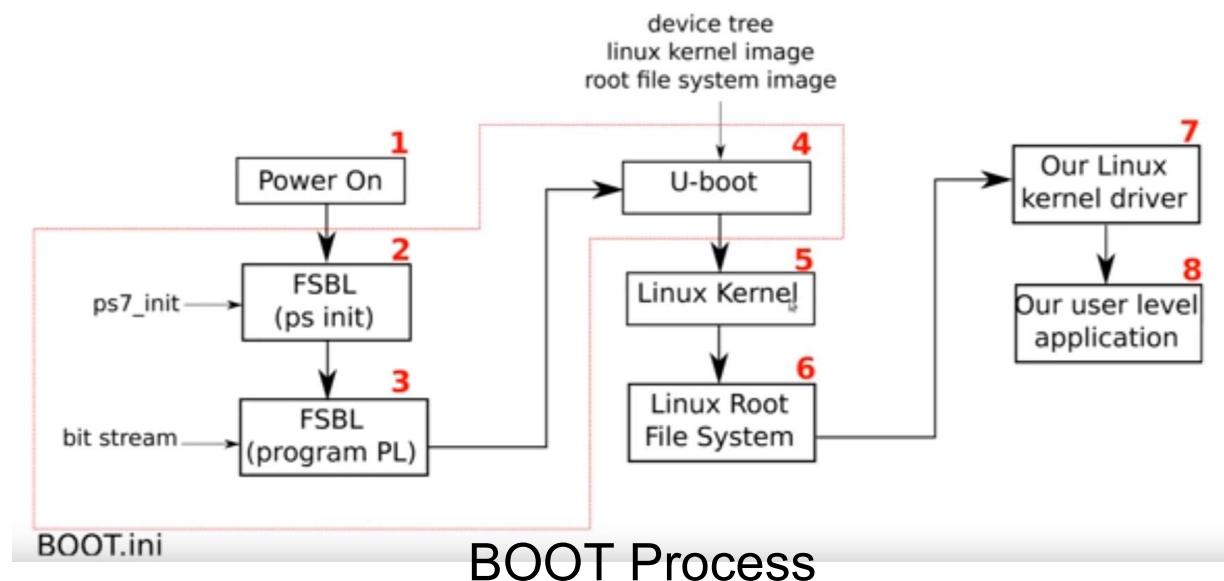
Gigabit Ethernet, USB, SD, UART

28,000 logic cells, 240 KB BRAM



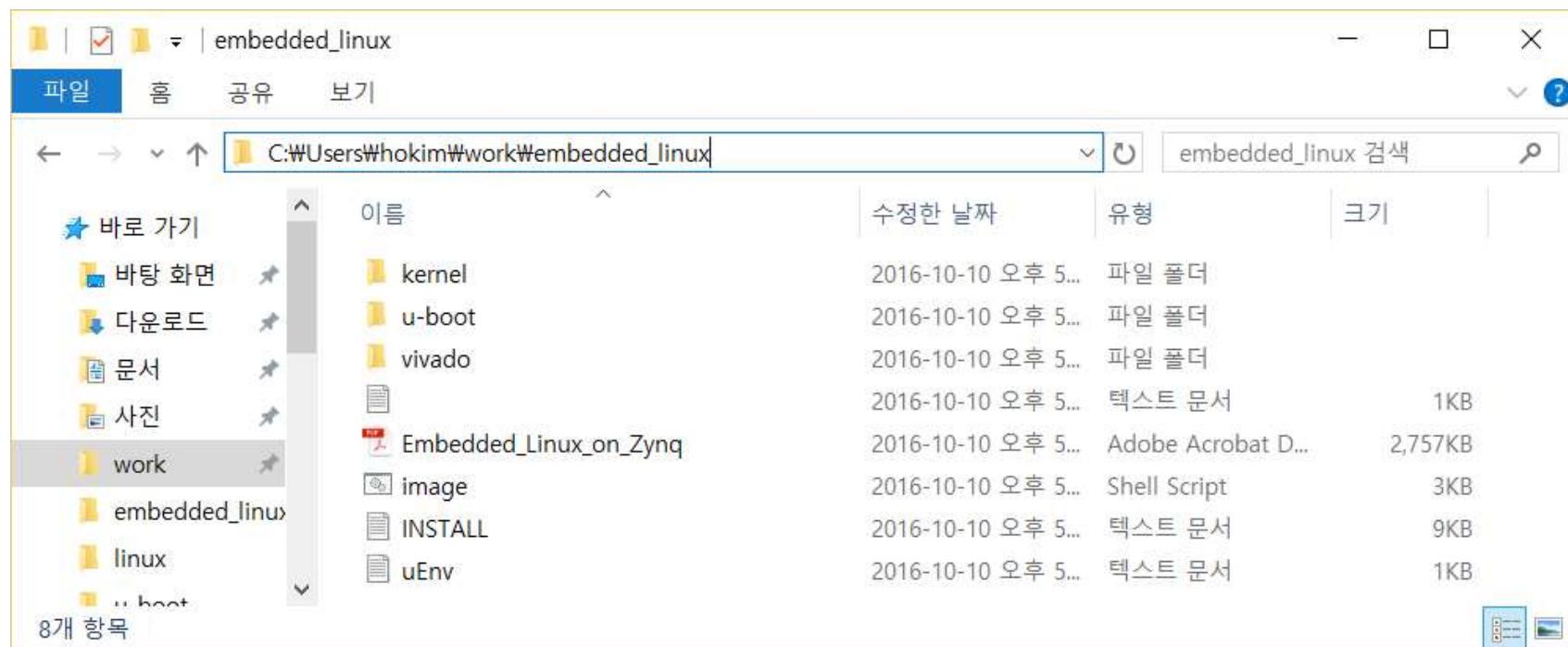
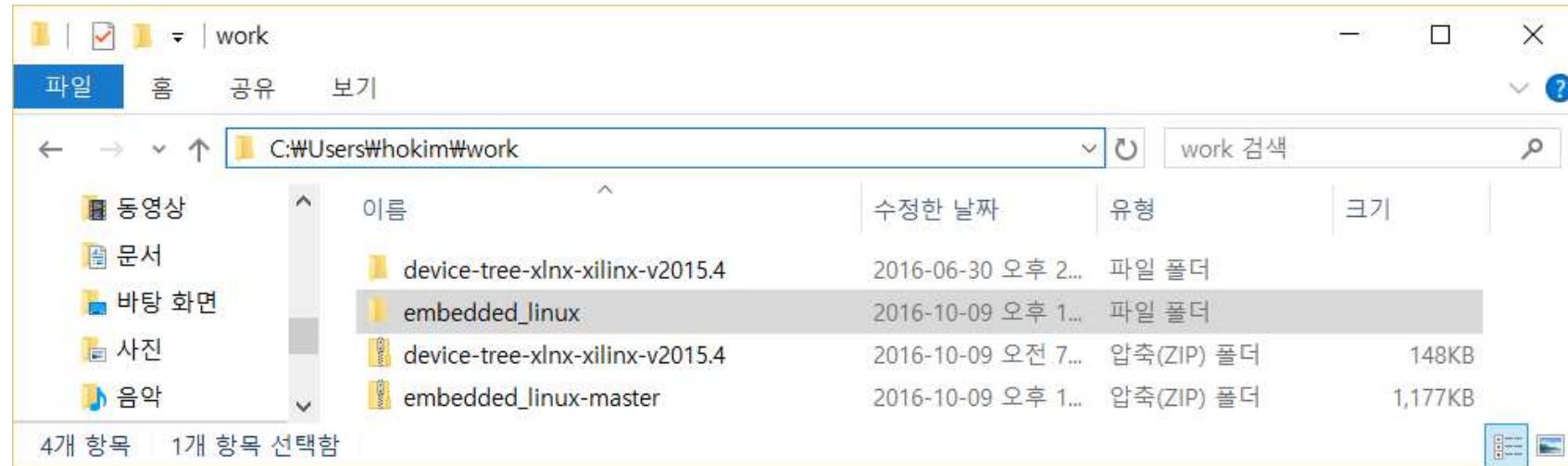
■ Development Process

1. bit stream
2. FSBL
3. DT Generation
4. u-boot
5. DT Compile
6. linux kernel image
7. root file system image
8. BOOT.bin(fsbl + bit + u-boot)
9. Update BOOT.bin / DT / kernel
10. (user kernel driver)/user application



■ Source code

https://github.com/inipro/embedded_linux



Design of Zynq PS Hardware

■ Vivado Design

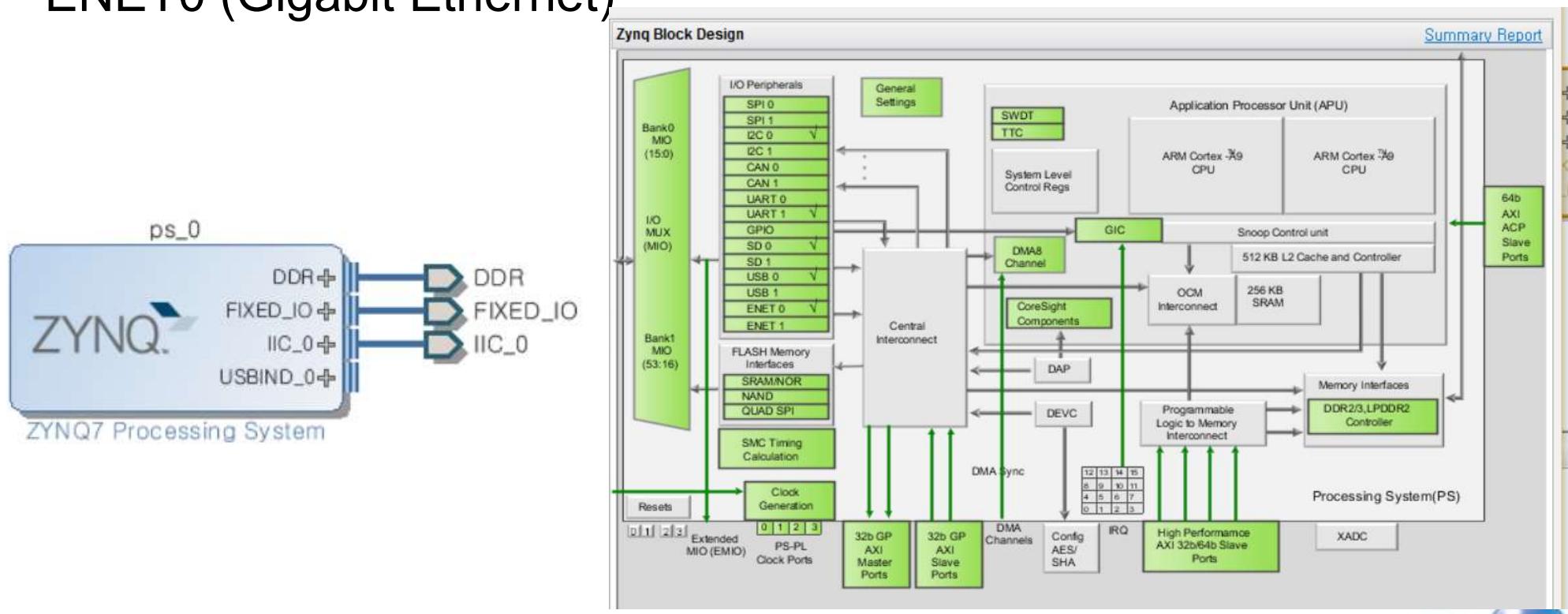
SD0 (boot files, root file system)

UART1 (console)

I2C0 (eeprom w/ Ethernet mac address; EMIO)

USB0 (usb host)

ENET0 (Gigabit Ethernet)



embedded_linux.tcl

```
set project_name embedded_linux
set part_name xc7z010clg400-1
#set ip_dir ip
set bd_path $project_name/$project_name.srcs/sources_1/bd/system
file delete -force $project_name
create_project -part $part_name $project_name $project_name
#set_property ip_repo_paths $ip_dir [current_project]
#update_ip_catalog
create_bd_design system

create_bd_cell -type ip -vlnv xilinx.com:ip:processing_system7:5.5 ps_0
source embedded_linux_preset.tcl
set_property -dict [apply_preset IPINST] [get_bd_cells ps_0]
apply_bd_automation -rule xilinx.com:bd_rule:processing_system7 -config {
    make_external {FIXED_IO, DDR}
} [get_bd_cells ps_0]
create_bd_intf_port -mode Master -vlnv xilinx.com:interface:iic_rtl:1.0 IIC_0
connect_bd_intf_net [get_bd_intf_pins ps_0/IIC_0] [get_bd_intf_ports IIC_0]

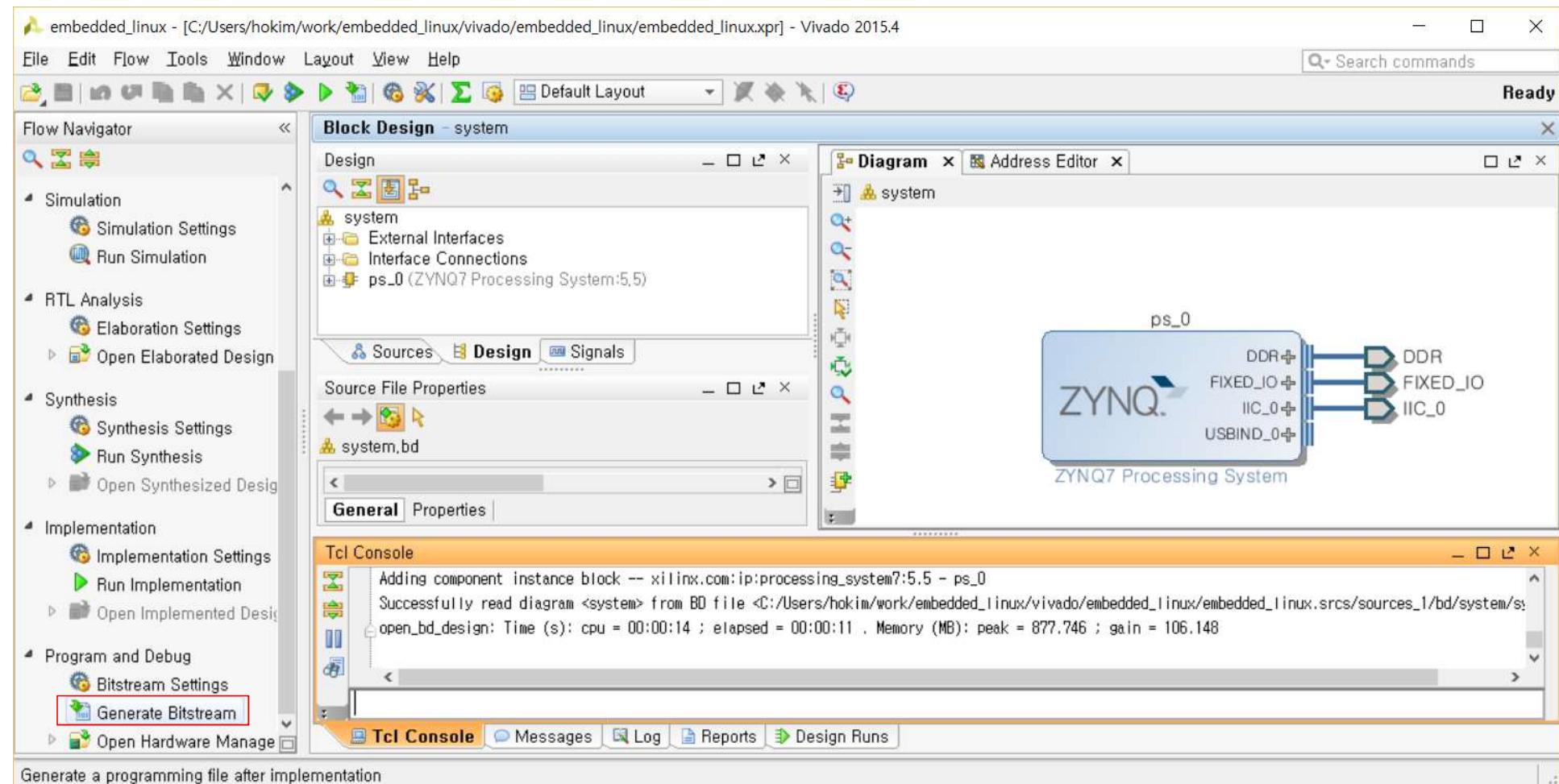
generate_target all [get_files $bd_path/system.bd]
make_wrapper -files [get_files $bd_path/system.bd] -top
add_files -norecurse $bd_path/hdl/system_wrapper.v
add_files -norecurse -fileset constrs_1 zybo.xdc
set_property verilog_define {TOOL_VIVADO} [current_fileset]
close_project
```

In cmd window for Vivado

```
C:\ cd C:\Users\hokim\work\embedded_linux\vivado
C:\ vivado -nolog -nojournal -mode batch -source embedded_linux.tcl
```

output : embedded_linux\embedded_linux.xpr...

Bit Generation



hwdef.tcl

```
# vivado -nolog -nojournal -mode batch -source hwdef.tcl

set project_name embedded_linux

open_project $project_name/$project_name.xpr

if {[get_property PROGRESS [get_runs synth_1]] != "100%"} {
    launch_runs synth_1
    wait_on_run synth_1
}

file delete -force $project_name/$project_name.hwdef

write_hwdef -force -file $project_name/$project_name.hwdef

close_project
```

```
C:\ vivado -nolog -nojournal -mode batch -source hwdef.tcl
```

```
output : embedded_linux\embedded_linux.hwdef
```



fsbl.tcl

```
# hsi -nolog -nojournal -mode batch -source fsbl.tcl

set project_name embedded_linux

set hard_path $project_name/$project_name.hard
set fsbl_path $project_name/$project_name.fsbl

file delete -force $hard_path $fsbl_path

file mkdir $hard_path
file copy -force $project_name/$project_name.hwdef $hard_path/$project_name.hdf

open_hw_design $hard_path/$project_name.hdf
create_sw_design -proc ps7_cortexa9_0 -os standalone fsbl

add_library xilffs
add_library xilrsa

generate_app -proc ps7_cortexa9_0 -app zynq_fsbl -dir $fsbl_path -compile

close_hw_design [current_hw_design]
```

C:\ hsi -nolog -nojournal -mode batch -source fsbl.tcl

output : embedded_linux\embedded_linux.fsbl\executable.elf



devicetree.tcl

```
# hsi -nolog -nojournal -mode batch -source devicetree.tcl

set project_name embedded_linux

set boot_args {console=ttyPS0,115200 root=/dev/mmcblk0p2 ro rootfstype=ext4 earlyprintk rootwait}

set hard_path $project_name/$project_name.hard
set tree_path $project_name/$project_name.tree

file delete -force $hard_path $tree_path

file mkdir $hard_path
file copy -force $project_name/$project_name.hwdef $hard_path/$project_name.hdf

set_repo_path $::env(HOME)/work/device-tree-xlnx-xilinx-v2015.4

open_hw_design $hard_path/$project_name.hdf
create_sw_design -proc ps7_cortexa9_0 -os device_tree devicetree

set_property CONFIG.kernel_version {2015.4} [get_os]
set_property CONFIG.bootargs $boot_args [get_os]

generate_bsp -dir $tree_path

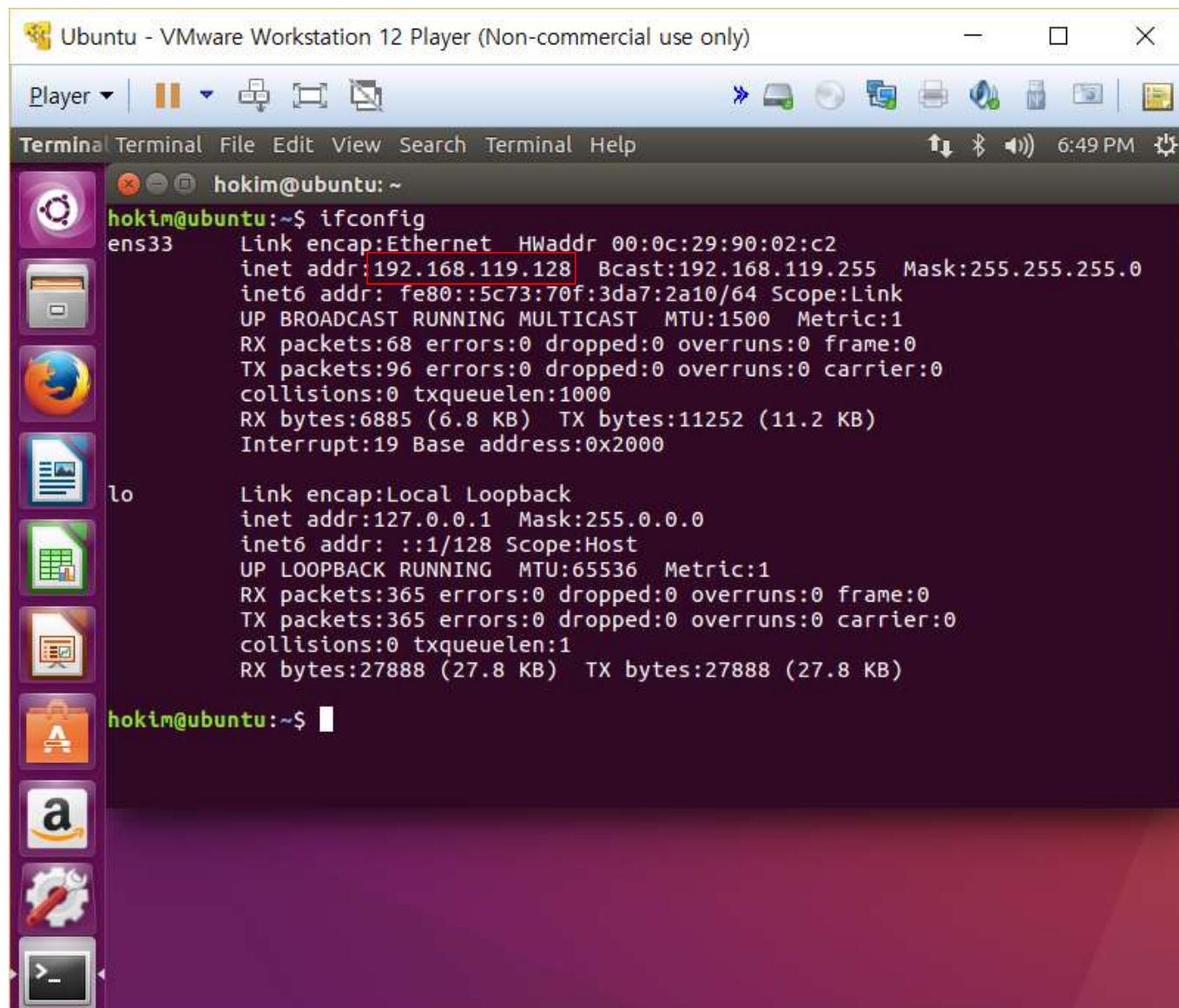
close_sw_design [current_sw_design]
close_hw_design [current_hw_design]
```

