INIPRO

Embedded Linux on Zynq

김현옥 (hokim1972@naver.com)

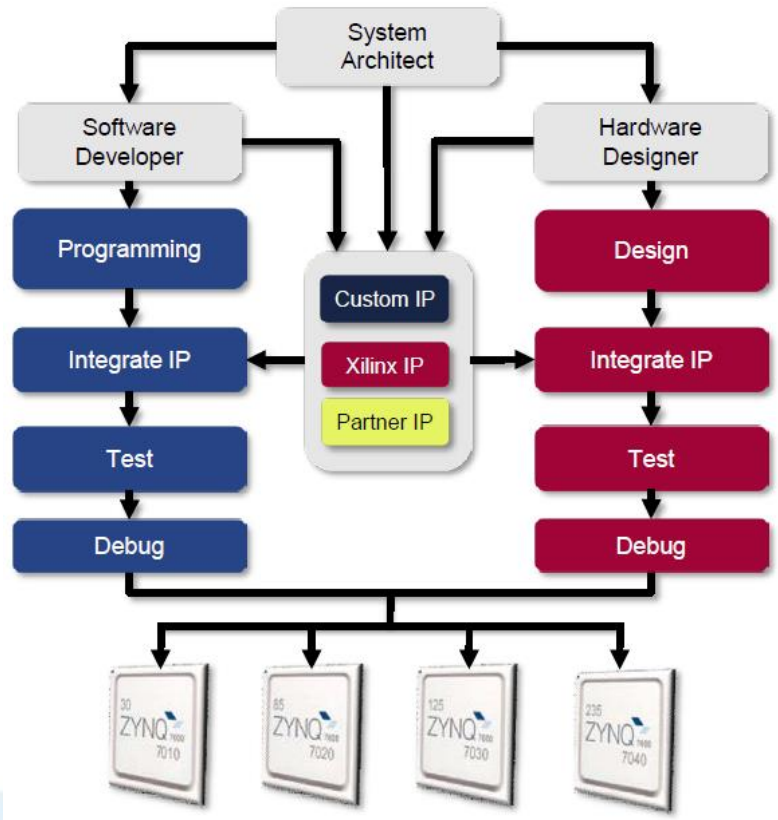
Zynq

- **Zynq = Dual core ARM (PS: Processing System) + FPGA (PL: Programmable Logic)**
- **Processor concentrated chip**

  Processor runs with minimal(?) PL programming.

  Less flexibility / Higher Performance

More efficient parallel hardware and software development



Zynq PS Software Development

INIPRO

- **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

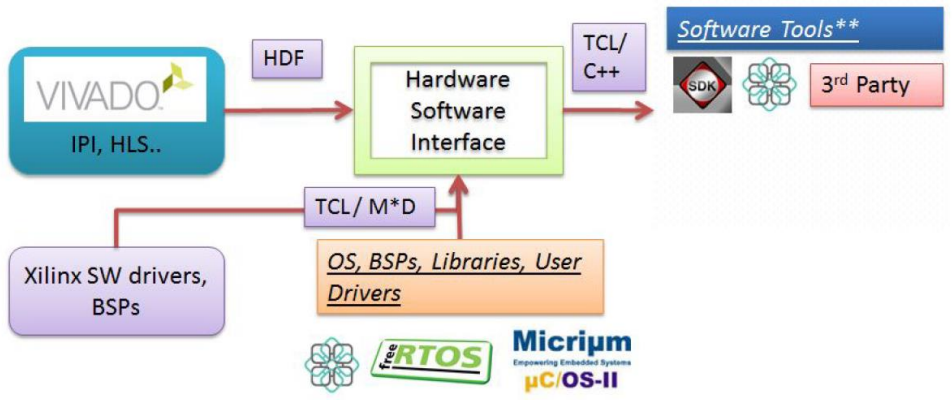
  Most are commercial

- **Linux**

  Greatest flexibility

  Many drivers, libraries and software package

Development Tool



- **Vivado** GUI, CUI(2016.4)

  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

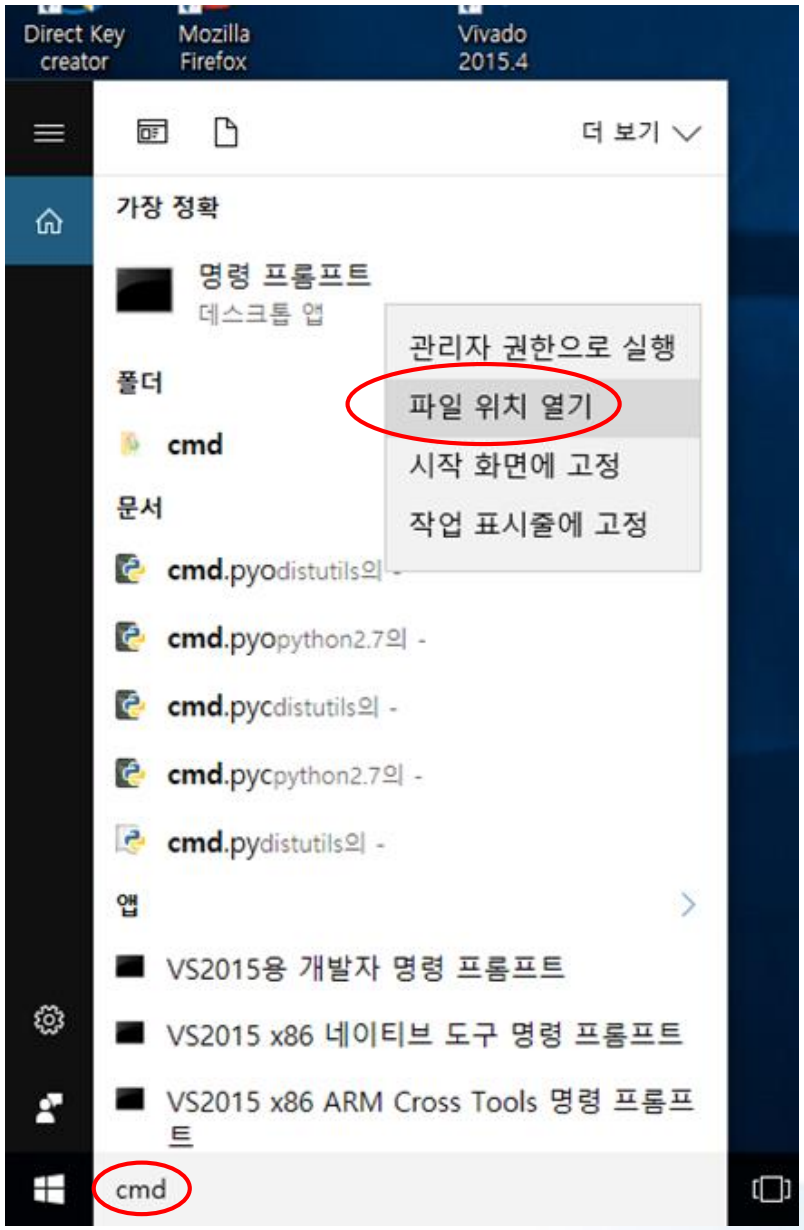
  (Target) Compile drivers / applications