C:\ hsi -nolog -nojournal -mode batch -source devicetree.tcl

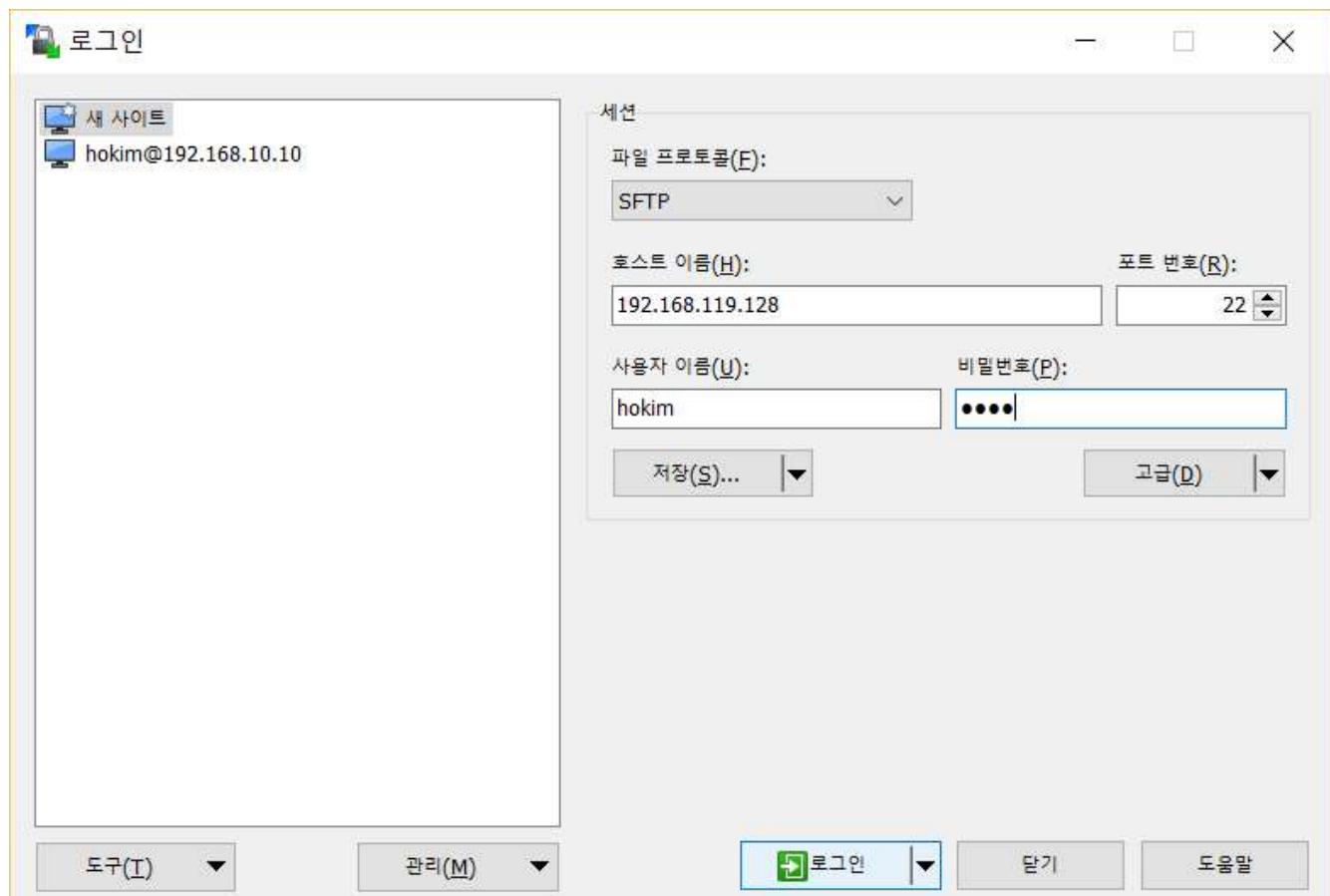
output : embedded_linux\embedded_linux.tree/*



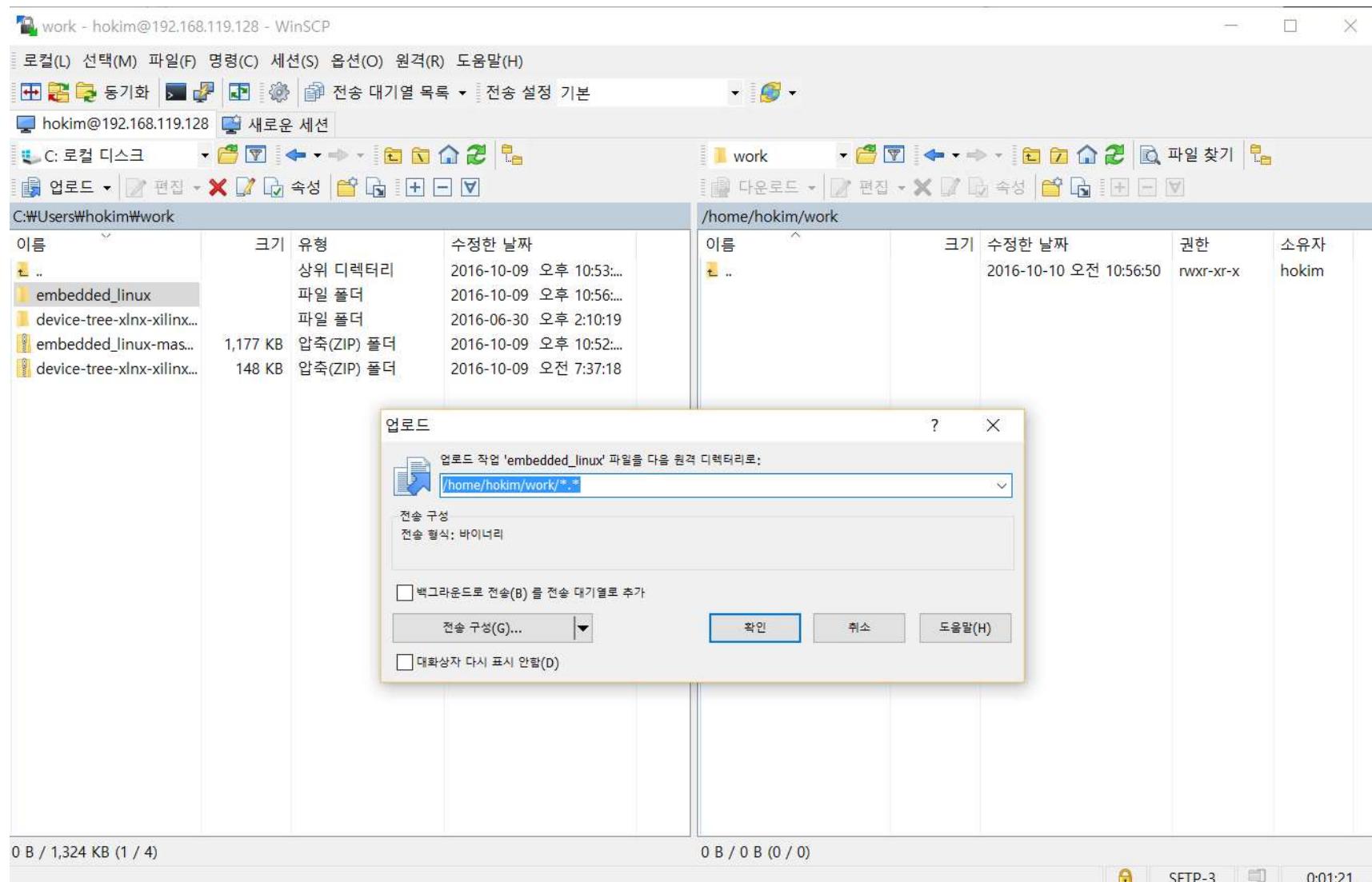
Ubuntu network IP



embedded_linux using WinSCP



embedded_linux using WinSCP



■ **Boot loader for linux**

Load linux kernel(ulImage, devicetree.dtb) from SD to DRAM.

Set ethernet using mac address from i2c eeprom.

Boot ulimage.



In Terminal of Ubuntu

Can use gedit instead of vi

- : deletion
- + : insert

```
$ cd ~/work/embedded_linux
$ mkdir -p dl
$ wget https://github.com/Xilinx/u-boot-xlnx/archive/xilinx-
v2015.4.tar.gz -O dl/u-boot-xlnx-xilinx-v2015.4.tar.gz
$ cd u-boot
$ tar xvzf ..../dl/u-boot-xlnx-xilinx-v2015.4.tar.gz
$ cd u-boot-xlnx-xilinx-v2015.4
$ nano board/xilinx/zynq/board.c
```



board/Xilinx/zynq/board.c

```
int board_late_init(void)
{
+ #if defined(CONFIG_ZYNQ_GEM_EEPROM_ADDR) && \
+ defined(CONFIG_ZYNQ_GEM_I2C_MAC_OFFSET)
+     unsigned char enetaddr[6];
+
+     if (eeprom_read(CONFIG_ZYNQ_GEM_EEPROM_ADDR,
+                     CONFIG_ZYNQ_GEM_I2C_MAC_OFFSET,
+                     enetaddr, ARRAY_SIZE(enetaddr)))
+         printf("I2C EEPROM MAC address read failed\n");
+     else
+         eth_setenv_enetaddr("ethaddr", enetaddr);
+ #endif
+
    switch ((zynq_slcr_get_boot_mode()) & ZYNQ_BM_MASK) {
    case ZYNQ_BM_QSPI:
```

```
$ nano common/main.c
```

common/main.c

```
if (cli_process_fdt(&s))
    cli_secure_boot_cmd(s);

+
+     setenv("fdt_high", "0x1E000000");
+     setenv("sdboot", "echo Importing environment from SD... && mmcinfo && fatload mmc 0 0x2000000 uEnv.txt &&
env import -t 0x2000000 ${filesize} && boot");

autoboot_command(s);
```

```
$ cp ..../zynq-zyboc.dts arch/arm/dts  
$ cp ..../zynq_zyboc.h include/configs  
$ cp ..../zynq_zyboc_defconfig configs
```

arch/arm/dts/zynq-zyboc.dts

```
/dts-v1/;  
#include "zynq-7000.dtsci"  
  
{  
    model = "Zynq ZYBOC Development Board";  
    compatible = "inipro,zynq-zyboc", "xlnx,zynq-7000";  
  
    aliases {  
        ethernet0 = &gem0;  
        serial0 = &uart1;  
    };  
  
    memory {  
        device_type = "memory";  
        reg = <0x0 0x20000000>;  
    };  
  
    chosen {  
        bootargs = "earlyprintk";  
        linux,stdout-path = &uart1;  
        stdout-path = &uart1;  
    };  
};
```

```
&clkc {  
    ps-clk-frequency = <50000000>;  
};  
  
&gem0 {  
    status = "okay";  
    phy-mode = "rgmii-id";  
    phy-handle = <&ethernet_phy>;  
  
    ethernet_phy: ethernet-phy@0 {  
        reg = <0>;  
    };  
  
&sdhci0 {  
    status = "okay";  
};  
  
&uart1 {  
    status = "okay";  
};
```

include/configs/zynq_zyboc.h

```
#ifndef __CONFIG_ZYNQ_ZYB0C_H
#define __CONFIG_ZYNQ_ZYB0C_H

#define CONFIG_SYS_SDRAM_SIZE (512 * 1024 * 1024)

#define CONFIG_ZYNQ_SERIAL_UART1
#define CONFIG_ZYNQ_GEM0
#define CONFIG_ZYNQ_GEM_PHY_ADDR0 0

#define CONFIG_SYS_NO_FLASH

#define CONFIG_ZYNQ_SDHCI0
#define CONFIG_ZYNQ_I2C0
#define CONFIG_SYS_I2C_EEPROM_ADDR_LEN 1
#define CONFIG_CMD_EEPROM
#define CONFIG_ZYNQ_GEM_EEPROM_ADDR 0x50
#define CONFIG_ZYNQ_GEM_I2C_MAC_OFFSET 0xFA
#define CONFIG_ZYNQ_BOOT_FREEBSD

/* Define ZYBO PS Clock Frequency to 50MHz */
#define CONFIG_ZYNQ_PS_CLK_FREQ 50000000UL

#include <configs/zynq-common.h>

#endif /* __CONFIG_ZYNQ_ZYB0C_H */
```



include/configs/zynq_zyboc.h

```
CONFIG_ARM=y
CONFIG_ARCH_ZYNQ=y
CONFIG_TARGET_ZYNQ_ZYB0C=y
CONFIG_DEFAULT_DEVICE_TREE="zynq-zyboc"
# CONFIG_SYS_MALLOC_F is not set
CONFIG_SPL=y
CONFIG_FIT=y
CONFIG_FIT_VERBOSE=y
CONFIG_FIT_SIGNATURE=y
# CONFIG_CMD_IMLS is not set
# CONFIG_CMD_FLASH is not set
# CONFIG_CMD_SETEXPR is not set
CONFIG_OF_EMBED=y
```

```
$ nano arch/arm/dts/Makefile
```

```
dtb-$(CONFIG_ARCH_ZYNQ) += zynq-zc702.dtb \
    zynq-zc706.dtb \
    zynq-zed.dtb \
    zynq-zybo.dtb \
    zynq-microzed.dtb \
    zynq-cc108.dtb \
    zynq-afx-nand.dtb \
    zynq-afx-nor.dtb \
    zynq-afx-qspi.dtb \
    zynq-cse-nand.dtb \
    zynq-cse-nor.dtb \
    zynq-cse-qspi.dtb \
    zynq-picoded.dtb \
    zynq-zc770-xm010.dtb \
    zynq-zc770-xm011.dtb \
    zynq-zc770-xm012.dtb \
    - zynq-zc770-xm013.dtb
+ zynq-zc770-xm013.dtb \
+ zynq-zyboc.dtb
```

```
$ nano arch/arm/mach-zynq/Kconfig
```

```
config TARGET_ZYNQ_ZYBO
    bool "Zynq Zybo Board"
    select ZYNQ_CUSTOM_INIT

config TARGET_ZYNQ_AFX
    bool "Zynq AFX Board"
    select ZYNQ_CUSTOM_INIT

config TARGET_ZYNQ_CSE
    bool "Zynq CSE Board"
    select ZYNQ_CUSTOM_INIT

config TARGET_ZYNQ_CC108
    bool "Zynq CC108 Board"
    select ZYNQ_CUSTOM_INIT

+config TARGET_ZYNQ_ZYBOC
+  bool "Zynq ZyboC Board"
+  select ZYNQ_CUSTOM_INIT
.....
config SYS_CONFIG_NAME
    default "zynq_zed" if TARGET_ZYNQ_ZED
    default "zynq_microzed" if TARGET_ZYNQ_MICROZED
    default "zynq_picozed" if TARGET_ZYNQ_PICOZED
    default "zynq_zc70x" if TARGET_ZYNQ_ZC702 || TARGET_ZYNQ_ZC706 \
        || TARGET_ZYNQ_ZC70X
    default "zynq_zc770" if TARGET_ZYNQ_ZC770
    default "zynq_zybo" if TARGET_ZYNQ_ZYBO
    default "zynq_cse" if TARGET_ZYNQ_CSE
    default "zynq_afx" if TARGET_ZYNQ_AFX
    default "zynq_cc108" if TARGET_ZYNQ_CC108
+  default "zynq_zyboC" if TARGET_ZYNQ_ZYBOC

endif
```

```
$ make arch=ARM zynq_zyboc_defconfig  
$ make arch=ARM CROSS_COMPILE=arm-linux-gnueabihf- CFLAGS="-O2  
-mtune=cortex-a9 -mfpu=neon -mfloat-abi=hard" all  
$ cp u-boot ~/work/embedded_linux/vivado/u-boot.elf
```



- **Kernel driver** for platform devices

- = Kernel source code + Device Tree

- Hierarchy of Devices

- Provide register address, irq number, so on as property of device.

- **Platform device**

- Instead of being dynamically detected, must be statically described in either:

- Kernel source code or Device Tree

- The devices on I2C buses or SPI buses, or the devices directly part of the system-on-chip.



Device Tree Compile

```
$ cd ~/work/embedded_linux  
$ cp vivado/embedded_linux/embedded_linux.tree/system.dts .  
$ nano system.dts
```

system.dts

```
&clkc {  
    fclk-enable = <0x0>;  
    ps-clk-frequency = <50000000>;  
};  
  
+&gem0 {  
+    phy-handle = <&phy0>;  
+    ps7_ethernet_0_mdio: mdio {  
+        #address-cells = <0x1>;  
+        #size-cells = <0x0>;  
+        phy0: phy@0 {  
+            compatible = "realtek,RTL8211E";  
+            device_type = "ethernet-phy";  
+            reg = <0>;  
+        };  
+    };  
+};
```

```
+&i2c0 {  
+    eeprom@50 {  
+        /* Microchip 24AA02E48 */  
+        compatible = "microchip,24c02";  
+        reg = <0x50>;  
+        pagesize = <8>;  
+    };  
+};  
+/  
+  usb_phy0: phy0 {  
+      compatible = "ulpi-phy";  
+      #phy-cells = <0>;  
+      reg = <0xe0002000 0x1000>;  
+      view-port = <0x0170>;  
+      drv-vbus;  
+  };  
+};  
+&usb0 {  
+    usb-phy = <&usb_phy0>;  
+};
```

```
$ dtc -O dtb -I dts -i vivado/embedded_linux/embedded_linux.tree/ -o  
devicetree.dtb system.dts
```

```
$ cd ~/work/embedded_linux
$ wget https://github.com/Xilinx/linux-xlnx/archive/xilinx-v2015.4.01.tar.gz -O
dl/linux-xlnx-xilinx-v2015.4.01.tar.gz
$ cd kernel
$ tar xvzf ..../dl/linux-xlnx-xilinx-v2015.4.01.tar.gz
$ cd linux-xlnx-xilinx-v2015.4.01
$ cp -r ./drivers/rtl8192cu drivers/net/wireless/
$ nano drivers/net/wireless/Kconfig
```

drivers/net/wireless/Kconfig

```
source "drivers/net/wireless/mwifiex/Kconfig"
source "drivers/net/wireless/cw1200/Kconfig"
source "drivers/net/wireless/rsi/Kconfig"
+source "drivers/net/wireless/rtl8192cu/Kconfig"

endif # WLAN
```

\$ nano drivers/net/wireless/Makefile

drivers/net/wireless/Makefile

```
obj-$(CONFIG_CW1200)  += cw1200/
obj-$(CONFIG_RSI_91X)  += rsi/
+obj-$(CONFIG_RTL8192CU)  += rtl8192cu/
```

```
$ make ARCH=arm xilinx_zynq_defconfig
$ make ARCH=arm menuconfig
```



Linux Kernel Compile

```
hokim@ubuntu: ~/work/embedded_linux/kernel/linux-xlnx-xilinx-v2015.4.01
.config - Linux/arm 4.0.0 Kernel Configuration
> Networking support > Wireless
    -Wireless-
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 [ ]
--- Wireless
<*> cfg80211 - wireless configuration API
[ ] nl80211 testmode command (NEW)
[ ] enable developer warnings (NEW)
[ ] cfg80211 regulatory debugging (NEW)
[ ] cfg80211 certification onus (NEW)
[*] enable powersave by default (NEW)
[ ] use statically compiled regulatory rules database (NEW)
[ ] cfg80211 wireless extensions compatibility (NEW)
<*> Generic IEEE 802.11 Networking Stack (mac80211)
(+)

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

```
hokim@ubuntu: ~/work/embedded_linux/kernel/linux-xlnx-xilinx-v2015.4.01
.config - Linux/arm 4.0.0 Kernel Configuration
> Device Drivers > Device Tree and Open Firmware support
    Device Tree and Open Firmware 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 [ ]
[ ] Device Tree runtime unit tests
[*] Device Tree overlays

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

Linux Kernel Compile

```
hokim@ubuntu: ~/work/embedded_linux/kernel/linux-xlnx-xilinx-v2015.4.01
.config - Linux/arm 4.0.0 Kernel Configuration
[...] rivers > Network device support > PHY Device support and infrastructure
    PHY Device support and infrastructure
        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 [ ]
        ^(-)
            < > Drivers for the Intel LXT PHYs
            < > Drivers for the Cicada PHYs
            <*> Drivers for the Vitesse PHYs
            < > Drivers for SMSC PHYs
            < > Drivers for Broadcom PHYs
            < > Drivers for Broadcom 7xxx SOCs internal PHYs
            < > Driver for Broadcom BCM8706 and BCM8727 PHYs
            < > Drivers for ICPlus PHYs
            <*> Drivers for Realtek PHYs
            < > Drivers for National Semiconductor PHYs
        +(+)

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

```
hokim@ubuntu: ~/work/embedded_linux/kernel/linux-xlnx-xilinx-v2015.4.01
.config - Linux/arm 4.0.0 Kernel Configuration
> Device Drivers > Network device support > Wireless LAN
    Wireless LAN
        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 [ ]
        ^(-)
            < > Hermes chipset 802.11b support (Orinoco/Prism2/Symbol) (NEW)
            < > Softmac Prism54 support (NEW)
            < > Ralink driver support (NEW) ----
            < > Realtek rtlwifi family of devices -----
            [ ] TI Wireless LAN support ----
            < > ZyDAS ZD1211/ZD1211B USB-wireless support (NEW)
            < > Marvell WiFi-Ex Driver (NEW)
            < > CW1200 WLAN support (NEW)
            < > Redpine Signals Inc 91x WLAN driver support (NEW)
            <*> Realtek 8192C USB WiFi

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

```
hokim@ubuntu: ~/work/embedded_linux/kernel/linux-xlnx-xilinx-v2015.4.01
.config - Linux/arm 4.0.0 Kernel Configuration
> Device Drivers > SPI support
  - SPI 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 [ ]
    ^(-)
      < > NXP SC18IS602/602B/603 I2C to SPI bridge
      < > Analog Devices AD-FMCOMMS1-EBZ SPI-I2C-bridge driver
      <*> Xilinx SPI controller common module
      <*> Xilinx Zynq QSPI controller
      [ ]   Xilinx Zynq QSPI Dual stacked configuration
      < > Xilinx ZynqMP GQSPI controller
      < > DesignWare SPI controller core support
      *** SPI Protocol Masters ***
      <*> User mode SPI device driver support
      < > Infineon TLE62X0 (for power switching)

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

```
hokim@ubuntu: ~/work/embedded_linux/kernel/linux-xlnx-xilinx-v2015.4.01
.config - Linux/arm 4.0.0 Kernel Configuration
> Device Drivers > GPIO Support
  - GPIO 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 [ ]
    ^(-)
      < > GPIO driver for 74xx-ICs with MMIO access
      -* Generic memory-mapped GPIO controller support (MMIO platform
      < > Synopsys DesignWare APB GPIO driver
      < > Emma Mobile GPIO
      [ ] LSI ZEVIO SoC memory mapped GPIOs
      [ ] PrimeCell PL061 GPIO support
      < > SMSC SCH311x SuperI/O GPIO
      < > GPIO based on SYSCON
      <*> Xilinx GPIO support
      <*> Xilinx Zynq GPIO support
    ^(+)

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

```
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- CFLAGS="-O2  
-mtune=cortex-a9 -mfpu=neon -mfloat-abi=hard" -j $(nproc)  
UIMAGE_LOADADDR=0x8000 ulimage  
$ cp arch/arm/boot/ulimage ../../
```



```
$ cd ~/work/embedded_linux  
$ sudo sh ./image.sh
```

image.sh

```
mkdir -p dl  
  
UBUNTU_URL=http://cdimage.ubuntu.com/ubuntu-base/releases/16.04/release  
UBUNTU=ubuntu-base-16.04-core-armhf.tar.gz  
if [ ! -f dl/$UBUNTU ]; then  
    wget $UBUNTU_URL/$UBUNTU -O dl/$UBUNTU  
fi  
  
ARCH=armhf  
SIZE=3500  
mkdir -p img  
IMAGE=img/ubuntu-core_${ARCH}_16.04.img  
dd if=/dev/zero of=$IMAGE bs=1M count=$SIZE  
DEVICE=$(losetup -f)  
losetup $DEVICE $IMAGE  
parted -s $DEVICE mklabel msdos  
parted -s $DEVICE mkpart primary fat16 4MB 128MB  
parted -s $DEVICE mkpart primary ext4 128MB 100%  
BOOT_DEV=/dev/$(lsblk -lno NAME $DEVICE | sed '2!d')  
ROOT_DEV=/dev/$(lsblk -lno NAME $DEVICE | sed '3!d')  
mkfs.vfat -v $BOOT_DEV  
mkfs.ext4 -F -j $ROOT_DEV  
ROOT_DIR=root  
mkdir -p $ROOT_DIR  
mount $ROOT_DEV $ROOT_DIR  
cd $ROOT_DIR  
tar xvf ../dl/$UBUNTU  
rm -fr boot  
cd ..
```

Root File System Build

image.sh

```
cat > $ROOT_DIR/etc/fstab << EOF_CAT
# /etc/fstab: static file system information.
# <file system> <mount point> <type> <options>      <dump> <pass>
/dev/mmcblk0p1 /boot      vfat  errors=remount-ro  0    0
/dev/mmcblk0p2 /       ext4   errors=remount-ro  0    1
EOF_CAT

cp /etc/resolv.conf      $ROOT_DIR/etc/
cp /usr/bin/qemu-arm-static $ROOT_DIR/usr/bin/
chroot $ROOT_DIR << EOF_CHROOT
sed -i 's/^# deb http://ports.ubuntu.com/ubuntu-ports/ xenial universe.*/deb http://ports.ubuntu.com/ubuntu-ports/
xenial universe/' /etc/apt/sources.list
sed -i 's/^# deb http://ports.ubuntu.com/ubuntu-ports/ xenial-updates universe.*/deb http://ports.ubuntu.com/ubuntu-
ports/ xenial-updates universe/' /etc/apt/sources.list
apt-get update
apt-get -y upgrade
DEBIAN_FRONTEND=noninteractive apt-get -y install vim nano sudo openssh-server udev usbutils u-boot-tools device-tree-
compiler kmod net-tools wpa_supplicant parted rfkill lshw wireless-tools gcc g++ cmake git i2c-tools iputils-ping
echo "Asia/Seoul" > /etc/timezone
ln -fs /usr/share/zoneinfo/Asia/Seoul /etc/localtime
locale-gen "en_US.UTF-8"
DEBIAN_FRONTEND=noninteractive dpkg-reconfigure locales
EOF_CHROOT
rm $ROOT_DIR/etc/resolv.conf
rm $ROOT_DIR/usr/bin/qemu-arm-static

mkdir -pv $ROOT_DIR/etc/systemd/system/serial-getty@ttyPS0.service.d
cat > $ROOT_DIR/etc/systemd/system/serial-getty@ttyPS0.service.d/autologin.conf << EOF_CAT
[Service]
ExecStart=
ExecStart=-/sbin/agetty --autologin root -s %I 115200,38400,9600 linux
EOF_CAT
umount -l $ROOT_DIR
rmdir $ROOT_DIR
losetup -d $DEVICE
```

using WinSCP

Ubuntu	Windows
~/work/embedded_linux/ devicetree.dtb, ulmage	C:\Users\hokim\work\embedded_linux
~/work/embedded_linux/vivado/ u-boot.elf	C:\Users\hokim\work\embedded_linux\vivado
~/work/embedded_linux/img/ ubuntu-core_armhf_16.04.img	C:\Users\hokim\work



bootbin.tcl

```
# tclsh bootbin.tcl

set project_name embedded_linux

set fileId [open $project_name/boot.bif "w"]
puts $fileId "img:{\[bootloader\]} $project_name/$project_name.fsbl/executable.elf
$project_name/$project_name.runs/impl_1/system_wrapper.bit u-boot.elf}"
close $fileId

file delete -force boot.bin

exec bootgen -image $project_name/boot.bif -w -o i boot.bin >&@stdout
```

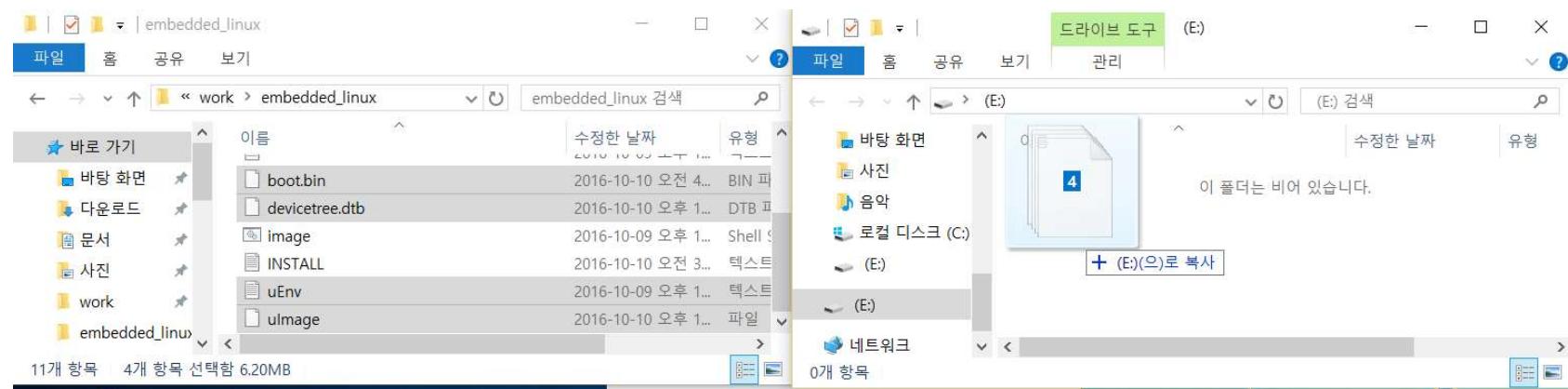
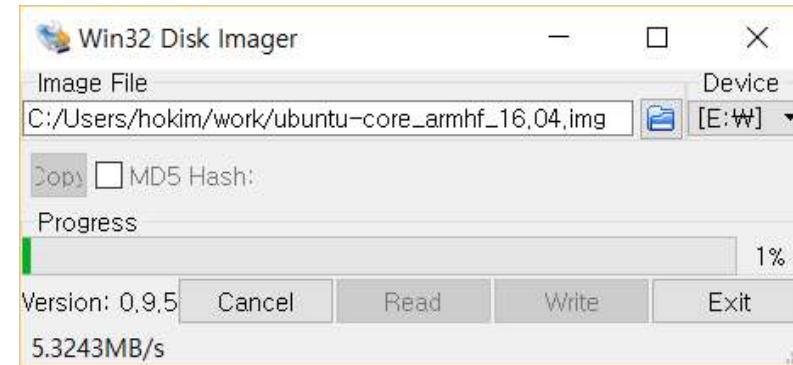
In cmd window for Vivado

```
C:\ cd c:\Users\hokim\work\embedded_linux\vivado
C:\ tclsh bootbin.tcl
C:\ copy boot.bin ..
```

output : boot.bin

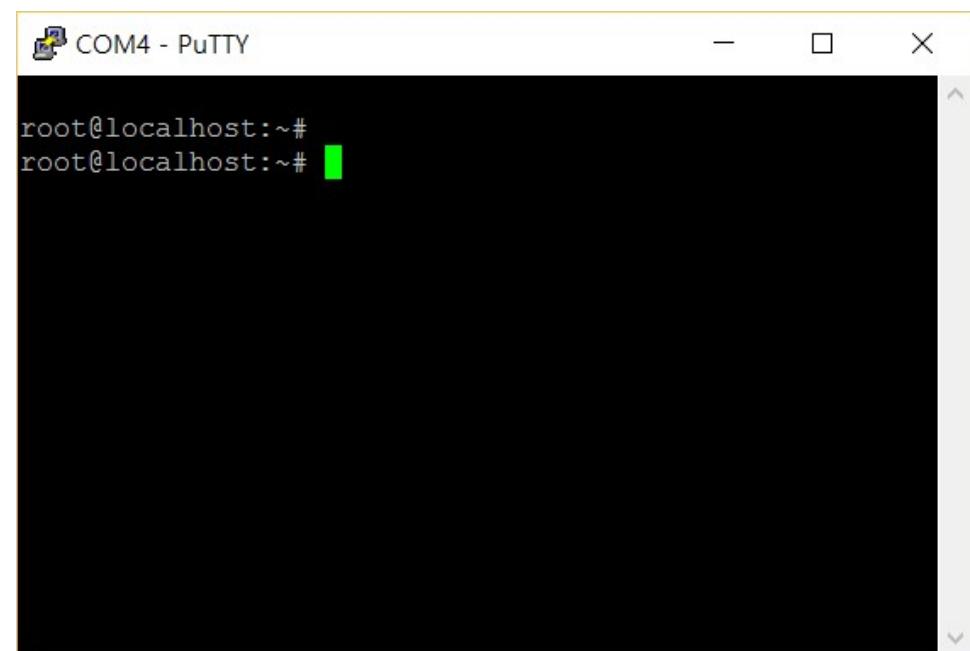
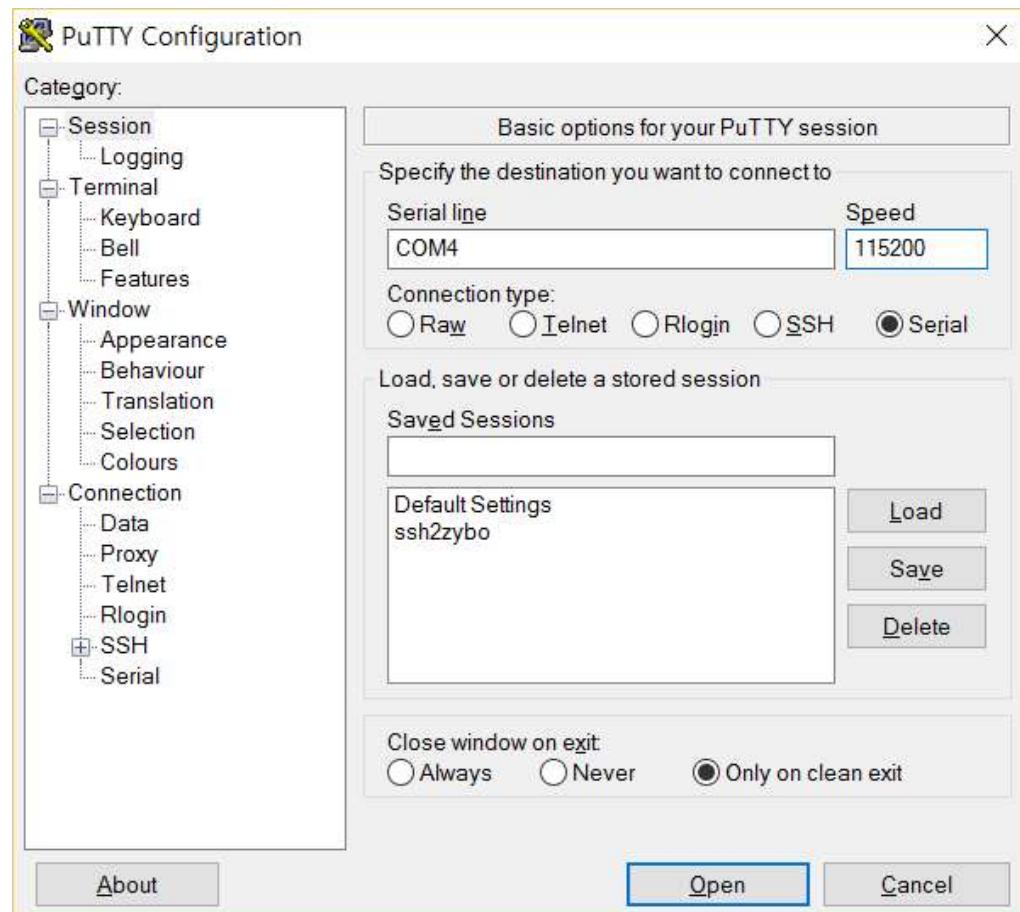


<https://sourceforge.net/projects/win32diskimager/>



Boot zybo with SD card

Log in through UART console using putty



```
# groupadd -g 1000 hokim
# groupadd -g 1001 admin
# useradd -u 1000 -g 1000 -G adm,dialout,cdrom,audio,dip,video,plugdev,admin
-d /home/hokim -m -s /bin/bash hokim
# passwd hokim
# nano /etc/network/interfaces.d/eth0
```

/etc/network/interfaces.d/eth0

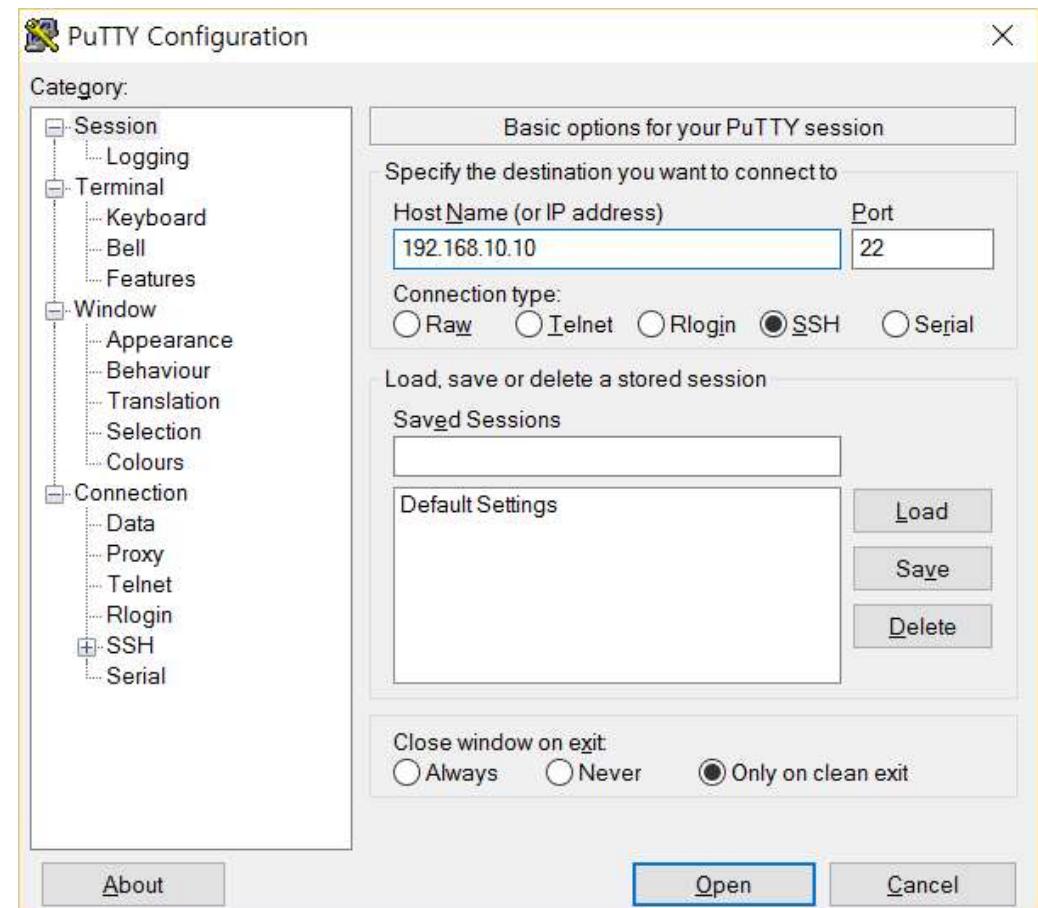
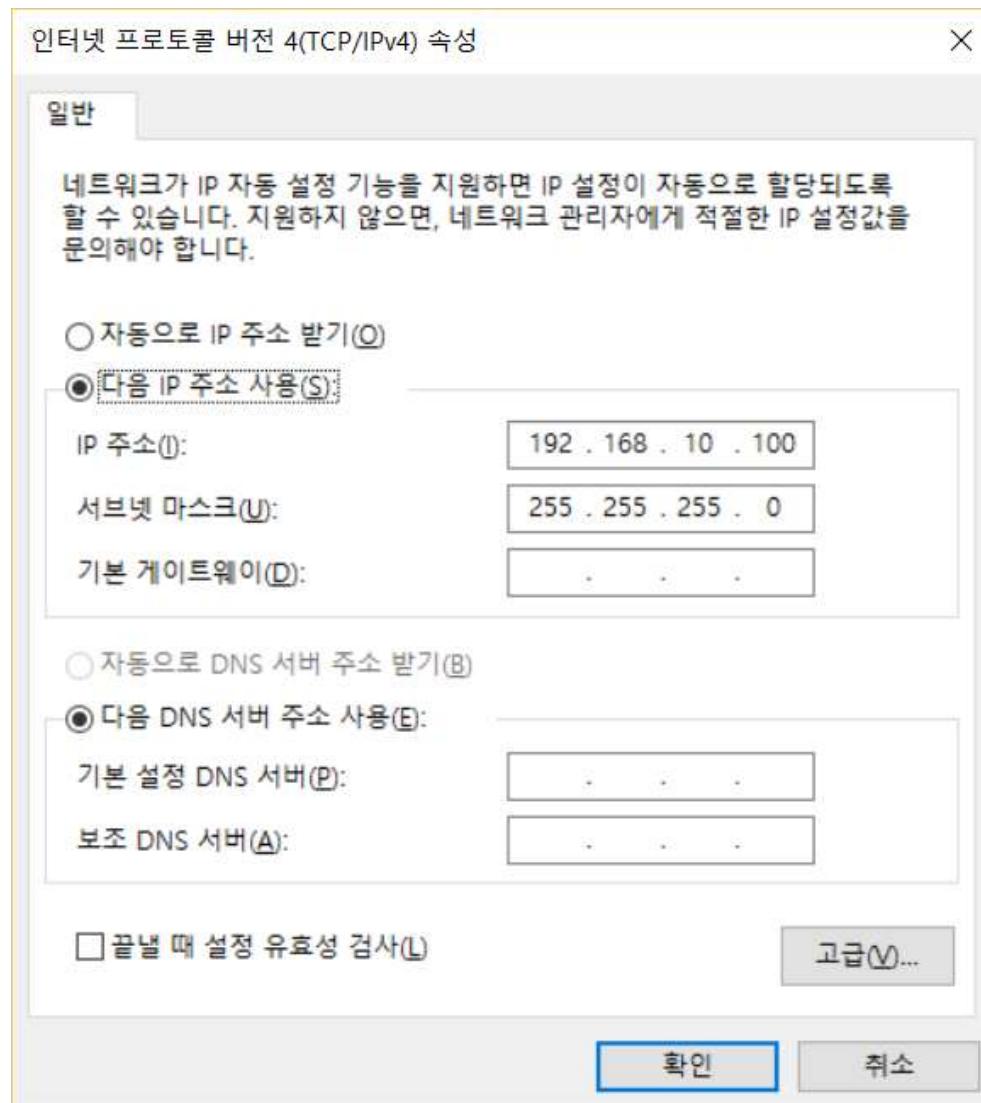
```
allow-hotplug eth0
iface eth0 inet static
address 192.168.10.10
netmask 255.255.255.0
```

```
# halt
```

Turn off/on zybo



Log in through ethernet using putty



```
$ sudo nano /etc/hostname
```

/etc/hostname

```
-localhost.localdomain  
+zybo
```

```
$ sudo nano /etc/hosts
```

/etc/hosts

```
127.0.0.1      localhost  
127.0.1.1      zybo  
  
# The following lines are desirable for IPv6 capable hosts  
::1            ip6-localhost ip6-loopback  
fe00::0        ip6-localnet  
ff00::0        ip6-mcastprefix  
ff02::1        ip6-allnodes  
ff02::2        ip6-allrouters
```

```
$ ls /sys/class/net
```

output:

```
enx74da38422193  eth0  lo
```



```
$ sudo nano /etc/network/interfaces.d/enx74da38422193
```