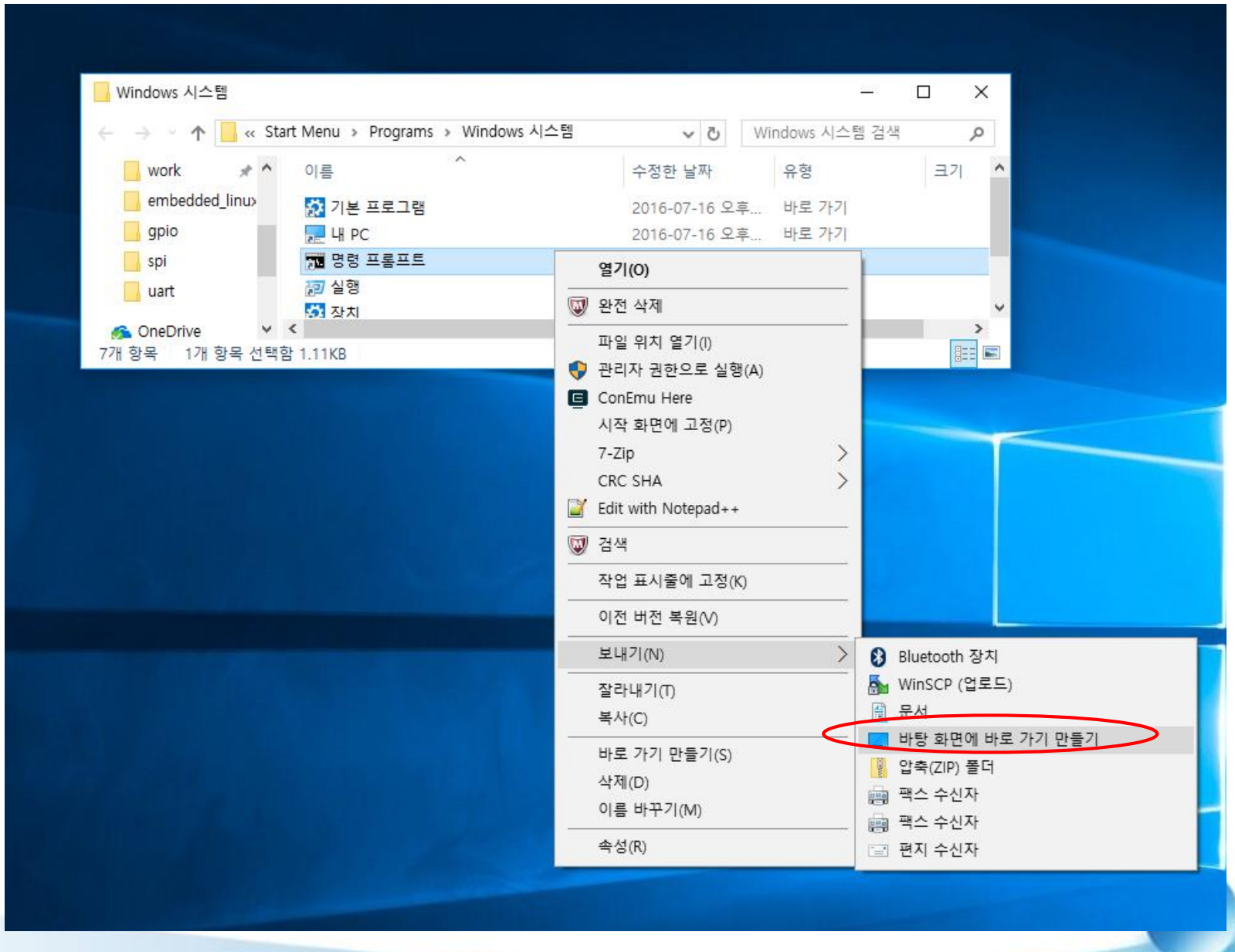
Development Environment Set-up

- **Windows** 10

  Install Vivado 2016.4 w/ SDK

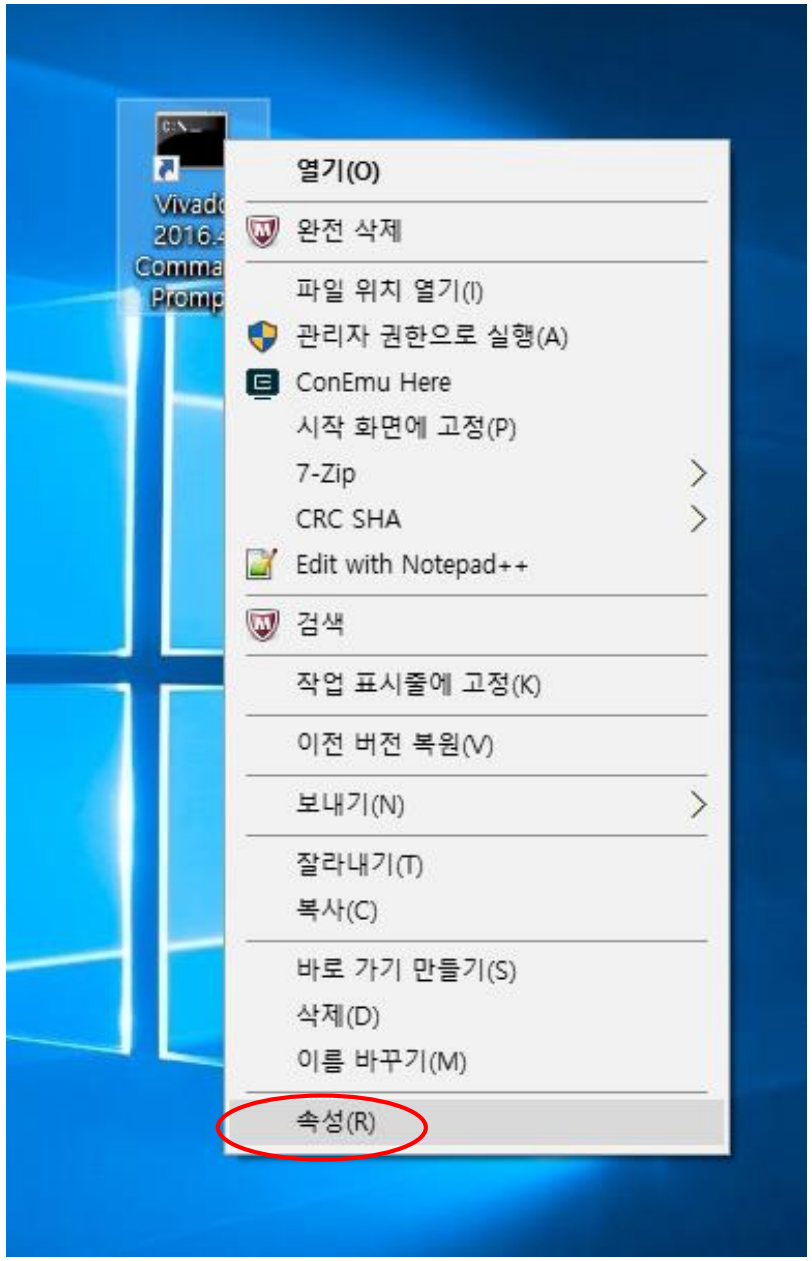
  cmd terminal for Vivado