/etc/network/interfaces.d/enx74da38422193

```
allow-hotplug enx74da38422193
iface enx74da38422193 inet dhcp
    pre-up wpa_supplicant -B -D wext -i enx74da38422193 -c /etc/wpa_supplicant.conf
    post-down killall -q wpa_supplicant
    udhcpc_opts -t7 -T3
```

```
$ sudo nano /etc/wpa_supplicant.conf
```

/etc/wpa_supplicant.conf

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
    ssid="INIPRO"
    key_mgmt=WPA-PSK
    psk="20471047"
}
```

```
$ sudo halt
```

Turn off/on zybo



```
$ sudo nano /etc/network/interfaces.d/enx74da38422193
```

/etc/network/interfaces.d/enx74da38422193

```
allow-hotplug enx74da38422193
iface enx74da38422193 inet dhcp
    pre-up wpa_supplicant -B -D wext -i enx74da38422193 -c /etc/wpa_supplicant.conf
    post-down killall -q wpa_supplicant
    udhcpc_opts -t7 -T3
```

```
$ sudo nano /etc/wpa_supplicant.conf
```

/etc/wpa_supplicant.conf

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
    ssid="INIPRO"
    key_mgmt=WPA-PSK
    psk="20471047"
}
```

```
$ sudo -s
# echo -e "d\n2\nw" | fdisk /dev/mmcblk0
# parted -s /dev/mmcblk0 mkpart primary ext4 128M 100%
# halt
```



Turn off / on zybo
Log in through ethernet using putty

```
$ sudo resize2fs /dev/mmcblk0p2  
$ df -h
```

output:

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/root	7.0G	632M	6.0G	10%	/
devtmpfs	242M	0	242M	0%	/dev
tmpfs	250M	0	250M	0%	/dev/shm
tmpfs	250M	6.5M	244M	3%	/run
tmpfs	5.0M	0	5.0M	0%	/run/lock
tmpfs	250M	0	250M	0%	/sys/fs/cgroup
/dev/mmcblk0p1	118M	6.3M	112M	6%	/boot



```
$ ifconfig
```

Output:

```
enx74da38422193 Link encap:Ethernet HWaddr 74:da:38:42:21:93
inet addr:192.168.0.148 Bcast:192.168.0.255 Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:418 errors:0 dropped:3 overruns:0 frame:0
      TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:77995 (77.9 KB) TX bytes:1772 (1.7 KB)
```

```
eth0    Link encap:Ethernet HWaddr d8:80:39:5c:48:82
inet addr:192.168.10.10 Bcast:192.168.10.255 Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:437 errors:0 dropped:1 overruns:0 frame:0
      TX packets:448 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:33219 (33.2 KB) TX bytes:57605 (57.6 KB)
      Interrupt:145 Base address:0xb000
```

```
lo     Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
      UP LOOPBACK RUNNING MTU:65536 Metric:1
      RX packets:80 errors:0 dropped:0 overruns:0 frame:0
      TX packets:80 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:5920 (5.9 KB) TX bytes:5920 (5.9 KB)
```



- **Virtual memory management is a key aspect of Linux**

The Memory Management Unit(MMU) of the processor translates virtual address to physical addresses

- **Linux divides virtual memory into kernel space and user space**

Kernel space is the memory area for the kernel and device drivers

 kernel space is the top 1 GB of memory, 0xC0000000 to 0xFFFFFFFF

User space is the memory area for user application software

 User space is the bottom 3 GB of memory, 0 to 0xBFFFFFFF

Other kernel/user space memory configurations are configurable in the kernel such as 2 GB kernel and 2GB user space

 Kernel space virtual address 0xC0000000 maps to physical address zero



- **Linux uses the processor modes to create privilege levels**

The kernel executes at a higher privilege level than user space code such that it can access any resources in the system

Applications execute at a lower privilege level such that they must use the kernel to get to the restricted resources in the system

- **Library functions run in user space and provide a more convenient interface for the programmer**

Linux applications require a C library to build which is provided by the tools

The Xilinx Linux GNU tools are based on the GNU C Library(glibc)

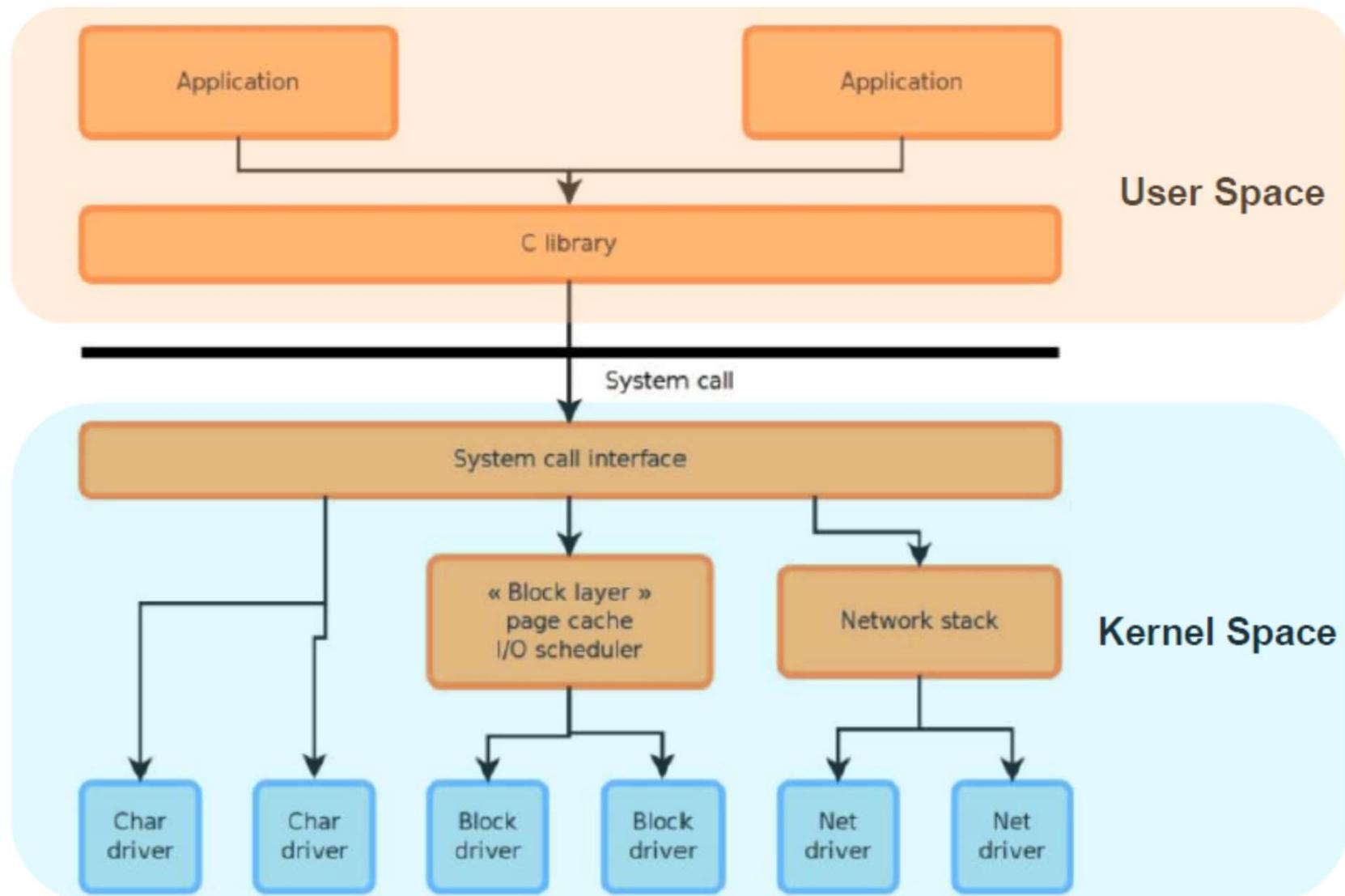
The Xilinx standalone GNU tools are based on newlib library rather than glibc

- **System calls run in kernel mode on the user's behalf and are provided by the kernel itself**



- A library function calls one or more system calls, and these system calls execute in supervisor mode since they are part of kernel itself
- Once the system call complete its task, it returns and execution is transferred back to user mode
- The user space application is typically blocked until the library function and system call return (just like a function call)
- System calls may interact with the kernel property, or with specific drivers and frameworks of the kernel





- You don't have to be a kernel expert, but understanding some terms will help a lot
- The Linux Device model is built around the concept of buses, devices and drivers
- All devices in the system are connected to a bus of some kind
- A bus may be a software abstraction rather than a real bus
- Buses primarily exist to gather similar devices together and coordinate initialization, shutdown and power management
- When a device in the system is found to match a driver, they are bound together. The specifics about how to match devices and drivers are bus-specific



■ Network devices

These are represented as network interfaces, visible in userspace using the ifconfig utility

■ Block devices

These are used to provide userspace applications access to raw storage devices (hard disks, USB keys)

Visible to the applications as device files in /dev

■ Character devices

These are used to provide userspace applications access to all other types of devices (input, sound, graphics, serial, etc.)

They are also visible to the applications as device files in /dev

Many devices are character devices and a lot of user IP could be accessed as a character device



- Many device drivers are not directly implemented as character devices or block devices. They are implemented under a framework, specific to a device type (framebuffer, V4L, serial, etc.)
- The framework factors out the common parts of drivers for the same type of devices to reduce code duplication
- From userspace, many are still seen as normal character devices
- The frameworks provide a coherent userspace interface (ioctl numbering and semantics, etc.) for every type of device, regardless of the driver

The network framework of Linux provides a socket API such that an application can connect to a network using any network driver without knowing the details of the network driver

- sockfd = socket(AF_INET, SOCK_STREAM, 0)

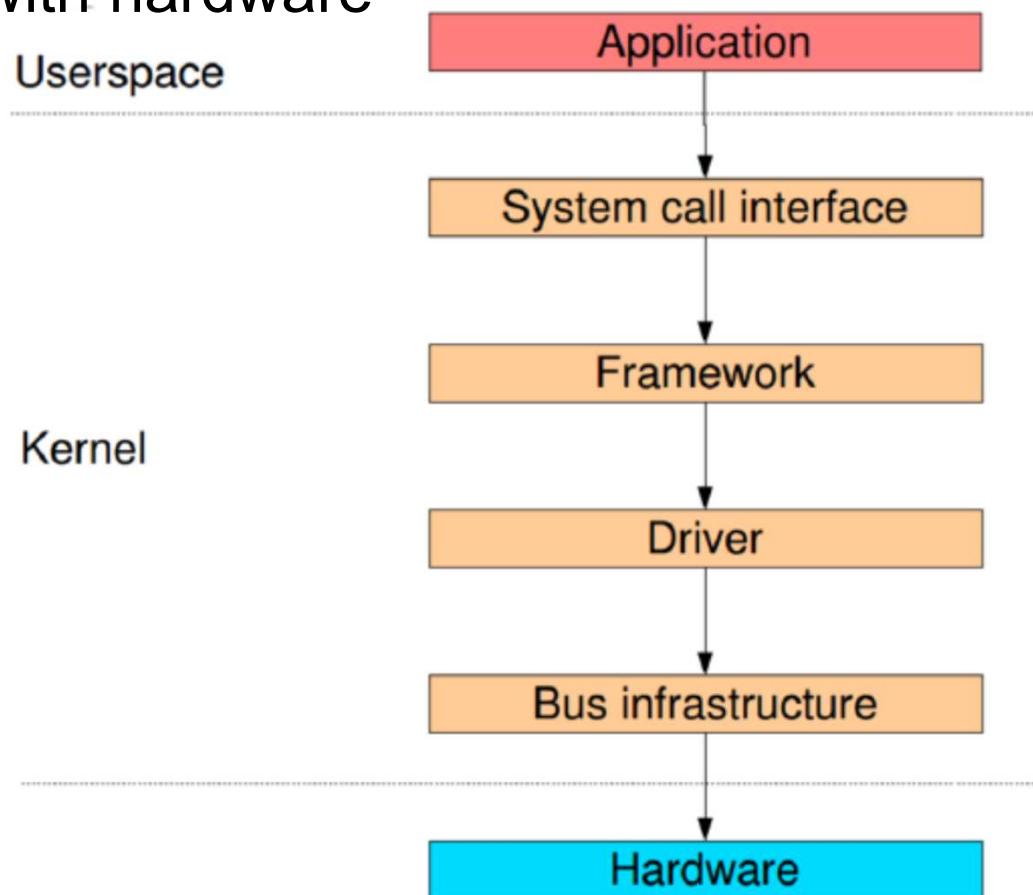


Linux Kernel Layers Focused on Frameworks

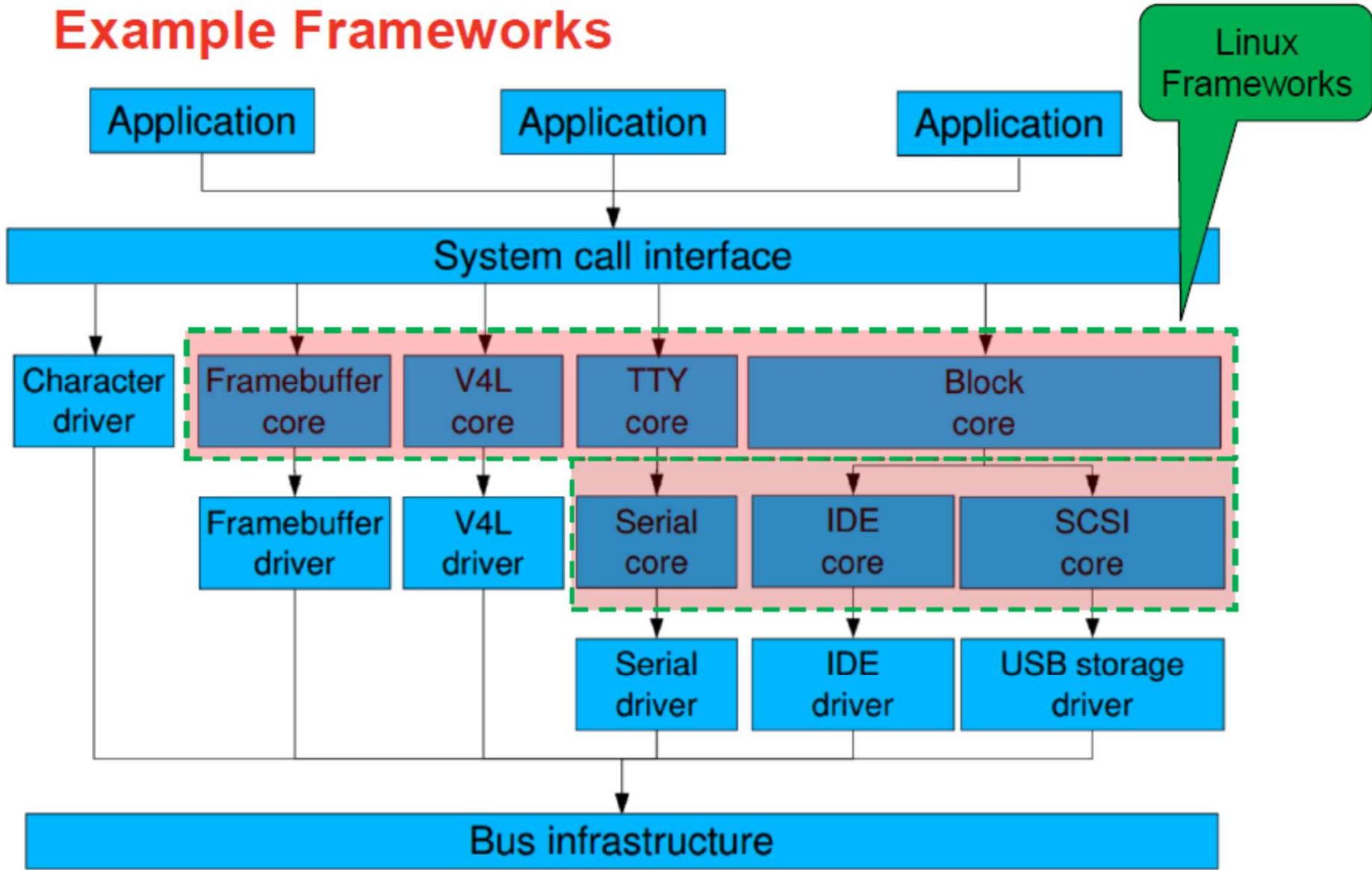
- A driver is always interfacing with:

A framework that allows the driver to expose the hardware features to userspace applications

A bus infrastructure (part of the device model), to detect /communicate with hardware



Example Frameworks



- **System and kernel information**

- Presented to user space application as virtual file systems

- Created dynamically and only exist in memory

- **Two virtual filesystems most known to users**

- proc, mounted on /proc, contains operating system related information (processes, memory management parameters...)

- This is an older mechanism that became somewhat chaotic

- sysfs, mounted on /sys, contains representation of the system as a set of devices and buses together with information about these devices

- This is the newer mechanism and is the preferred place to add system information



- The **sysfs** virtual filesystem is a mechanism for the kernel to export operating details to user space
- The kernel exports the following items to userspace
 - The bus, device, drivers, etc. structures internal to the kernel
 - </sys/bus/> contains the list of buses
 - </sys/devices/> contains the list of devices
 - </sys/class/> enumerates devices by class (net, input, block...), whatever the bus they are connected to
- Used for example by udev to provide automatic module loading, firmware loading, device file creation, etc.



- The principle of the Device Tree is to separate a large part of the hardware description from the kernel sources
- Device Tree allows a single kernel image to run on different boards with the differences being described in the device tree
- This mechanism takes its roots from OpenFirmware (OF) used on PowerPC platforms. This is why the “of” is part of some kernel functions.
- Device Tree is a tree of nodes that models the hierarchy of devices in the system, from the devices inside the processor to the devices on the board
- Each node can have a number of properties describing various properties of the devices: address, interrupts, clocks, etc.
- Written in a specialized language, the Device Tree source code is compiled into a Device Tree Blob by the Device Tree Compiler (DTC)

- The DTC checks the device tree syntax but the semantics of the device tree are checked at runtime by the kernel and drivers
- At boot time, the kernel is given a compiled device tree, referred to as a Device Tree Blob, which is parsed to instantiate all the devices described in the device tree
- Device trees are located in ~~the kernel tree at arch/<arm or microblaze>/boot/dts~~ <https://github.com/Xilinx/device-tree-xlnx>
- The device tree compiler is part of the Linux kernel tree
- Some key properties in a device tree node, referred to as bindings
 - The `compatible` property is used to bind a device with a device driver
 - The `interrupts` property contains the interrupt number used by the device
 - The `reg` property contains the memory range used by the device
- There is limited documentation for the device tree bindings for each device such that driver code inspection may be necessary
 - The docs are in the kernel tree at Documentation/devicetree/bindings



Device Tree Details and A Simple Example

INIPRO

- A simple example below illustrates a node of device tree

An AMBA bus with GPIO that has registers mapped to 0x4120000 and is using interrupt 91

$91 - 32 = 59$, where 32 is the first Shared Peripheral Interrupt

The device is compatible with a driver containing a matching compatible string of “[xlnx,simple](#)”

The device driver source code may be the only way to really understand what properties it is expecting from the device tree

```
ps7_axi_interconnect_0: amba@0 {
    #address-cells = <1>;
    #size-cells = <1>;
    compatible = "xlnx,ps7-axi-interconnect-1.00.a", "simple-bus";
    ranges ;
    axi_gpio_0: gpio@41200000 {
        #gpio-cells = <2>;
        compatible = "xlnx,simple";
        gpio-controller ;
        interrupt-parent = <&ps7_scugic_0>;
        interrupts = <0 59 4>;
        reg = <0x41200000 0x10000>;
        xlnx,is-dual = <0x1>;
    };
};
```



- The dtsi files are included files while the dts file is the final device tree
- A dts file includes dtsi files and the inclusion process works by overlaying the tree of the including file over the tree of the included file
- When properties are repeated in dtsi files the last one is the final
- The PL and PS are separate DTSI files while there is top level dts file that includes them
- The device tree compiler can be used to create the final device tree which is handy for debug (by specifying DTS input and output)



Device Tree – Inclusion Example

ps.dtsi (included file)

```
ps7_ttc_1: ps7-ttc@0xf8002000 {  
    clocks = <&clkc 6>;  
    compatible = "xlnx,ps7-ttc-1.00.a";  
    interrupt-parent = <&ps7_scugic_0>;  
    interrupts = <0 37 4>, <0 38 4>, <0 39 4>;  
    reg = <0xF8002000 0x1000>;  
    status = "disabled";  
};
```

system-top.dts (including file)

```
/include/ "ps.dtsi"  
  
&ps7_ttc_1 {  
    compatible = "xlnx,psttc", "generic-uio";  
    status = "okay";  
};
```



```
ps7_ttc_1: ps7-ttc@0xf8002000 {  
    clocks = <&clkc 6>;  
    compatible = "xlnx,psttc", "generic-uio";  
    interrupt-parent = <&ps7_scugic_0>;  
    interrupts = <0 37 4>, <0 38 4>, <0 39 4>;  
    reg = <0xF8002000 0x1000>;  
    status = "okay";  
};
```

Note the “&” used to reference an existing node (rather than creating a new node)

The result for the duplicated (red) properties is the same as the including file.

- General Purpose Input/Output
- Pin connection two electronic components (chips)
- Voltage is held at one of two level to indicate 1 or 0 logic
- Controlled by one chip, sensed by the other
- Usually grouped in banks
- Per GPIO pin configuration
- Configure pin direction mode to input or output
- Input mode

Sense the logic level

Interrupt source (asynchronous notification)

- Output mode

Set voltage logic level to 0 or 1



- Documented in Documentation/gpio.txt
- gpiolib framework
- Gpio drivers : drivers/gpio/(gpio-zynq.c, gpio-xilinx.c)
- GPIO control interface is via sysfs under /sys/class/gpio, and includes following control files:

export Make a specific GPIO pin available for userspace control. Write the pin number N (e.g. “55”, ASCII); the gpioN directory should appear.

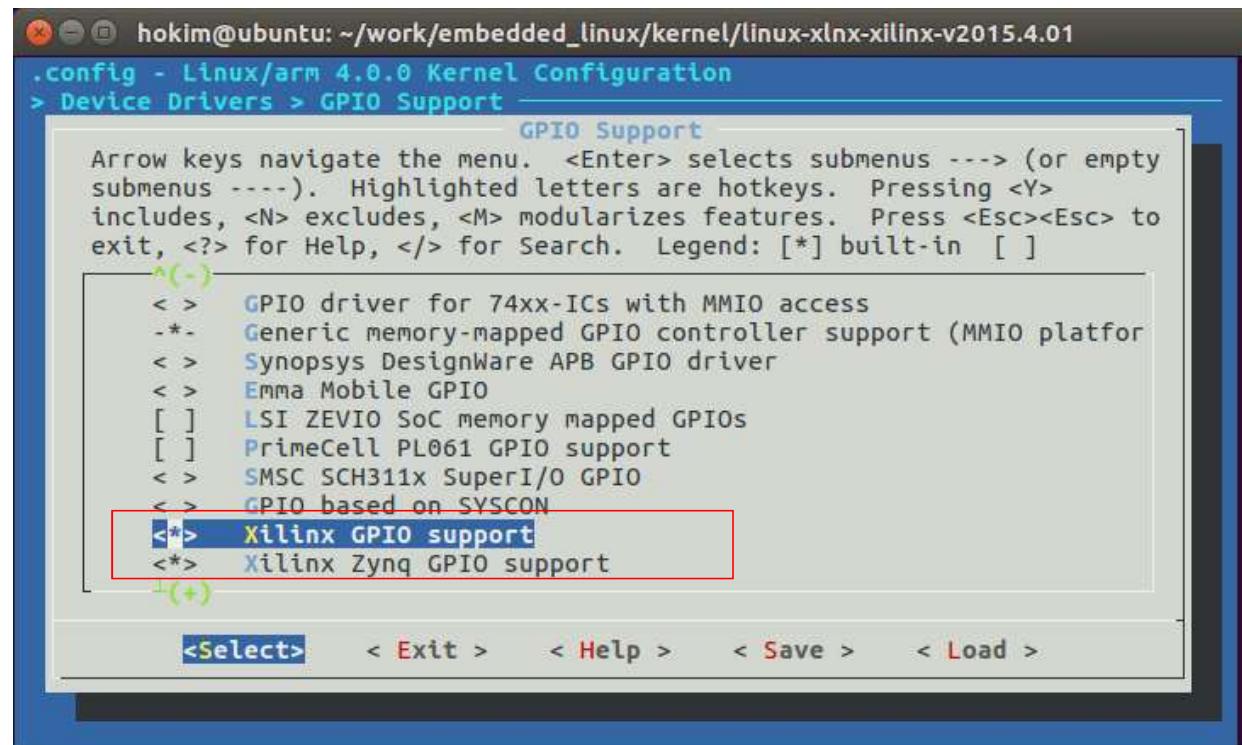
unexport Make a specific GPIO pin unavailable. Write the pin number; the gpioN directory should disappear

gpioN/direction Write “in” or “out” to set pin direction

gpioN/value Read the current pin status in input. For output, write “0” or “1” to set the pin status.



Kernel Configuration



gpio_sw/driver/gpio.c

```
#include <fcntl.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>

#include "gpio.h"

int setGpio(unsigned int index, const char *direction){
    int fd;
    char buf[50];
    sprintf(buf, "/sys/class/gpio/gpio%d/direction", index);
    if(access(buf, F_OK) != -1)
        unsetGpio(index);
    if((fd = open("/sys/class/gpio/export", O_WRONLY)) == -1)
        return -1;
    sprintf(buf, "%d", index);
    if(write(fd, buf, strlen(buf)) == -1)
        return -1;
    close(fd);

    sprintf(buf, "/sys/class/gpio/gpio%d/direction", index);
    if((fd = open(buf, O_WRONLY)) == -1)
        return -1;
    if(write(fd, direction, strlen(direction)) == -1)
        return -1;
    close(fd);
    return 0;
}
```

```
int unsetGpio(unsigned int index){
    int fd;
    char buf[50];
    if((fd = open("/sys/class/gpio/unexport", O_WRONLY)) == -1)
        return -1;
    sprintf(buf, "%d", index);
    if(write(fd, buf, strlen(buf)) == -1)
        return -1;
    close(fd);
    return 0;
}

int writeGpio(unsigned int index, int value){
    int fd;
    char buf[50];
    sprintf(buf, "/sys/class/gpio/gpio%d/value", index);
    if((fd = open(buf, O_WRONLY)) == -1)
        return -1;
    sprintf(buf, "%d", value);
    if(write(fd, buf, strlen(buf)) == -1)
        return -1;
    close(fd);
    return 0;
}

int readGpio(unsigned int index){
    int fd;
    char buf[50];
    sprintf(buf, "/sys/class/gpio/gpio%d/value", index);
    if((fd = open(buf, O_RDONLY)) == -1)
        return -1;
    if(read(fd, buf, 10) == -1)
        return -1;
    close(fd);
    return atoi(buf);
}
```

(/dev/mem)

- A character driver referred to as /dev/mem exists in the kernel that will map device memory into userspace
- With this driver userspace applications can access device memory
- Must be root user
- A great tool for prototyping or maybe testing new hardware, but is not considered to be an acceptable production solution for a userspace device driver
- Since it can map any address into userspace, a buggy userspace driver could crash the kernel



Linux GPIO Userspace Interface



gpio_sw/mmap/main.c

```
#include <stdio.h>
#include <stdint.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <signal.h>
#include "xgpio.h"

XGpio_Config XGpio_ConfigTable[3] = {
    { 0, 0, 1, 0 },
    { 1, 0, 1, 0 },
    { 2, 0, 1, 0 }
};

static int loop_exit;

void sig_handler(int signo)
{
    if (signo == SIGINT)
        loop_exit = 1;
}

int main()
{
    int fd;
    uint32_t btns, leds, sws;

    if ((fd = open("/dev/mem", O_RDWR)) < 0)
        perror("open");
    return 1;
}
```

```
btns = (uint32_t)mmap(NULL, sysconf(_SC_PAGESIZE),
PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0x40001000);
leds = (uint32_t)mmap(NULL, sysconf(_SC_PAGESIZE),
PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0x40000000);
sws = (uint32_t)mmap(NULL, sysconf(_SC_PAGESIZE),
PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0x40002000);

XGpio_ConfigTable[0].BaseAddress = btns;
XGpio_ConfigTable[1].BaseAddress = leds;
XGpio_ConfigTable[2].BaseAddress = sws;

if (signal(SIGINT, sig_handler) == SIG_ERR) {
    fprintf(stderr, "can't catch SIGINT\n");
    return 1;
}

XGpio Gpio;
int Status;
Status = XGpio_Initialize(&Gpio, 1);
if (Status != XST_SUCCESS) {
    return XST_FAILURE;
}

loop_exit = 0;
while (1) {
    XGpio_DiscreteWrite(&Gpio, 1, 0xF);
    sleep(1);
    XGpio_DiscreteWrite(&Gpio, 1, 0x0);
    sleep(1);
    if (loop_exit == 1) break;
}
munmap((void*)btns, sysconf(_SC_PAGESIZE));
munmap((void*)leds, sysconf(_SC_PAGESIZE));
munmap((void*)sws, sysconf(_SC_PAGESIZE));
close(fd);
return 0;
```

Zynq7 PS Re-customize IP for GPIO



ZYNQ7 Processing System (5.5) Re-customize IP

Documentation Presets IP Location Import XPS Settings

Page Navigator <> Peripheral I/O Pins Summary Report

Zynq Block Design

PS-PL Configuration

Peripheral I/O Pins

MIO Configuration

Clock Configuration

DDR Configuration

SMC Timing Calculation

Interrupts

Peripherals

- Quad SPI Flash
- SRAM/NOR Flash
- NAND Flash
- Ethernet 0
- Ethernet 1
- USB 0
- USB 1
- SD 0
- SD 1
- SPI 0
- SPI 1
- UART 0
- UART 1
- I2C 0
- I2C 1
- CAN 0
- CAN 1
- TTC0
- TTC1
- SWDT
- PJTAG
- TPIU
- GPIO MIO
- GPIO EMIO

Bank 0 LVCMOS 3.3V Bank 1 LVCMOS 1.8V

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 EMIO

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 EMIO

Quad SPI Flash

SRAM/NOR Flash

NAND Flash

Enet0

Enet1

USB0

USB1

SD0

SD1

SPI0

mos

SPI1

UART0

UART1

I2C0

I2C1

CAN0

CAN1

TTC0

TTC1

SWDT

PJTAG

Trace

OK Cancel

Zynq7 PS Re-customize IP for GPIO

Re-customize IP

ZYNQ7 Processing System (5.5)

Documentation Presets IP Location Import XPS Settings

Page Navigator << MIO Configuration Summary Report

Bank 0 I/O Voltage LVCMS 3,3V Bank 1 I/O Voltage LVCMS 1,8V

Search:

MIO Configuration

GPIO MIO MIO

GPIO	MIO	Signal	I/O Type	Speed	Pullup	Direction
GPIO 0	MIO 0	gpio[0]	LVCMS 3,3V	slow	disabled	inout
GPIO 1	MIO 1	gpio[1]	LVCMS 3,3V	slow	disabled	inout
GPIO 2	MIO 2	gpio[2]	LVCMS 3,3V	slow	disabled	inout
GPIO 3	MIO 3	gpio[3]	LVCMS 3,3V	slow	disabled	inout
GPIO 4	MIO 4	gpio[4]	LVCMS 3,3V	slow	disabled	inout
GPIO 5	MIO 5	gpio[5]	LVCMS 3,3V	slow	disabled	inout
GPIO 6	MIO 6	gpio[6]	LVCMS 3,3V	slow	disabled	inout
GPIO 7	MIO 7	gpio[7]	LVCMS 3,3V	slow	disabled	out
GPIO 8	MIO 8	gpio[8]	LVCMS 3,3V	slow	disabled	out
GPIO 9	MIO 9	gpio[9]	LVCMS 3,3V	slow	disabled	inout
GPIO 10	MIO 10	gpio[10]	LVCMS 3,3V	slow	disabled	inout
GPIO 11	MIO 11	gpio[11]	LVCMS 3,3V	slow	disabled	inout
GPIO 12	MIO 12	gpio[12]	LVCMS 3,3V	slow	disabled	inout
GPIO 13	MIO 13	gpio[13]	LVCMS 3,3V	slow	disabled	inout
GPIO 14	MIO 14	gpio[14]	LVCMS 3,3V	slow	disabled	inout
GPIO 15	MIO 15	gpio[15]	LVCMS 3,3V	slow	disabled	inout
GPIO 46	MIO 46	gpio[46]	LVCMS 1,8V	slow	disabled	inout
GPIO 47	MIO 47	gpio[47]	LVCMS 1,8V	slow	disabled	inout
GPIO 50	MIO 50	gpio[50]	LVCMS 1,8V	slow	disabled	inout
GPIO 51	MIO 51	gpio[51]	LVCMS 1,8V	slow	disabled	inout

OK Cancel

Zynq7 PS Re-customize IP for GPIO

Re-customize IP

ZYNQ7 Processing System (5.5)

Documentation Presets IP Location Import XPS Settings

Page Navigator <> Zynq Block Design

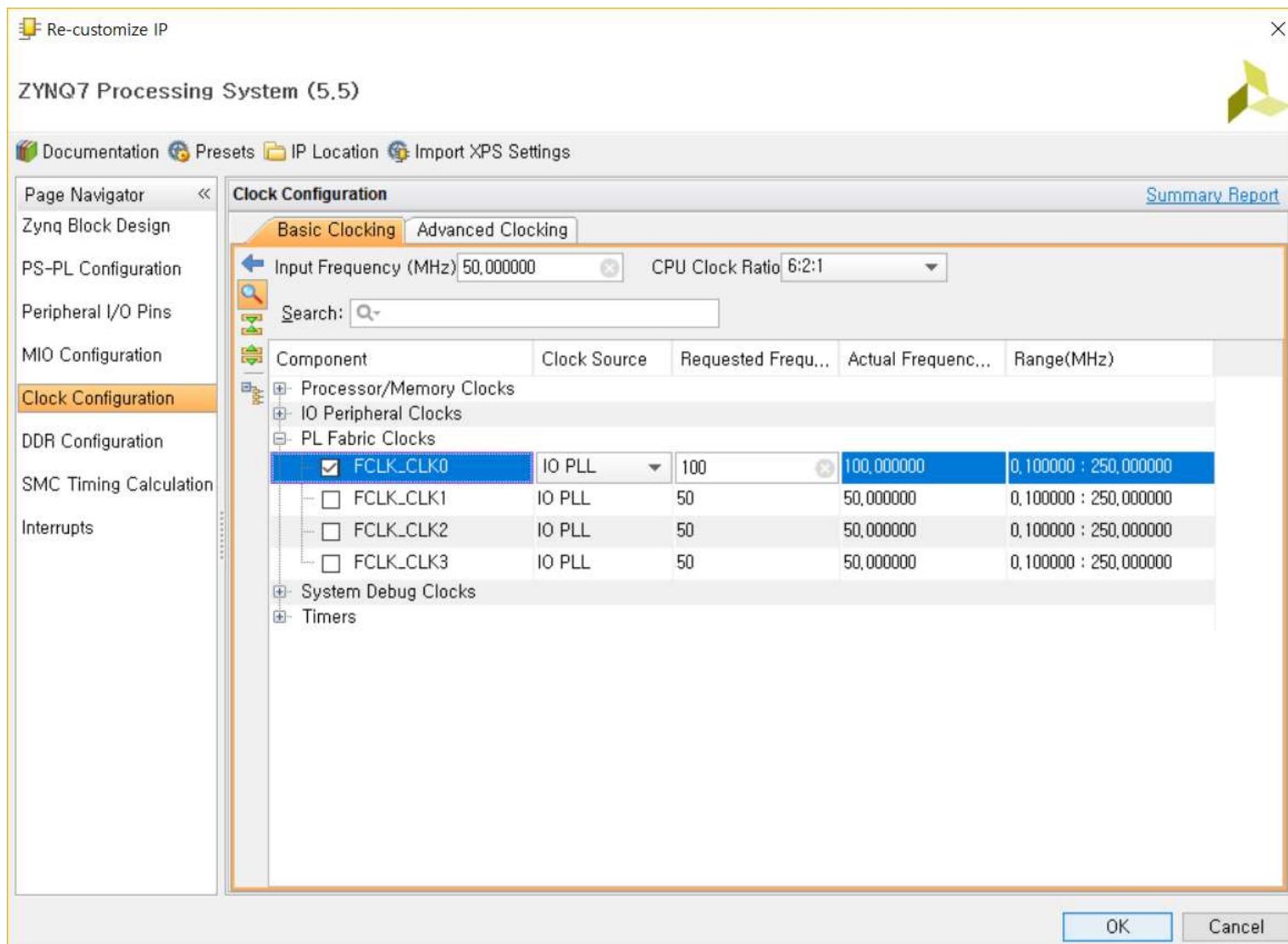
PS-PL Configuration [Summary Report](#)

Search:

Name	Select	Description
FTM Trace buffer FIFO size	128	FTM Trace buffer FIFO size
FTM Trace buffer clock delay	12	Number of clock cycles interval for a trace data output from FIFO.
Include ACP transaction checker	<input type="checkbox"/>	Enables ACP transaction checker.
Trace data/control signal pipeline width	8	Enables configurable number of pipeline stages on the TRACE
Power-on reset(POR) 4k timer	<input type="checkbox"/>	Enables power-on reset(POR) 4k timer. By default, 64k timer is
Processor event interface	<input type="checkbox"/>	Enables event bus which provides a low-latency and direct me
Address Editor		
Enable Clock Triggers		
Enable Clock Resets		
FCLK_RESET0_N	<input checked="" type="checkbox"/>	Enables general purpose reset signal 0 for PL logic
FCLK_RESET1_N	<input type="checkbox"/>	Enables general purpose reset signal 1 for PL logic
FCLK_RESET2_N	<input type="checkbox"/>	Enables general purpose reset signal 2 for PL logic
FCLK_RESET3_N	<input type="checkbox"/>	Enables general purpose reset signal 3 for PL logic
AXI Non Secure Enable	0	Enable AXI Non Secure Transaction
GP Master AXI Interface		
M AXI GP0 Interface	<input checked="" type="checkbox"/>	Enables General purpose AXI master interface 0
M AXI GP1 Interface	<input type="checkbox"/>	Enables General purpose AXI master interface 1
GP Slave AXI Interface		
HP Slave AXI Interface		
ACP Slave AXI Interface		
DMA Controller		

OK Cancel

Zynq7 PS Re-customize IP for GPIO



Zynq7 PS Re-customize IP for GPIO

Re-customize IP

ZYNQ7 Processing System (5.5)

Documentation Presets IP Location Import XPS Settings

Page Navigator << Summary Report

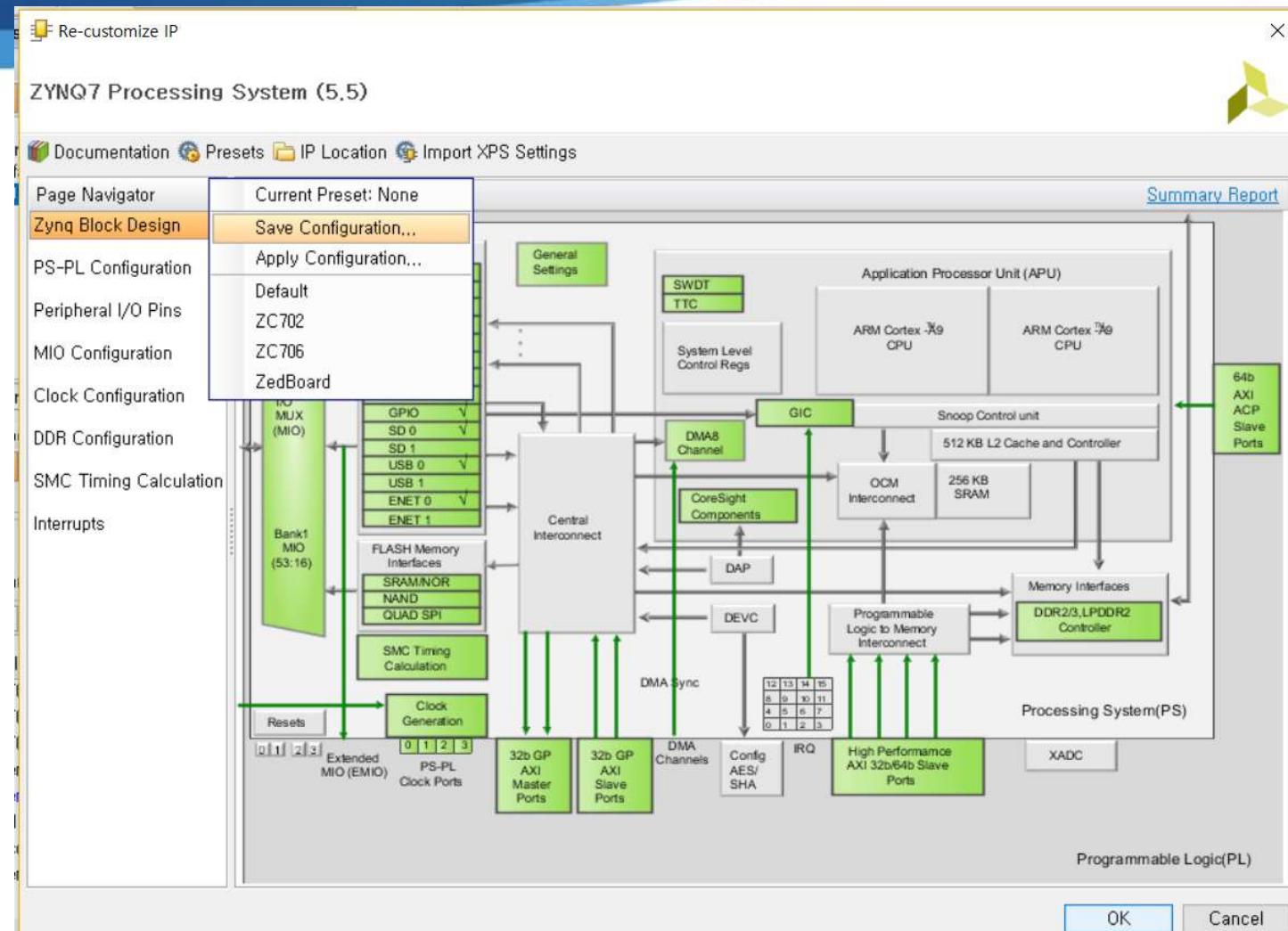
Interrupts

Search:

Interrupt Port	ID	Description
<input checked="" type="checkbox"/> Fabric Interrupts		Enable PL Interrupts to PS and vice versa
<input checked="" type="checkbox"/> PL-PS Interrupt Ports		
<input checked="" type="checkbox"/> IRQ_F2P[15:0]	[91:84], [6..	Enables 16-bit shared interrupt port from the PL, MSB is assigned to...
<input type="checkbox"/> Core0_nFIQ	28	Enables fast private interrupt signal for CPU0 from the PL
<input type="checkbox"/> Core0_nIRQ	31	Enables private interrupt signal for CPU0 from the PL
<input type="checkbox"/> Core1_nFIQ	28	Enables fast private interrupt signal for CPU1 from the PL
<input type="checkbox"/> Core1_nIRQ	31	Enables private interrupt signal for CPU1 from the PL
<input checked="" type="checkbox"/> PS-PL Interrupt Ports		

OK Cancel

Zynq7 PS Re-customize IP for GPIO



Save Current Configuration...

Save the current configuration to a file on disk.