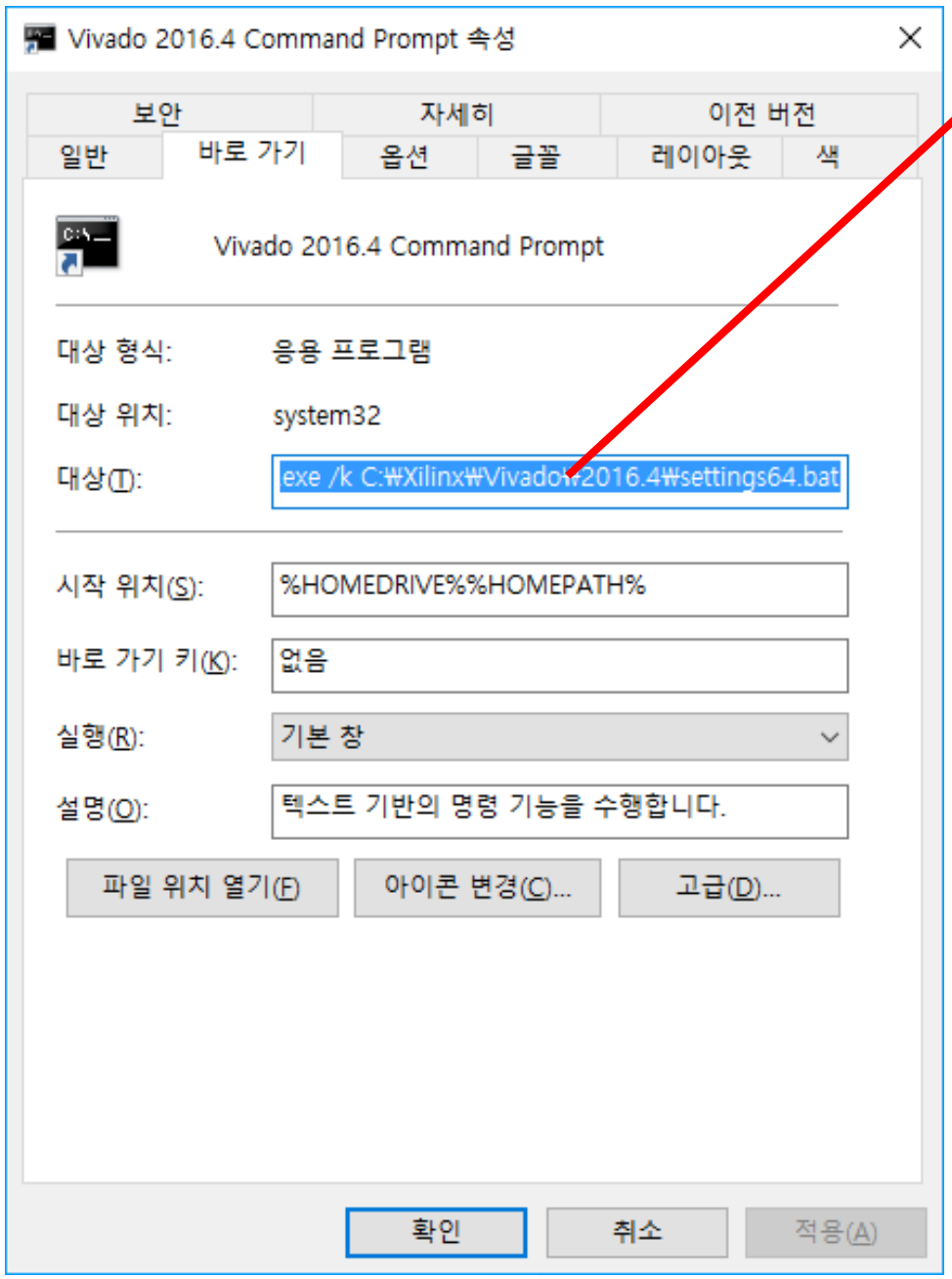
Development Environment Set-up

Vivado 2016.4 Command Prompt



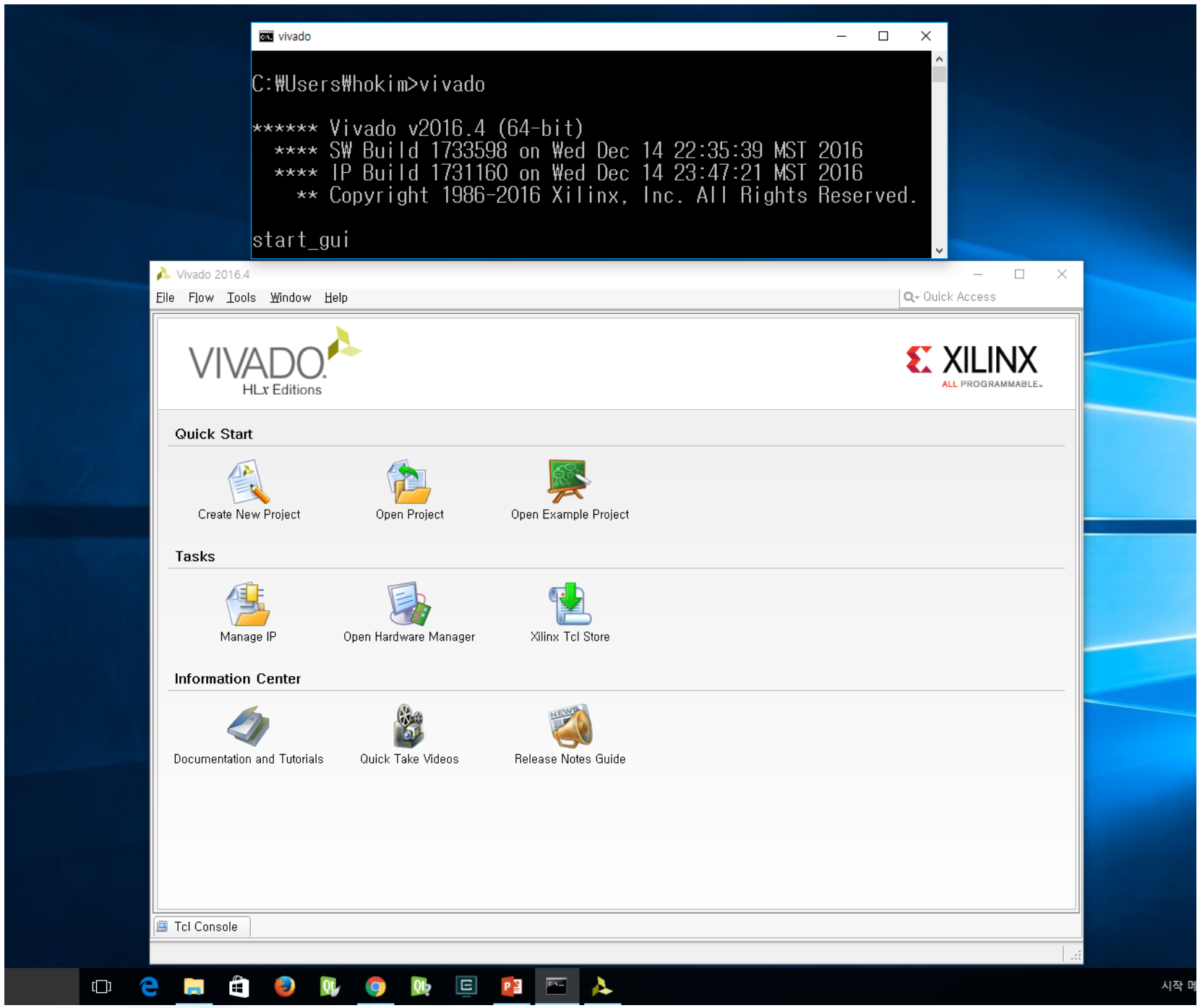
Development Environment Set-up