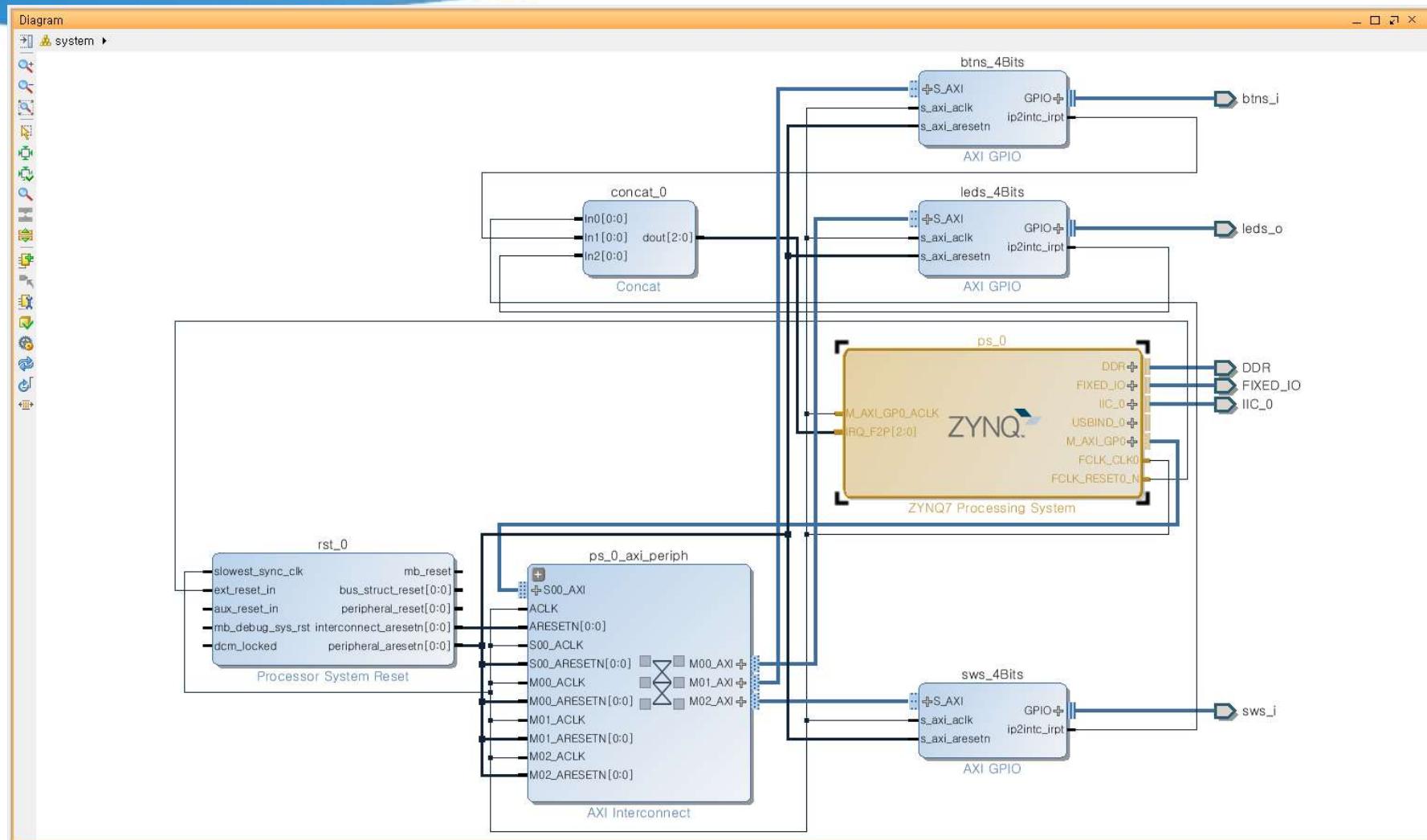
Preset Name: gpio

File name: /dded_linux/projects/gpio/gpio_preset.tcl

OK Cancel

This dialog box is used to save the current configuration settings. The 'Preset Name' field is set to 'gpio', and the 'File name' field shows the path '/dded_linux/projects/gpio/gpio_preset.tcl'. The 'OK' and 'Cancel' buttons are at the bottom.

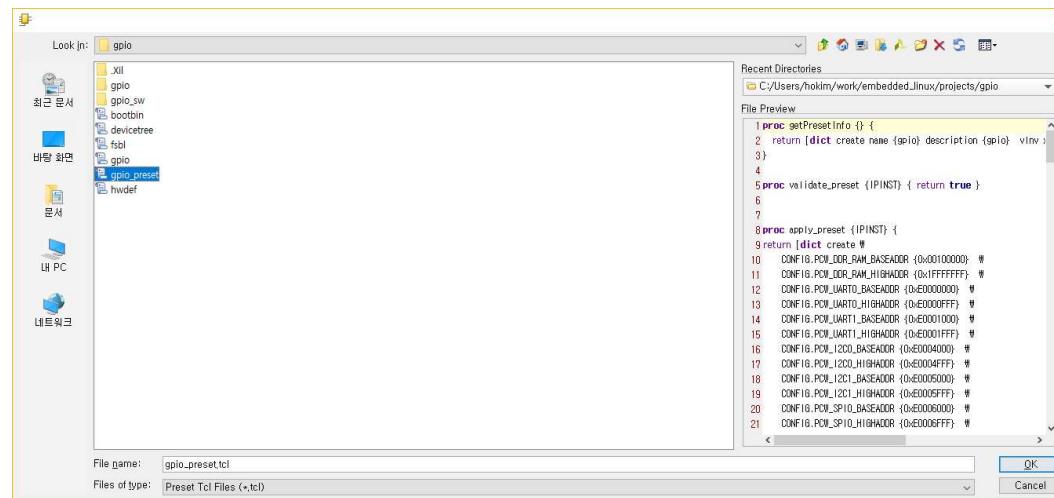
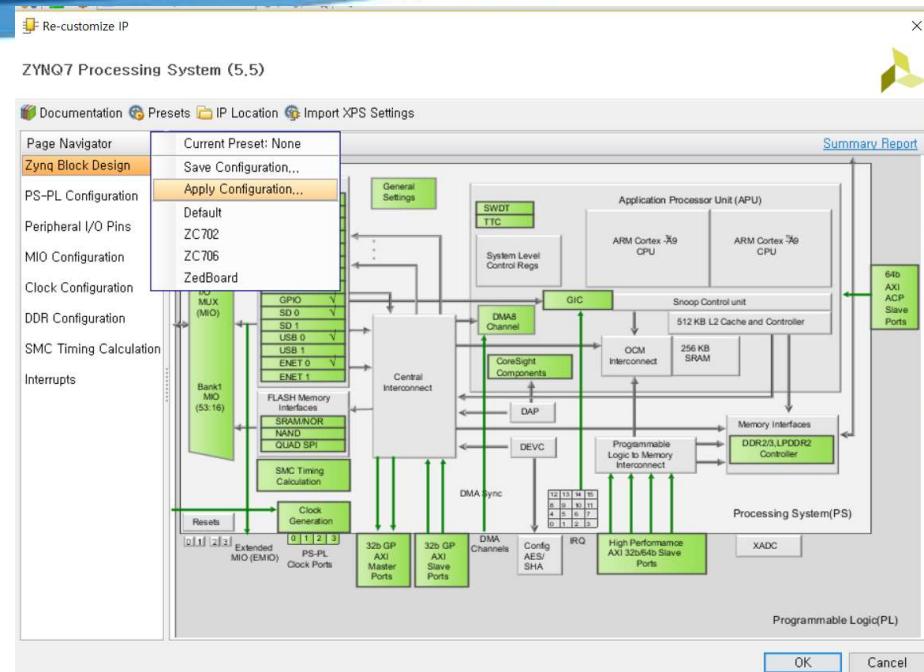
IP Integrator Diagram for GPIO



Address Editor

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
ps_0	S_AXI				
Data (32 address bits : 0x40000000 [1G])					
leds_4Bits	S_AXI	Reg	0x4000_0000	4K	0x4000_0FFF
bt2s_4Bits	S_AXI	Reg	0x4000_1000	4K	0x4000_1FFF
sws_4Bits	S_AXI	Reg	0x4000_2000	4K	0x4000_2FFF

IP Integrator Diagram for GPIO



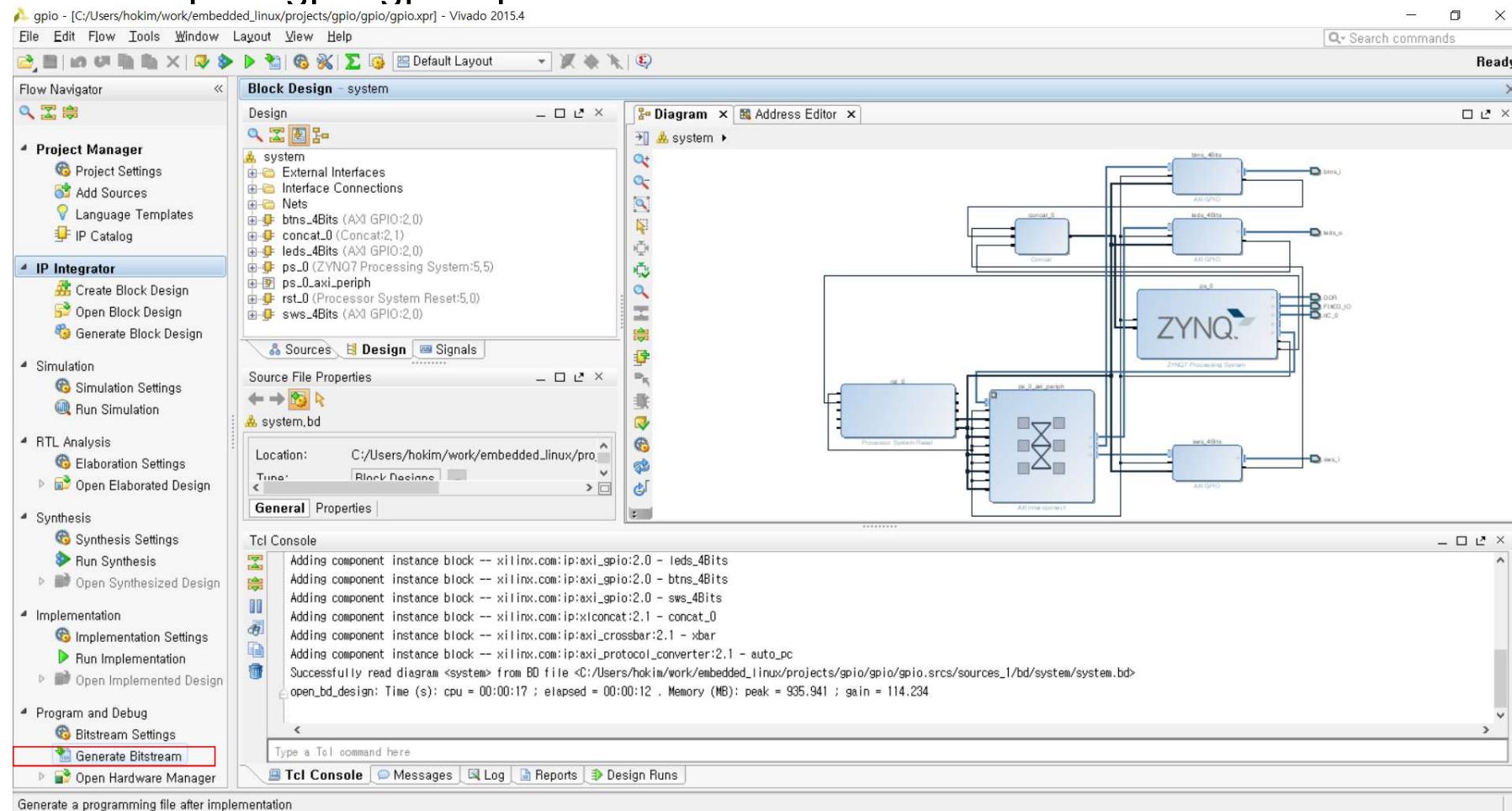
Bit Generation for GPIO

https://github.com/inipro/embedded_linux/projects/gpio

In cmd window for Vivado

```
C:\ cd C:\Users\hokim\work\embedded_linux\projects\gpio  
C:\ vivado -nolog -nojournal -mode batch -source gpio.tcl
```

output : gpio\gpio.xpr...



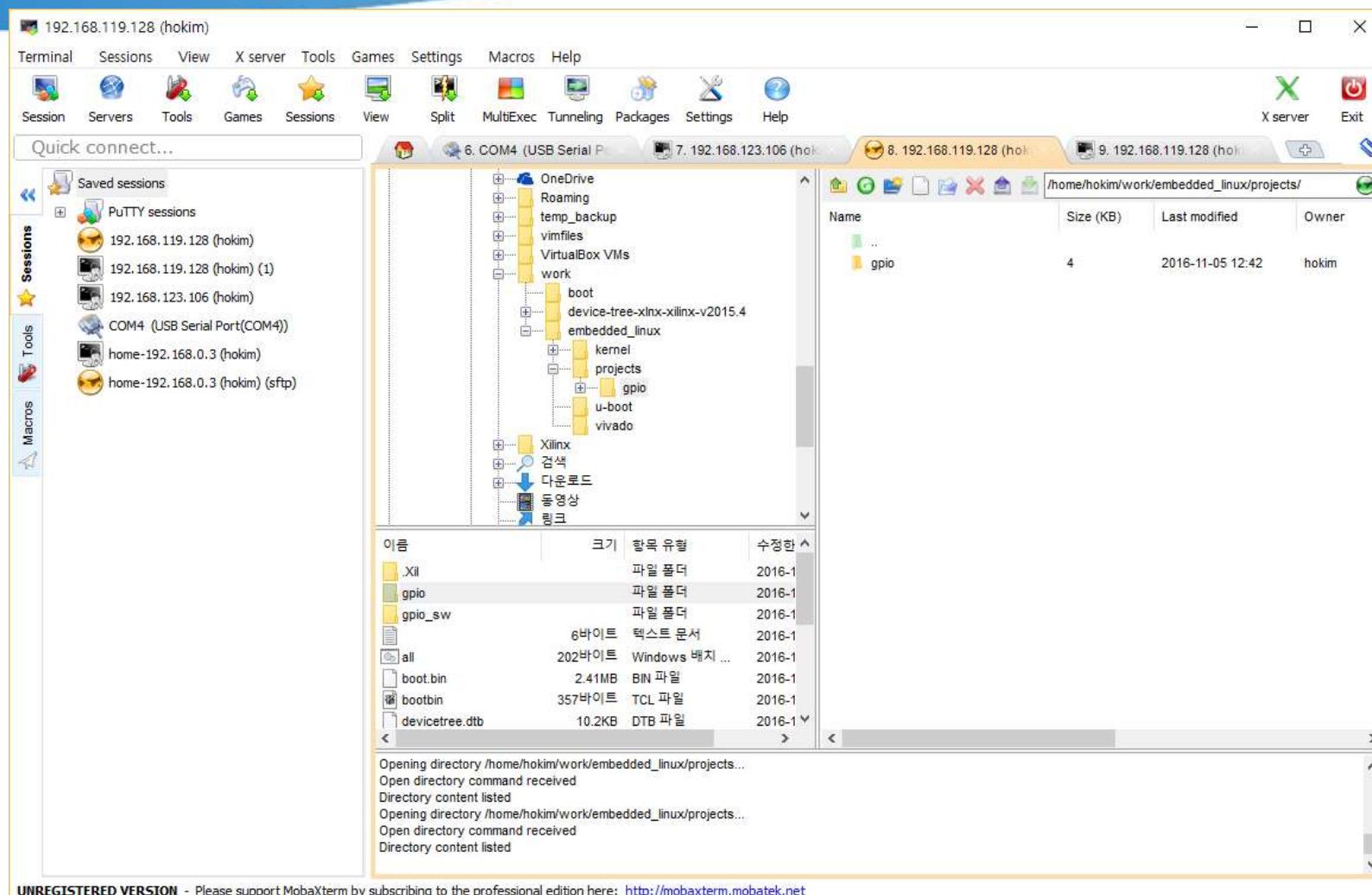
C:\Users\hokim\work\embedded_linux\projects\gpio\all.bat

```
call vivado -nolog -nojournal -mode batch -source hwdef.tcl  
call hsi -nolog -nojournal -mode batch -source fsbl.tcl  
call tclsh bootbin.tcl  
  
call hsi -nolog -nojournal -mode batch -source devicetree.tcl
```

```
C:\ cd C:\Users\hokim\work\embedded_linux\projects\gpio  
C:\ all
```



Device Tree Compile for GPIO



```
$ cd ~/work/embedded_linux/projects
$ cp gpio/gpio.tree/system.dts  system_gpio.dts
$ nano system_gpio.dts
```

Device Tree Compile for GPIO

system_gpio.dts

```
/dts-v1/;  
/include/ "zynq-7000.dtsci"  
/include/ "pl.dtsci"  
.....  
.....  
&clkc {  
    fclk-enable = <0x1>;  
    ps-clk-frequency = <50000000>;  
};  
  
+&gem0 {  
    phy-handle = <&phy0>;  
    ps7_ethernet_0_mdio: mdio {  
        #address-cells = <0x1>;  
        #size-cells = <0x0>;  
        phy0: phy@0 {  
            compatible = "realtek,RTL8211E";  
            device_type = "ethernet-phy";  
            reg = <0>;  
        };  
    };  
};  
};
```

```
+&i2c0 {  
+    eeprom@50 {  
+        /* Microchip 24AA02E48 */  
+        compatible = "microchip,24c02";  
+        reg = <0x50>;  
+        pagesize = <8>;  
+    };  
+};  
+/  
+ {  
+    usb_phy0: phy0 {  
+        compatible = "ulpi-phy";  
+        #phy-cells = <0>;  
+        reg = <0xe0002000 0x1000>;  
+        view-port = <0x0170>;  
+        drv-vbus;  
+    };  
+};  
+&usb0 {  
+    usb-phy = <&usb_phy0>;  
+};
```



Device Tree Compile for GPIO

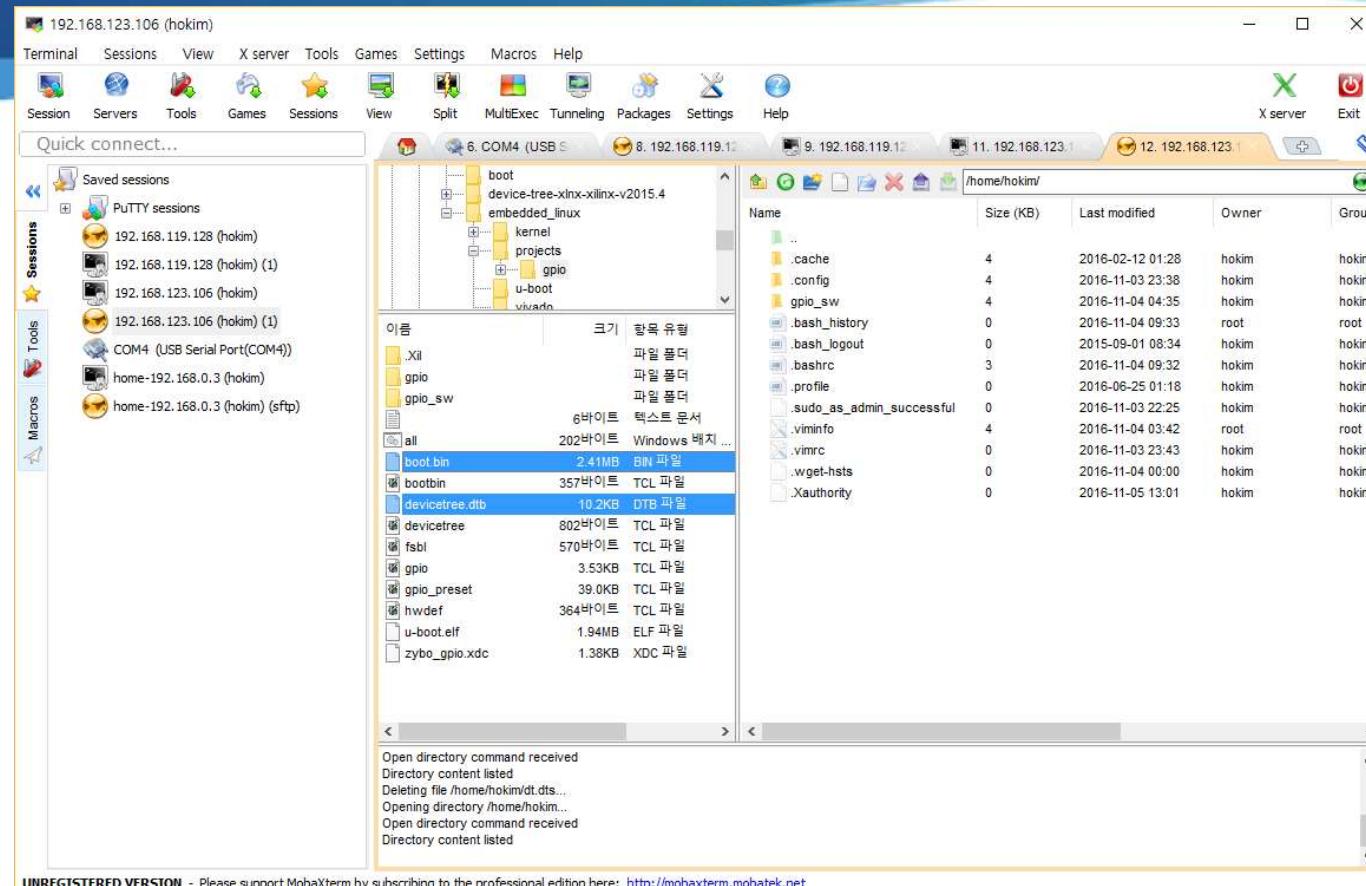
gpio/gpio.tree/pl.dtsi

```
/ {  
    amba_pl: amba_pl {  
        #address-cells = <1>;  
        #size-cells = <1>;  
        compatible = "simple-bus";  
        ranges ;  
        btrfs_4Bits: gpio@40001000 {  
            #gpio-cells = <2>;  
            compatible = "xlnx,xps-  
                gpio-1.00.a";  
            reg = <0x40001000  
            0x1000>;  
            .....  
            .....  
        };  
        leds_4Bits: gpio@40000000 {  
            #gpio-cells = <2>;  
            compatible = "xlnx,xps-  
                gpio-1.00.a";  
            reg = <0x40000000  
            0x1000>;  
            .....  
            .....  
        };  
    };  
};
```

```
sws_4Bits: gpio@40002000 {  
    #gpio-cells = <2>;  
    compatible = "xlnx,xps-  
        gpio-1.00.a";  
    reg = <0x40002000  
    0x1000>;  
    .....  
    .....  
};  
};
```

```
$ dtc -O dtb -l dts -i gpio/gpio.tree/ -o devicetree.dtb system_gpio.dts
```