%windir%\system32\cmd.exe /k C:\Xilinx\Vivado\2016.4\settings64.bat



Development Environment Set-up

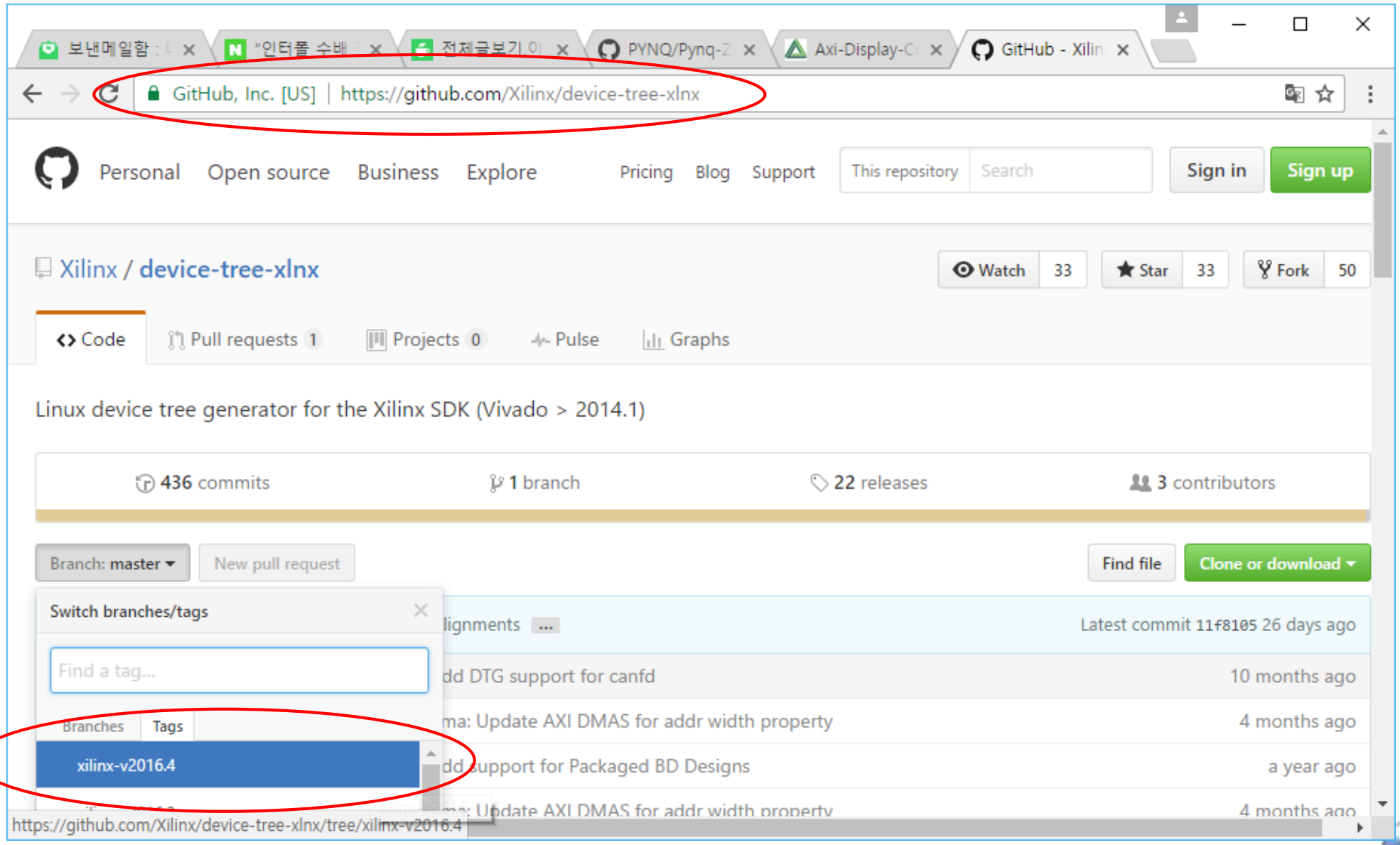
- **cmd terminal** for Vivado.



Development Environment Set-up

INIPRO

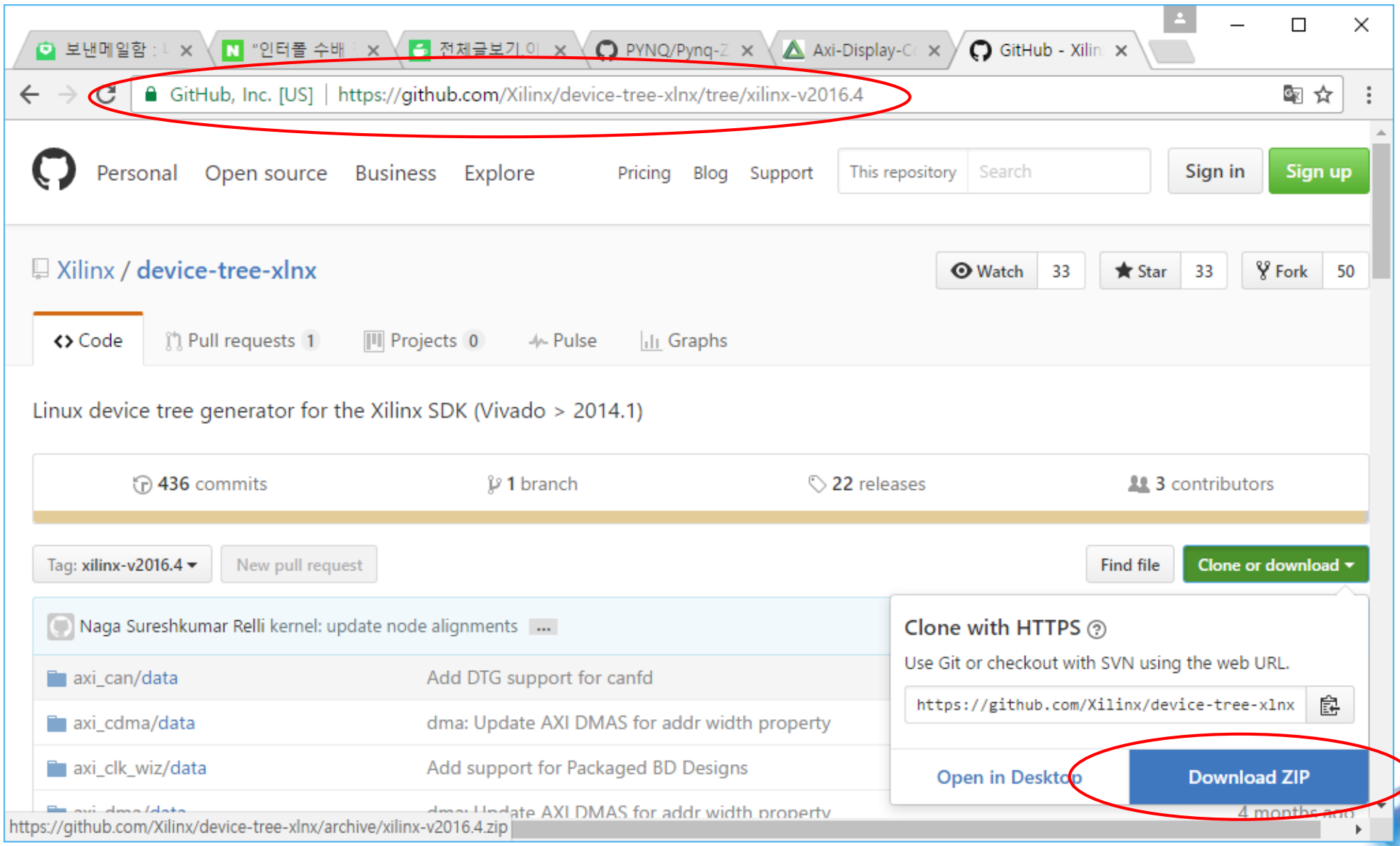
Device Tree Source



Development Environment Set-up

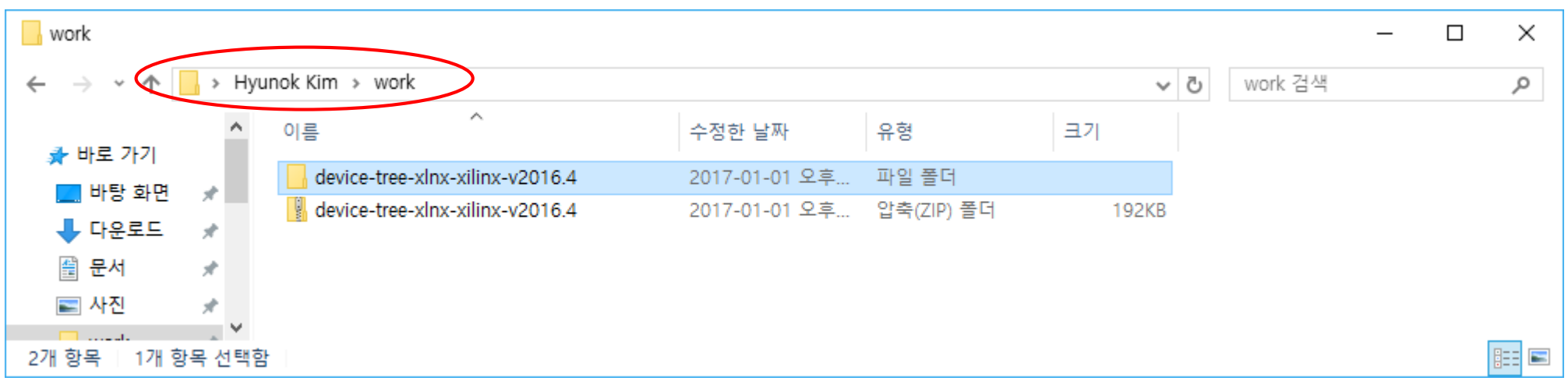