Update boot.bin, devicetree.dtb for GPIO



login into zybo

```
$ sudo -s
# cd /boot
# rm boot.bin devicetree.dtb
# mv ~hokim/boot.bin .
# mv ~hokim/devicetree.dtb
# sync
# reboot
```

```
hokim@zybo:/sys/class/gpio$ ls  
export gpiochip894 gpiochip898 gpiochip902 gpiochip906 unexport
```

```
hokim@zybo:/sys/class/gpio$ ls gpiochip906  
base device label ngpio power subsystem uevent
```

```
hokim@zybo:/sys/class/gpio$ cat gpiochip906/base  
906
```

```
hokim@zybo:/sys/class/gpio$ cat gpiochip906/label  
zynq_gpio
```

```
hokim@zybo:/sys/class/gpio$ cat gpiochip906/ngpio  
118
```

```
hokim@zybo:/sys/class/gpio$ ls gpiochip894  
base label ngpio power subsystem uevent
```

```
hokim@zybo:/sys/class/gpio$ cat gpiochip894/base  
894
```

```
hokim@zybo:/sys/class/gpio$ cat gpiochip894/label  
/amba_pl/gpio@40002000
```

```
hokim@zybo:/sys/class/gpio$ cat gpiochip894/ngpio  
4
```

```
hokim@zybo:/sys/class/gpio$ cat gpiochip898/base  
898
```

```
hokim@zybo:/sys/class/gpio$ cat gpiochip898/label  
/amba_pl/gpio@40000000
```

```
hokim@zybo:/sys/class/gpio$ cat gpiochip898/ngpio  
4
```

```
hokim@zybo:/sys/class/gpio$ cat gpiochip902/base  
902
```

```
hokim@zybo:/sys/class/gpio$ cat gpiochip902/label  
/amba_pl/gpio@40001000
```

```
hokim@zybo:/sys/class/gpio$ cat gpiochip902/ngpio  
4
```



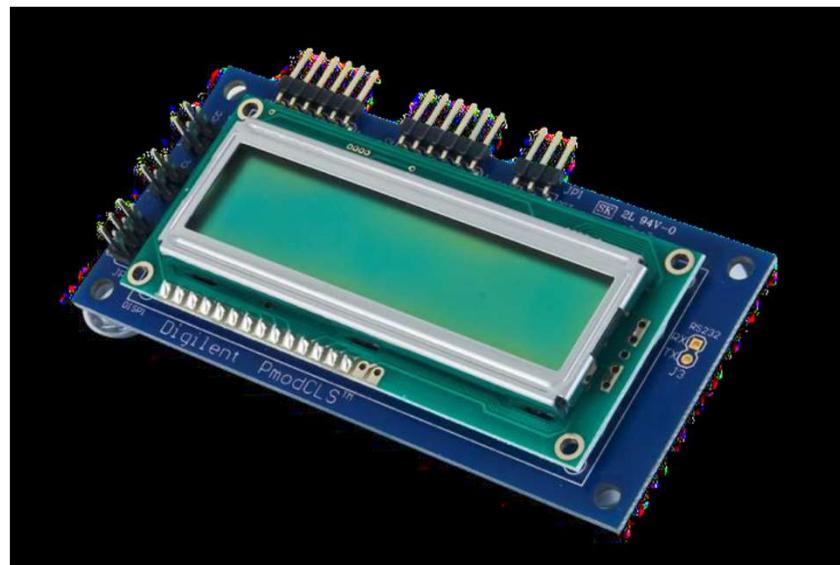
```
hokim@zybo:~$ cd gpio_sw/
hokim@zybo:~/gpio_sw$ ls
driver mmap
hokim@zybo:~/gpio_sw$ cd driver
hokim@zybo:~/gpio_sw/driver$ ls
CMakeLists.txt build gpio.c gpio.h main.c
hokim@zybo:~/gpio_sw/driver$ 
hokim@zybo:~/gpio_sw/driver$ mkdir build
hokim@zybo:~/gpio_sw/driver$ cd build
hokim@zybo:~/gpio_sw/driver/build$ cmake ..
hokim@zybo:~/gpio_sw/driver/build$ sudo ./gpio_test
```

```
hokim@zybo:~$ cd gpio_sw/driver/python
hokim@zybo:~/gpio_sw/driver/python$ ls
gpio.py gpio_test.py
hokim@zybo:~/gpio_sw/driver/python$ sudo python gpio_test.py
```

```
hokim@zybo:~$ cd gpio_sw
hokim@zybo:~/gpio_sw$ ls
driver mmap
hokim@zybo:~/gpio_sw$ cd mmap
hokim@zybo:~/gpio_sw/mmap$ ls
CMakeLists.txt lib main.c
hokim@zybo:~/gpio_sw/mmap$ mkdir build
hokim@zybo:~/gpio_sw/mmap$ cd build
hokim@zybo:~/gpio_sw/mmap/build$ cmake ..
hokim@zybo:~/gpio_sw/mmap/build$ make
hokim@zybo:~/gpio_sw/mmap/build$ sudo ./gpio_test
```



- 16x2 character LCD module
- Driven by ATmega48 microcontroller
- Multiple communication options including UART, SPI, and I2C



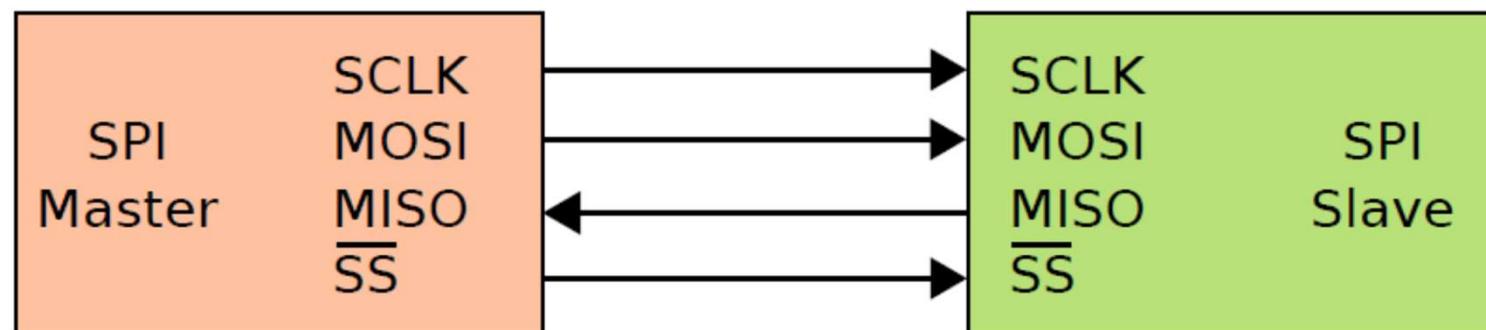
- Synchronous serial digital data link by Motorola
- SPI master controls

CLK : synchronization clock

SS or CS : slave select or chip-select; when active the slave is allowed to talk

MOSI : master out, slave in; carries data from master to slave

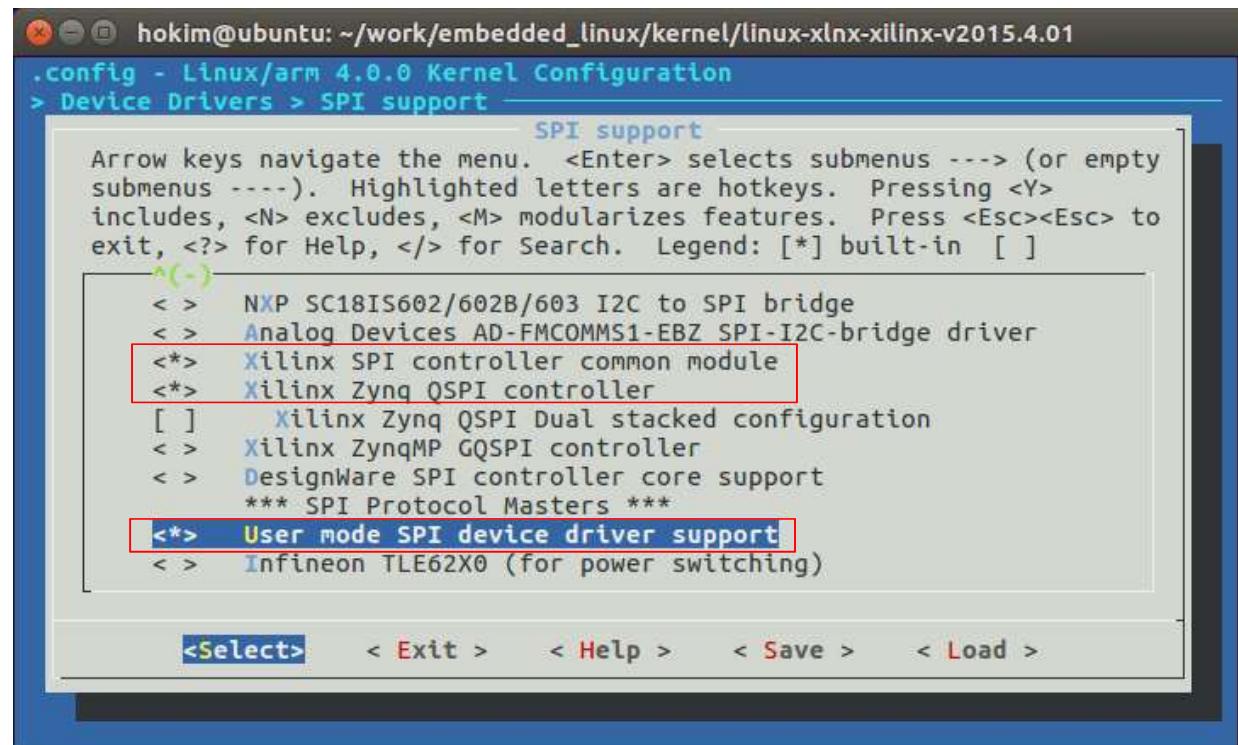
MISO : master in, slave out; carries data from slave to master



- Documented in Documentation/spi/spidev
- SPI kernel framework
- SPI master drivers : drivers/spi/(spi-zynq-qspi.c, spi-xilinx.c)
- Kernel platform code registers the “spidev” platform device
- Create character device nodes at /dev/spidevB.C where:
 - B is the SPI bus (master) number
 - C is the chip-select number of specific SPI slave.



Kernel Configuration



Re-customize IP

ZYNQ7 Processing System (5.5)

Documentation Presets IP Location Import XPS Settings

Page Navigator <>

Zynq Block Design

PS-PL Configuration

Peripheral I/O Pins

MIO Configuration

Clock Configuration

DDR Configuration

SMC Timing Calculation

Interrupts

MIO Configuration

Bank 0 I/O Voltage LVC MOS 3.3V Bank 1 I/O Voltage LVC MOS 1.8V

Search:

MIO Configuration

Peripheral	IO	Signal	IO Type	Speed	Pullup	Direction
Memory Interfaces						
I/O Peripherals						
ENET 0	MIO 16 .. 27					
ENET 1						
USB 0	MIO 28 .. 39					
USB 1						
SD 0	MIO 40 .. 45					
SD 1						
UART 0						
UART 1	MIO 48 .. 49					
I2C 0	EMIO					
I2C 1						
SPI 0	EMIO					
SS[0] IO	EMIO					
SS[1] IO	EMIO					
SS[2] IO	EMIO					
SPI 1						
CAN 0						

OK Cancel

The screenshot shows the MIO Configuration dialog box for a Zynq7 PS Re-customize IP. The 'MIO Configuration' tab is selected in the left sidebar. The main area displays a table of I/O peripherals and their assigned MIO ranges. A red box highlights the SPI 0 configuration row, which includes three SS[0], SS[1], and SS[2] IO entries, all of which are checked. Other peripherals listed include ENET 0, USB 0, SD 0, UART 0, I2C 0, and CAN 0.

Zynq7 PS Re-customize IP for SPI

Re-customize IP

ZYNQ7 Processing System (5.5)

Documentation Presets IP Location Import XPS Settings

Page Navigator <> Zynq Block Design

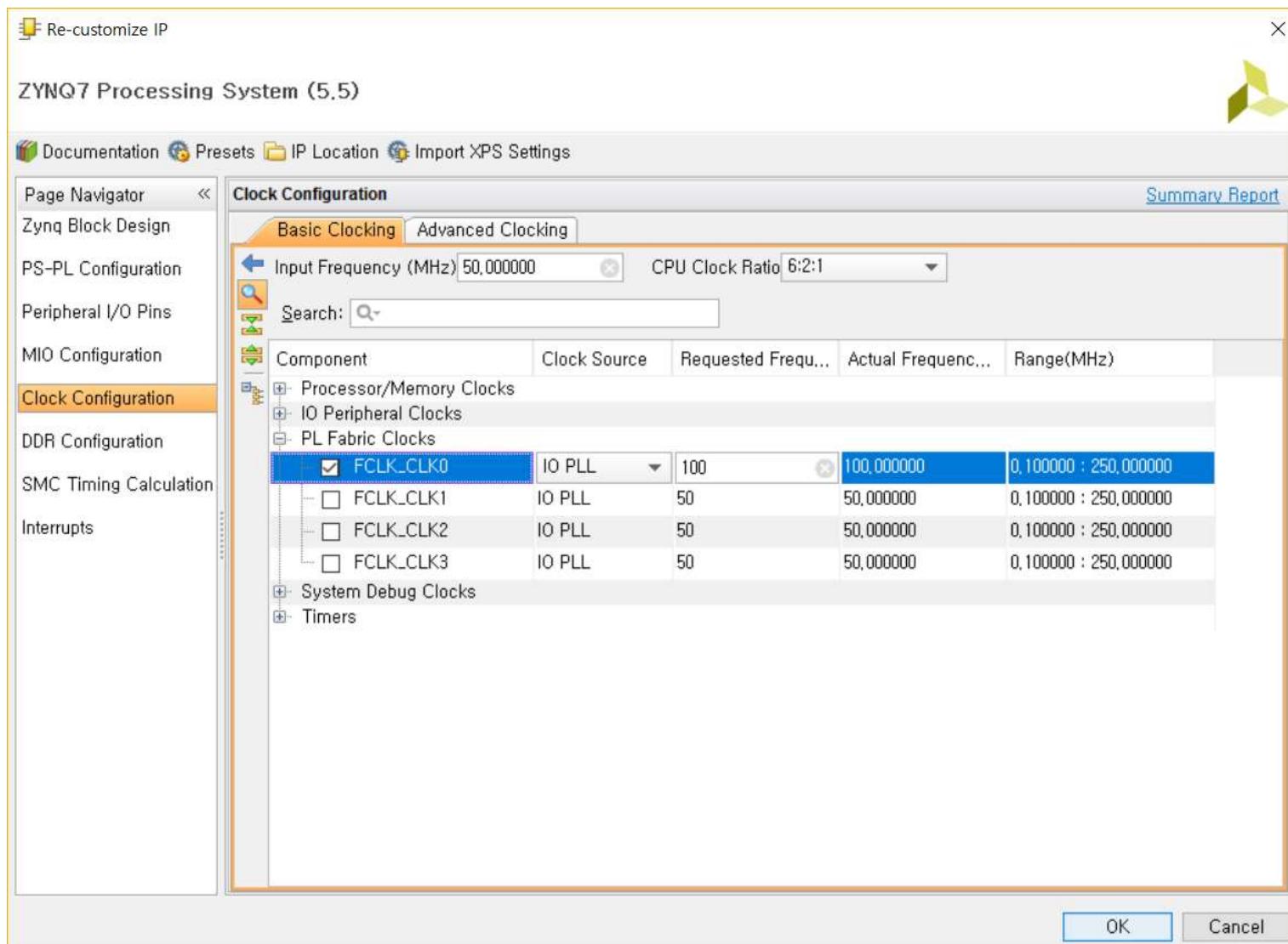
PS-PL Configuration [Summary Report](#)

Search:

Name	Select	Description
FTM Trace buffer FIFO size	128	FTM Trace buffer FIFO size
FTM Trace buffer clock delay	12	Number of clock cycles interval for a trace data output from FIFO.
Include ACP transaction checker	<input type="checkbox"/>	Enables ACP transaction checker.
Trace data/control signal pipeline width	8	Enables configurable number of pipeline stages on the TRACE
Power-on reset(POR) 4k timer	<input type="checkbox"/>	Enables power-on reset(POR) 4k timer. By default, 64k timer is
Processor event interface	<input type="checkbox"/>	Enables event bus which provides a low-latency and direct me
Address Editor		
Enable Clock Triggers		
Enable Clock Resets		
FCLK_RESET0_N	<input checked="" type="checkbox"/>	Enables general purpose reset signal 0 for PL logic
FCLK_RESET1_N	<input type="checkbox"/>	Enables general purpose reset signal 1 for PL logic
FCLK_RESET2_N	<input type="checkbox"/>	Enables general purpose reset signal 2 for PL logic
FCLK_RESET3_N	<input type="checkbox"/>	Enables general purpose reset signal 3 for PL logic
AXI Non Secure Enable	0	Enable AXI Non Secure Transaction
GP Master AXI Interface		
M AXI GP0 Interface	<input checked="" type="checkbox"/>	Enables General purpose AXI master interface 0
M AXI GP1 Interface	<input type="checkbox"/>	Enables General purpose AXI master interface 1
GP Slave AXI Interface		
HP Slave AXI Interface		
ACP Slave AXI Interface		
DMA Controller		

OK Cancel

Zynq7 PS Re-customize IP for SPI



Zynq7 PS Re-customize IP for SPI

Re-customize IP

ZYNQ7 Processing System (5.5)

Documentation Presets IP Location Import XPS Settings

Page Navigator << Summary Report

Interrupts

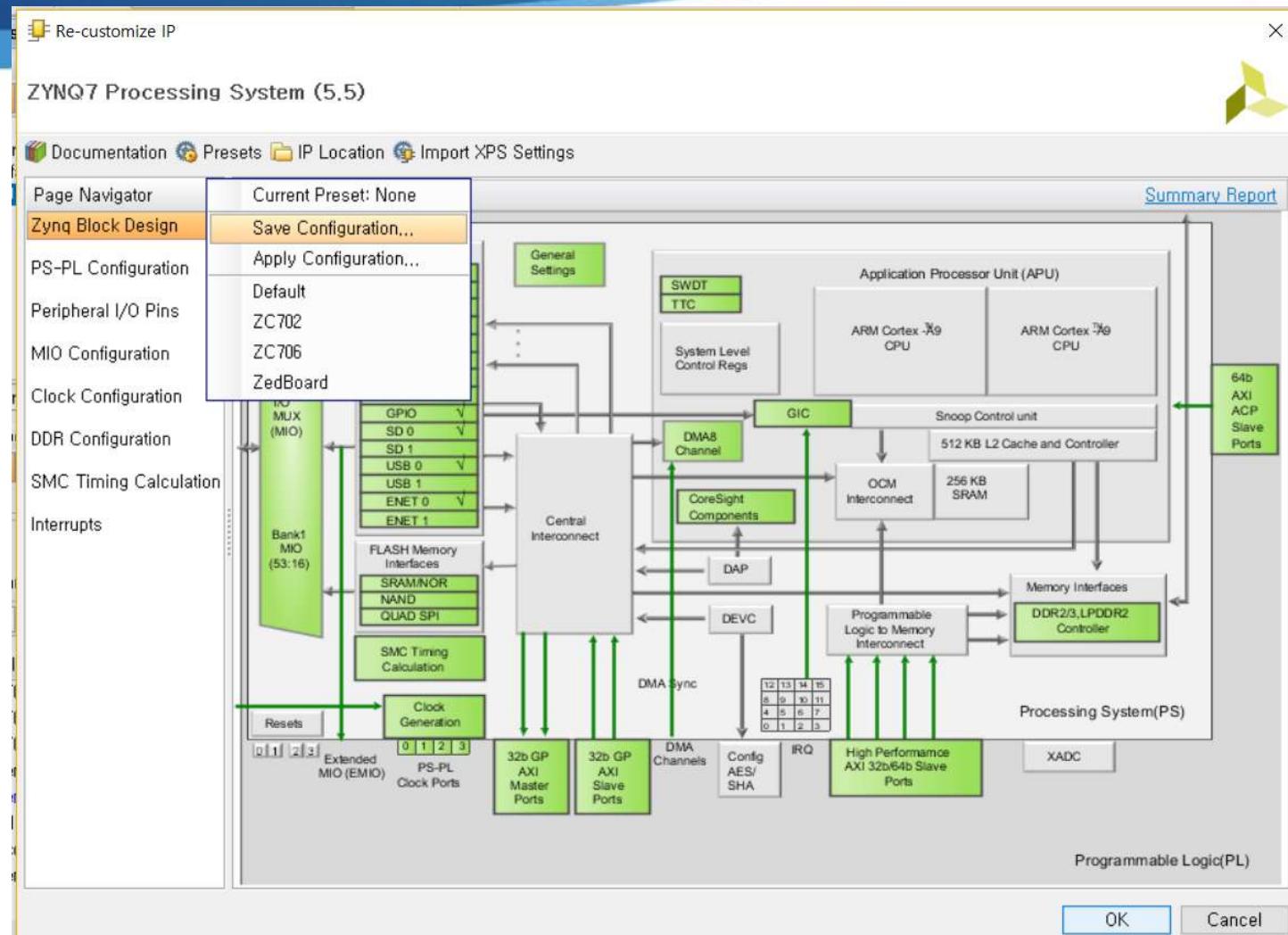
Search:

Interrupt Port	ID	Description
<input checked="" type="checkbox"/> Fabric Interrupts		Enable PL Interrupts to PS and vice versa
<input checked="" type="checkbox"/> PL-PS Interrupt Ports		
<input checked="" type="checkbox"/> IRQ_F2P[15:0]	[91:84], [6..	Enables 16-bit shared interrupt port from the PL, MSB is assigned to...
<input type="checkbox"/> Core0_nFIQ	28	Enables fast private interrupt signal for CPU0 from the PL
<input type="checkbox"/> Core0_nIRQ	31	Enables private interrupt signal for CPU0 from the PL
<input type="checkbox"/> Core1_nFIQ	28	Enables fast private interrupt signal for CPU1 from the PL
<input type="checkbox"/> Core1_nIRQ	31	Enables private interrupt signal for CPU1 from the PL
<input checked="" type="checkbox"/> PS-PL Interrupt Ports		