INIPRO

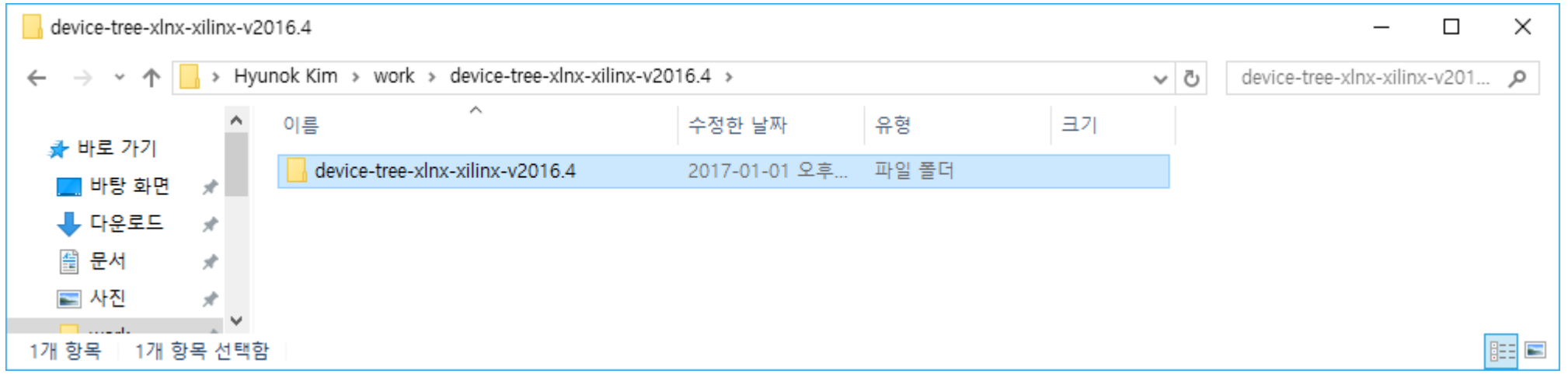
Device Tree Source

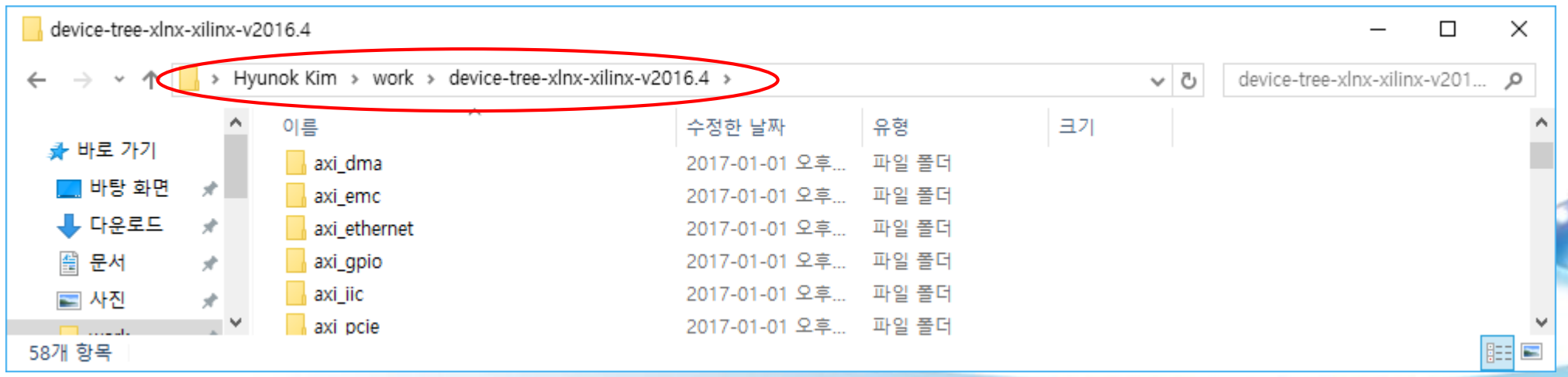


Development Environment Set-up

Device Tree Source



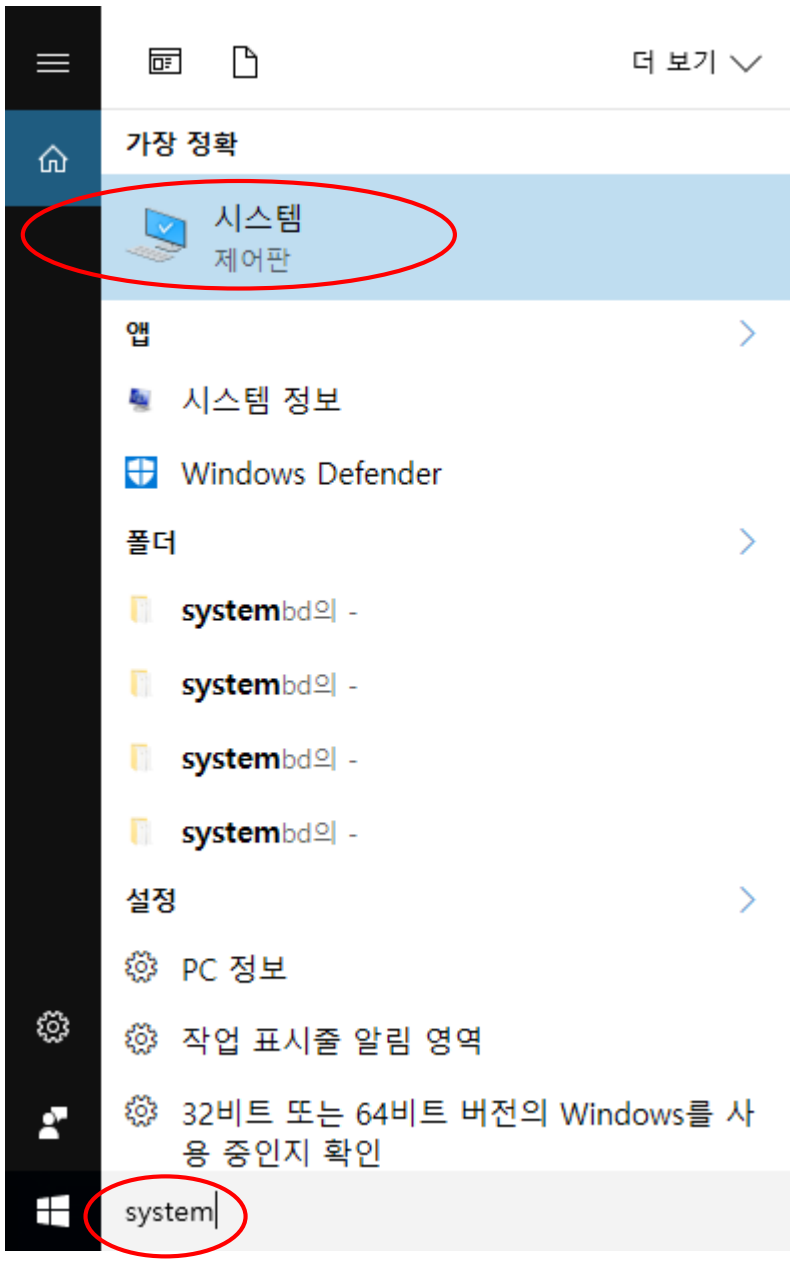


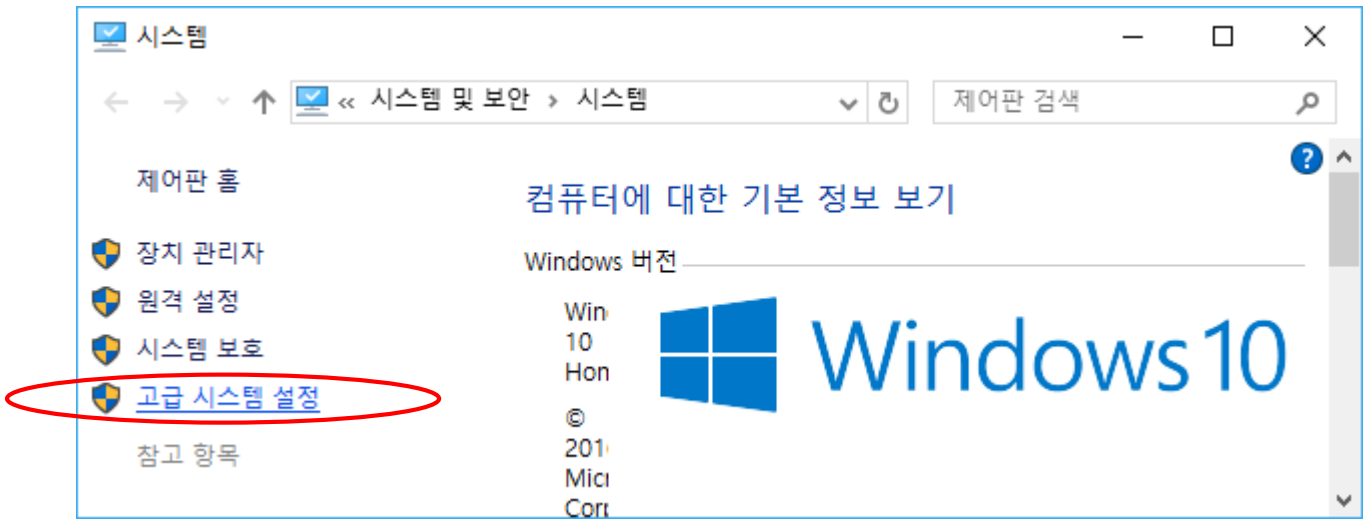


Development Environment Set-up

INIPRO