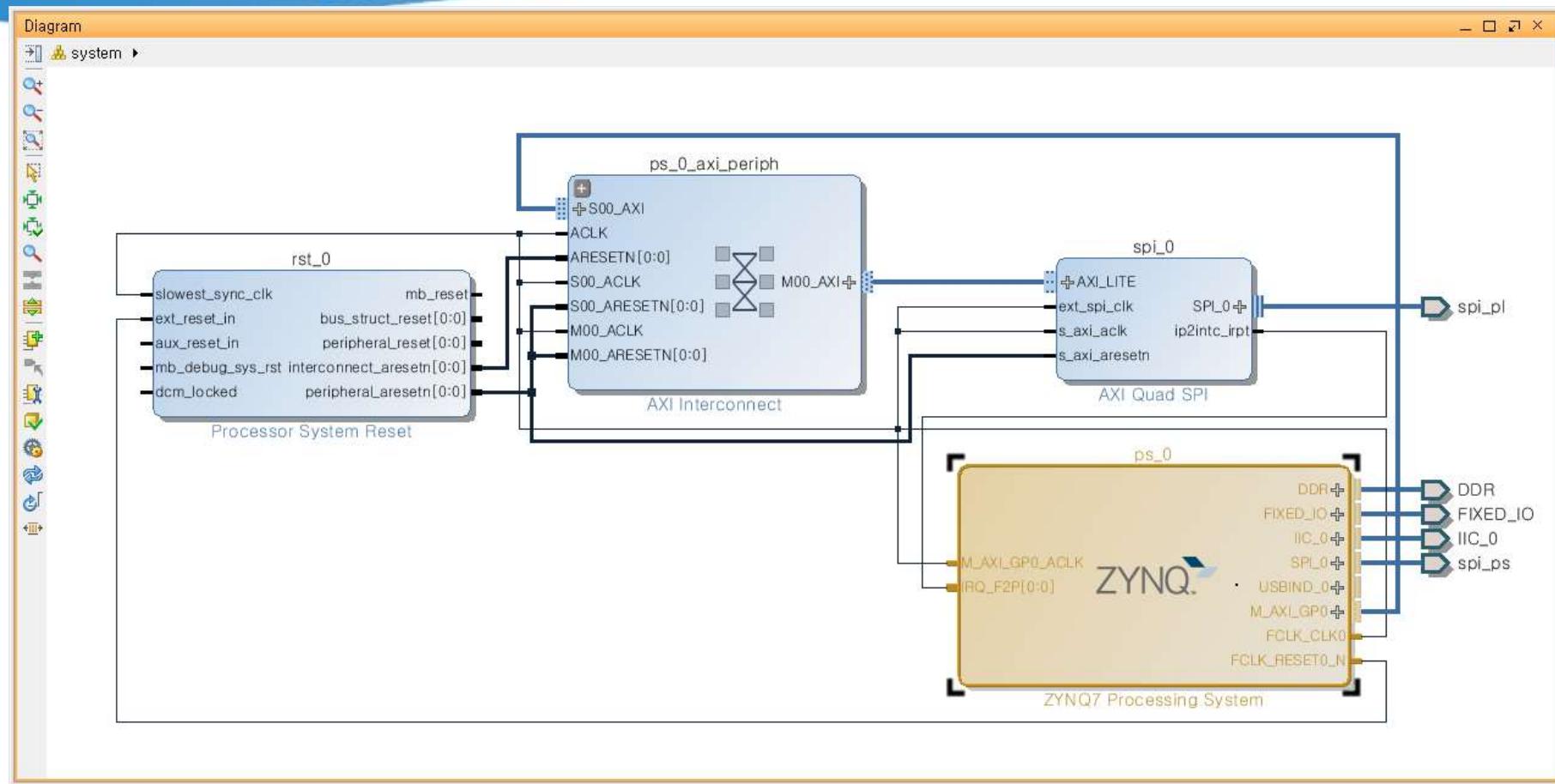
OK Cancel



Zynq7 PS Re-customize IP for SPI



IP Integrator Diagram for SPI



Address Editor

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
ps_0 Data (32 address bits : 0x40000000 [1G]) spi_0	AXI_LITE	Reg	0x4000_0000	4K	0x4000_0FFF

IP Integrator Diagram for SPI

Re-customize IP

ZYNQ7 Processing System (5.5)

Documentation Presets IP Location Import XPS Settings

Page Navigator Current Preset: None

Zynq Block Design

PS-PL Configuration Save Configuration... Apply Configuration...

Peripheral I/O Pins Default ZC702 ZC706 ZedBoard

MIO Configuration

Clock Configuration

DDR Configuration

SMC Timing Calculation

Interrupts

Bank1 MIO (53:16)

GPIO Y SD 0 Y SD 1 Y USB 0 Y USB 1 Y ENET 0 Y ENET 1 Y

FLASH Memory Interface SRAM/NOR NAND QUAD SPI

SMC Timing Calculation

Resets

Extended MIO (EMIO) [1][2][3]

PS-PL Clock Ports [9][1][2][3]

Central Interconnect

CoreSight Components

SWD/TTC System Level Control Regs

GIC

ARM Cortex-A9 CPU ARM Cortex-A9 CPU

Snoop Control Unit 512 KB L2 Cache and Controller 256 KB SRAM

DMA Channel

DAP

OCM Interconnect

DEVC

Programmable Logic to Memory Interconnect

Memory Interfaces DDR2/3/LPDDR2 Controller

Processing System(PS)

XADC

64b AXI ACP Slave Ports

DMA Sync

32b GP AXI Master Ports

DMA Channels

Config AES/ SHA

IRQ

High Performance AXI 32b/64b Slave Ports

OK Cancel

Look in:

Recent Directories: C:/Users/hokim/work/embedded_linux/projects/spi

File Preview:

```
1 proc getPresetInfo {} {
2     return [dict create name {spi} description {spi} vlnv xi
3 }
4
5 proc validate_preset {PINST} { return true }
6
7
8 proc apply_preset {PINST} {
9     return [dict create #
10    CONFIG16.PCI_DDR_RAM_BASEADDR {0x00100000} #
11    CONFIG16.PCI_DDR_RAM_HIADDR {0xFFFFFFF} #
12    CONFIG16.PCI_UART0_BASEADDR {0xE0000000} #
13    CONFIG16.PCI_UART0_HIADDR {0xE0000FFF} #
14    CONFIG16.PCI_UART1_BASEADDR {0xE0001000} #
15    CONFIG16.PCI_UART1_HIADDR {0xE0001FFF} #
16    CONFIG16.PCI_I2CO_BASEADDR {0xE0004000} #
17    CONFIG16.PCI_I2CO_HIADDR {0xE0004FFF} #
18    CONFIG16.PCI_I2C1_BASEADDR {0xE0005000} #
19    CONFIG16.PCI_I2C1_HIADDR {0xE0005FFF} #
20    CONFIG16.PCI_SPI10_BASEADDR {0xE0006000} #
21    CONFIG16.PCI_SPI10_HIADDR {0xE0006FFF} #
}
```

File name:

Files of type: Preset Tcl Files (*.tcl)

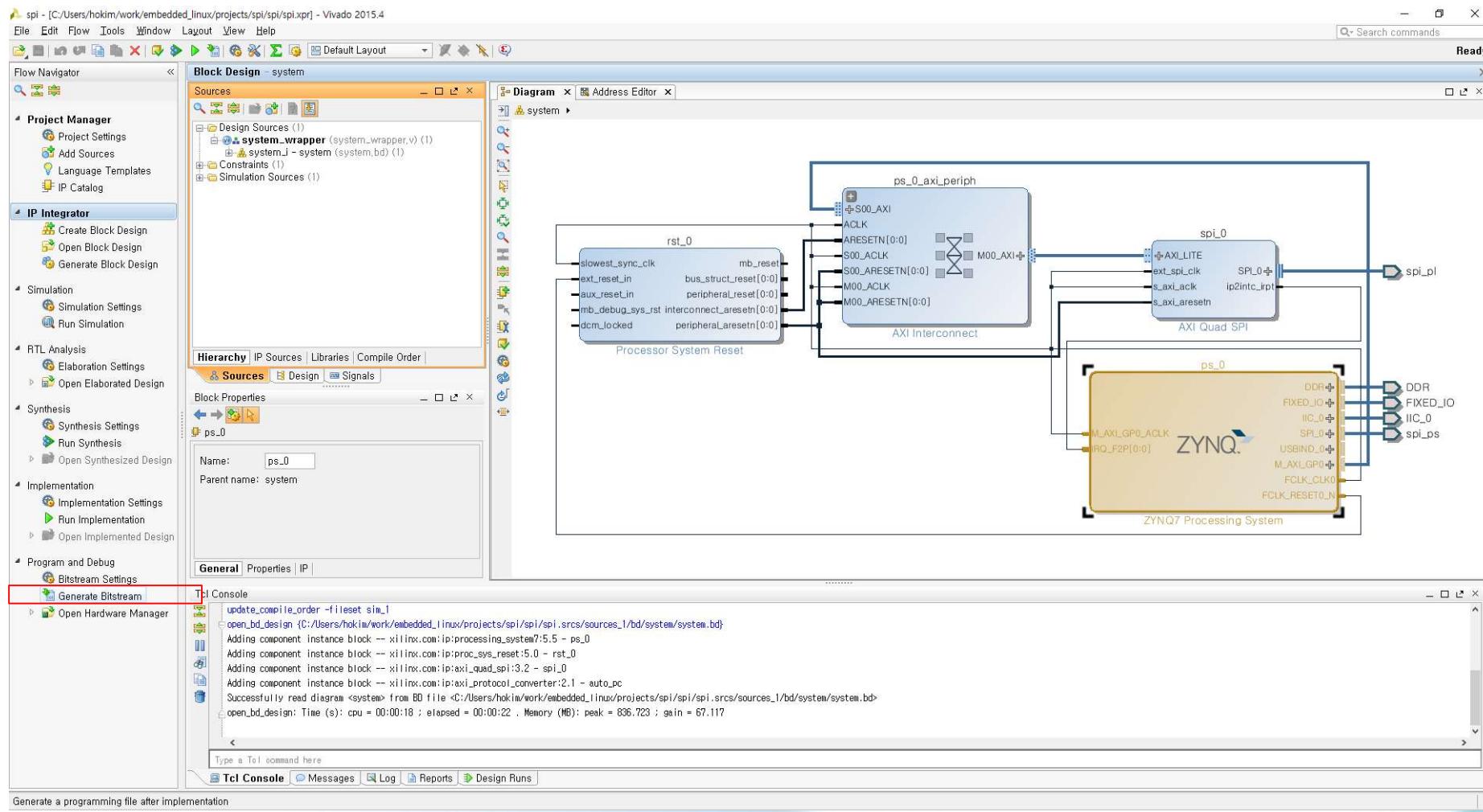
OK Cancel

https://github.com/inipro/embedded_linux/projects/spi

In cmd window for Vivado

```
C:\ cd C:\Users\hokim\work\embedded_linux\projects\spi\spi.xpr  
C:\ vivado -nolog -nojournal -mode batch -source spi.tcl
```

output : spi\spi.xpr...



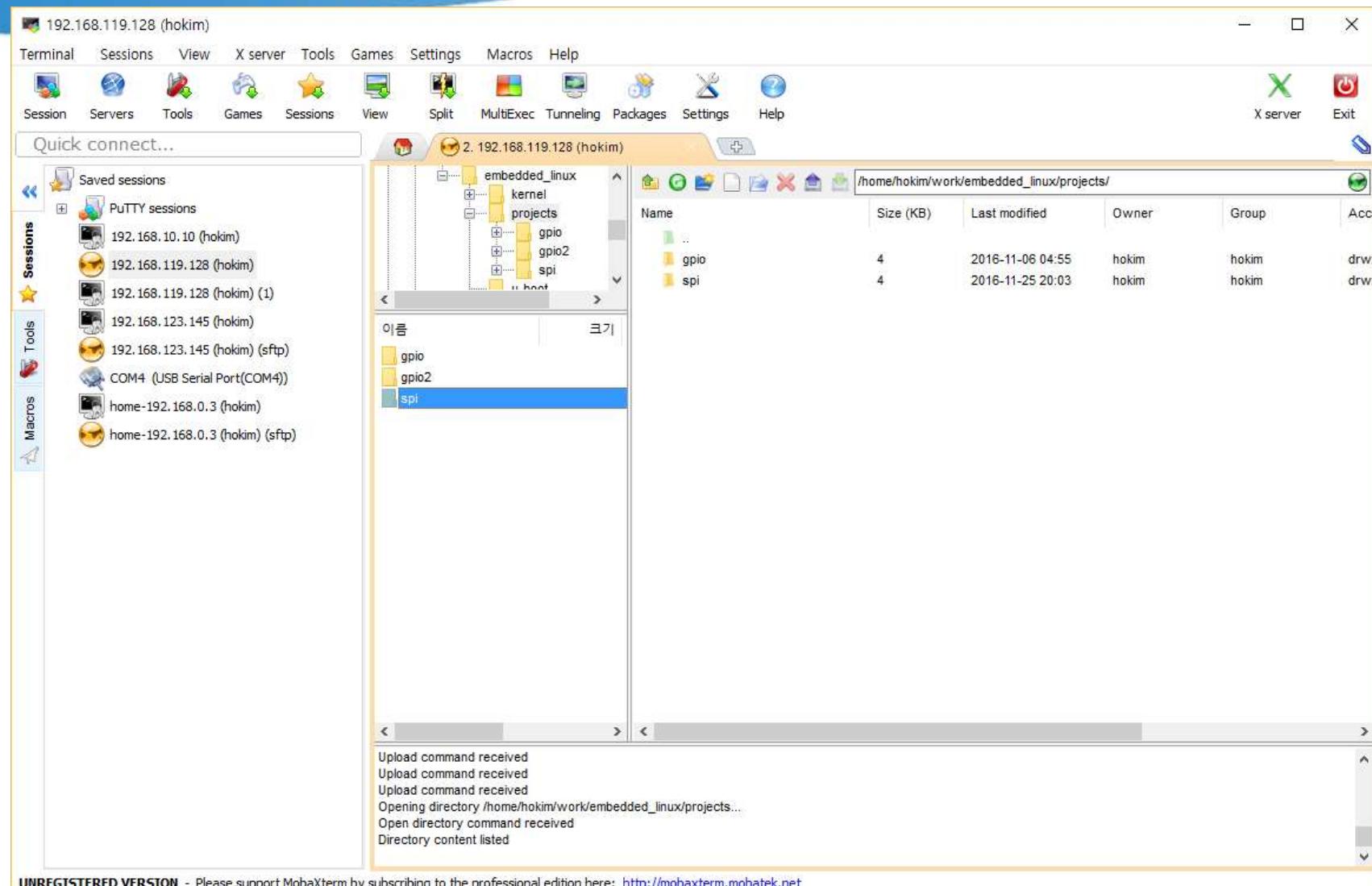
C:\Users\hokim\work\embedded_linux\projects\spi\all.bat

```
call vivado -nolog -nojournal -mode batch -source hwdef.tcl  
call hsi -nolog -nojournal -mode batch -source fsbl.tcl  
call tclsh bootbin.tcl  
  
call hsi -nolog -nojournal -mode batch -source devicetree.tcl
```

```
C:\ cd C:\Users\hokim\work\embedded_linux\projects\spi  
C:\ all
```



Device Tree Compile for SPI



```
$ cd ~/work/embedded_linux/projects
$ cp spi/spi.tree/system.dts  system_spi.dts
$ nano system_spi.dts
```

Device Tree Compile for SPI

system_spi.dts

```
/dts-v1/;
/include/ "zynq-7000.dtsi"
/include/ "pl.dtsi"
.....
.....
&clkc {
    fclk-enable = <0x1>;
    ps-clk-frequency = <50000000>;
};

+&gem0 {
+    phy-handle = <&phy0>;
+    ps7_ethernet_0_mdio: mdio {
+        #address-cells = <0x1>;
+        #size-cells = <0x0>;
+        phy0: phy@0 {
+            compatible = "realtek,RTL8211E";
+            device_type = "ethernet-phy";
+            reg = <0>;
+        };
+    };
+};
+&i2c0 {
+    eeprom@50 {
+        /* Microchip 24AA02E48 */
+        compatible = "microchip,24c02";
+        reg = <0x50>;
+        pagesize = <8>;
+    };
+};
```

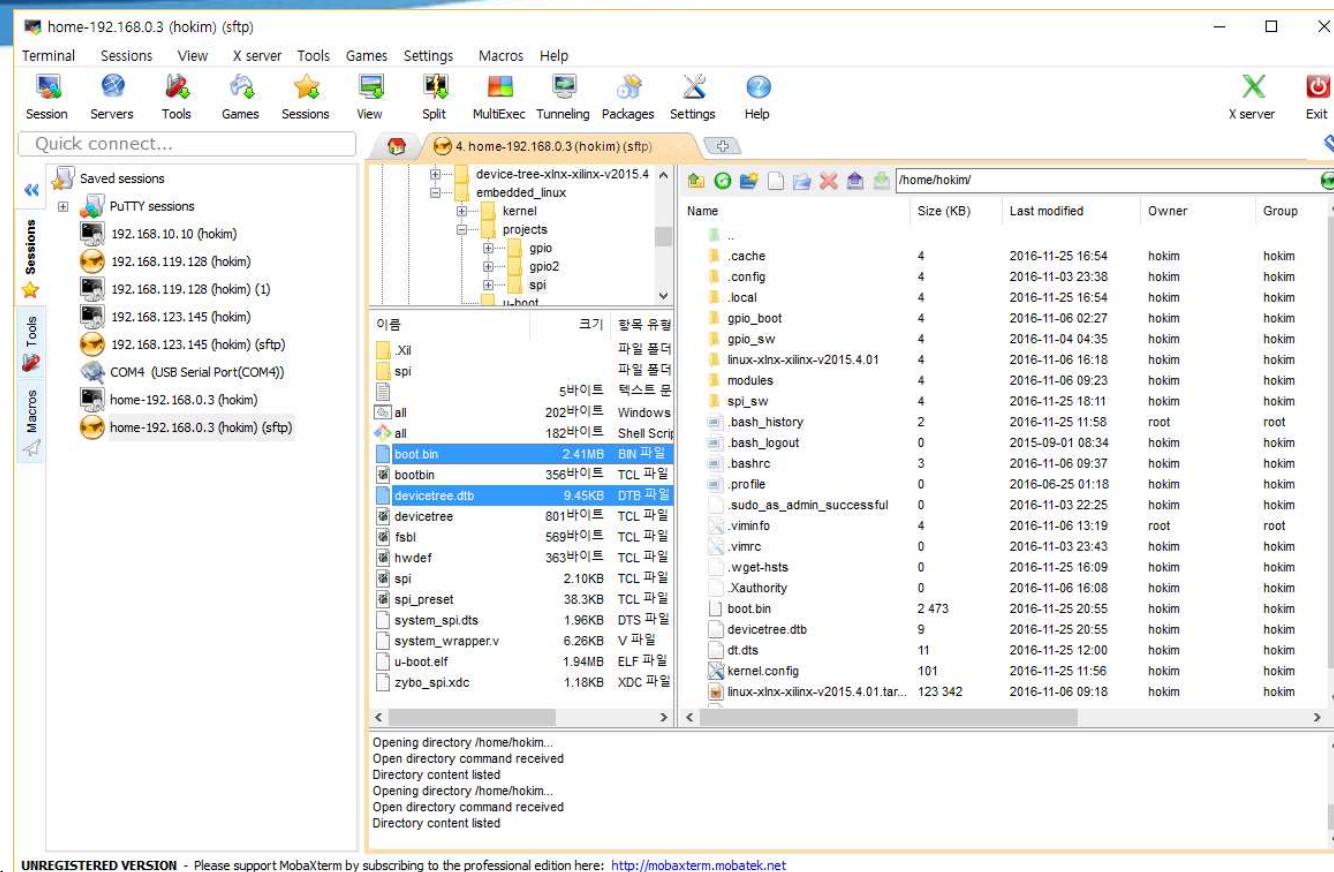
```
+/ {
+    usb_phy0: phy0 {
+        compatible = "ulpi-phy";
+        #phy-cells = <0>;
+        reg = <0xe0002000 0x1000>;
+        view-port = <0x0170>;
+        drv-vbus;
+    };
+};
+&usb0 {
+    usb-phy = <&usb_phy0>;
+};
+&spi0 {
+    num-cs = <1>;
+
+    spidev@0 {
+        compatible = "spidev";
+        reg = <0>; // chipselect 0
+        spi-max-frequency = <50000000>;
+    };
+};
+
+&spi_0 {
+    #address-cells = <1>;
+    #size-cells = <0>;
+
+    spidev@0 {
+        compatible = "spidev";
+        reg = <0>; // chipselect 0
+        spi-max-frequency = <50000000>;
+    };
+};
```

spi/spi.tree/pl.dtsi

```
/ {  
    amba_pl: amba_pl {  
        #address-cells = <1>;  
        #size-cells = <1>;  
        compatible = "simple-bus";  
        ranges ;  
        spi_0: axi_quad_spi@40000000 {  
            compatible = "xlnx,xps-spi-2.00.a";  
            interrupt-parent = <&intc>;  
            interrupts = <0 29 1>;  
            reg = <0x40000000 0x1000>;  
            xlnx,num-ss-bits = <0x1>;  
        };  
    };  
};
```

```
$ dtc -O dtb -l dts -i spi/spi.tree/ -o devicetree.dtb system_spi.dts
```

Update boot.bin, devicetree.dtb for SPI



login into zybo

```
$ sudo -s
# cd /boot
# rm boot.bin devicetree.dtb
# mv ~hokim/boot.bin .
# mv ~hokim/devicetree.dtb
# sync
# reboot
```

```
hokim@zybo:~$ ls /dev/spidev*
/dev/spidev0.0 /dev/spidev1.0
```

```
hokim@zybo:~$ cd spi_sw/
hokim@zybo:~/spi_sw$ ls
CMakeLists.txt PmodCLS.c PmodCLS.h main.c spi.c spi.h
hokim@zybo:~/spi_sw$ mkdir build
hokim@zybo:~/spi_sw$ cd build
hokim@zybo:~/spi_sw/build$ cmake ..
hokim@zybo:~/spi_sw/build$ make
hokim@zybo:~/spi_sw/build$ sudo ./spi_test
```