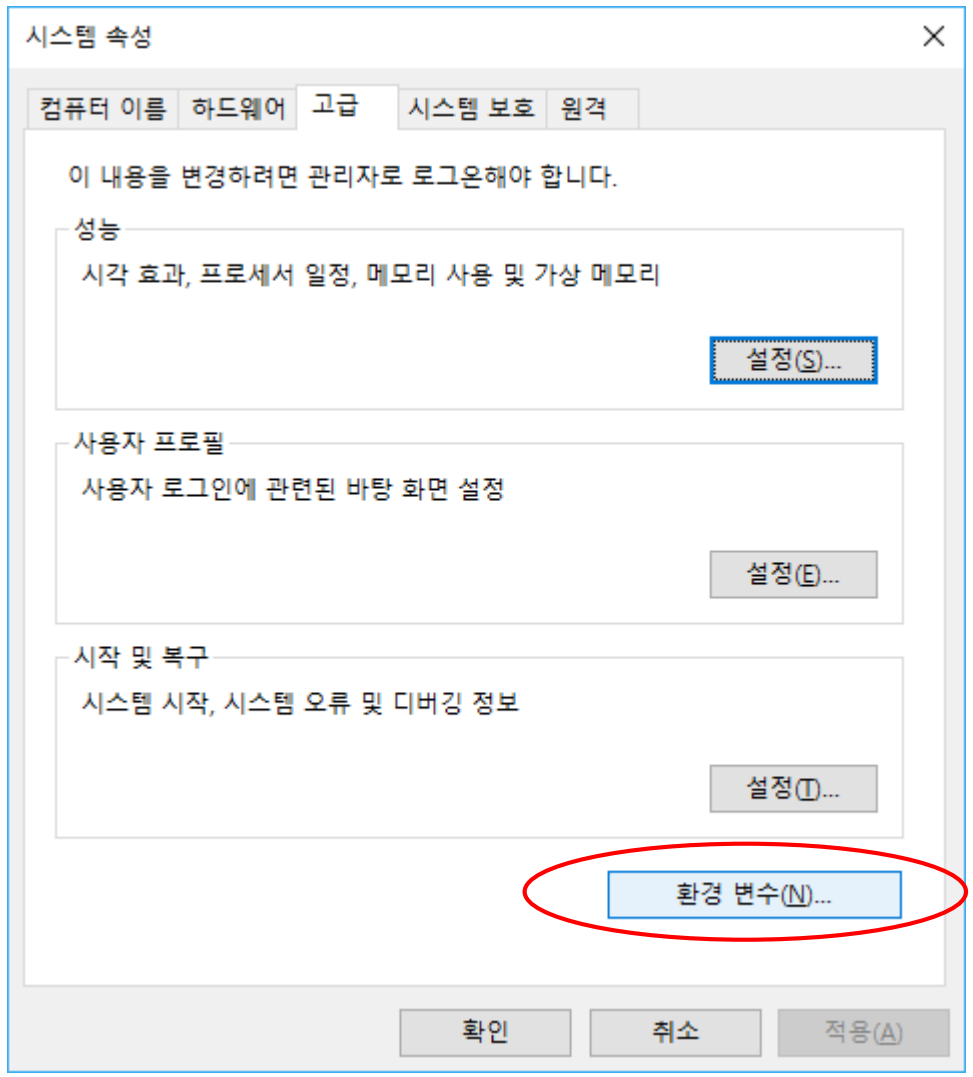
Environment Variable(HOME)

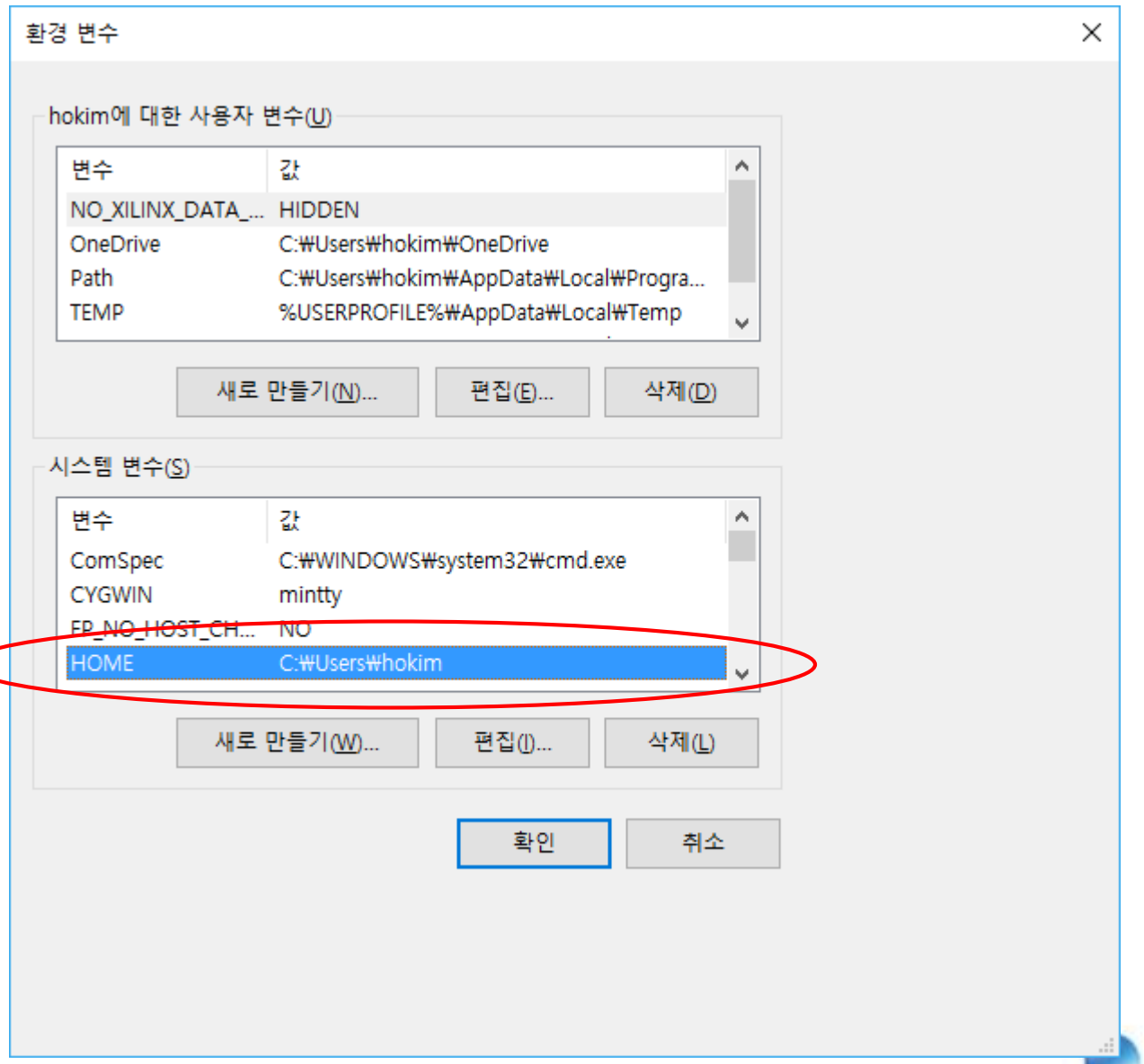




Development Environment Set-up

Environment Variable(HOME)

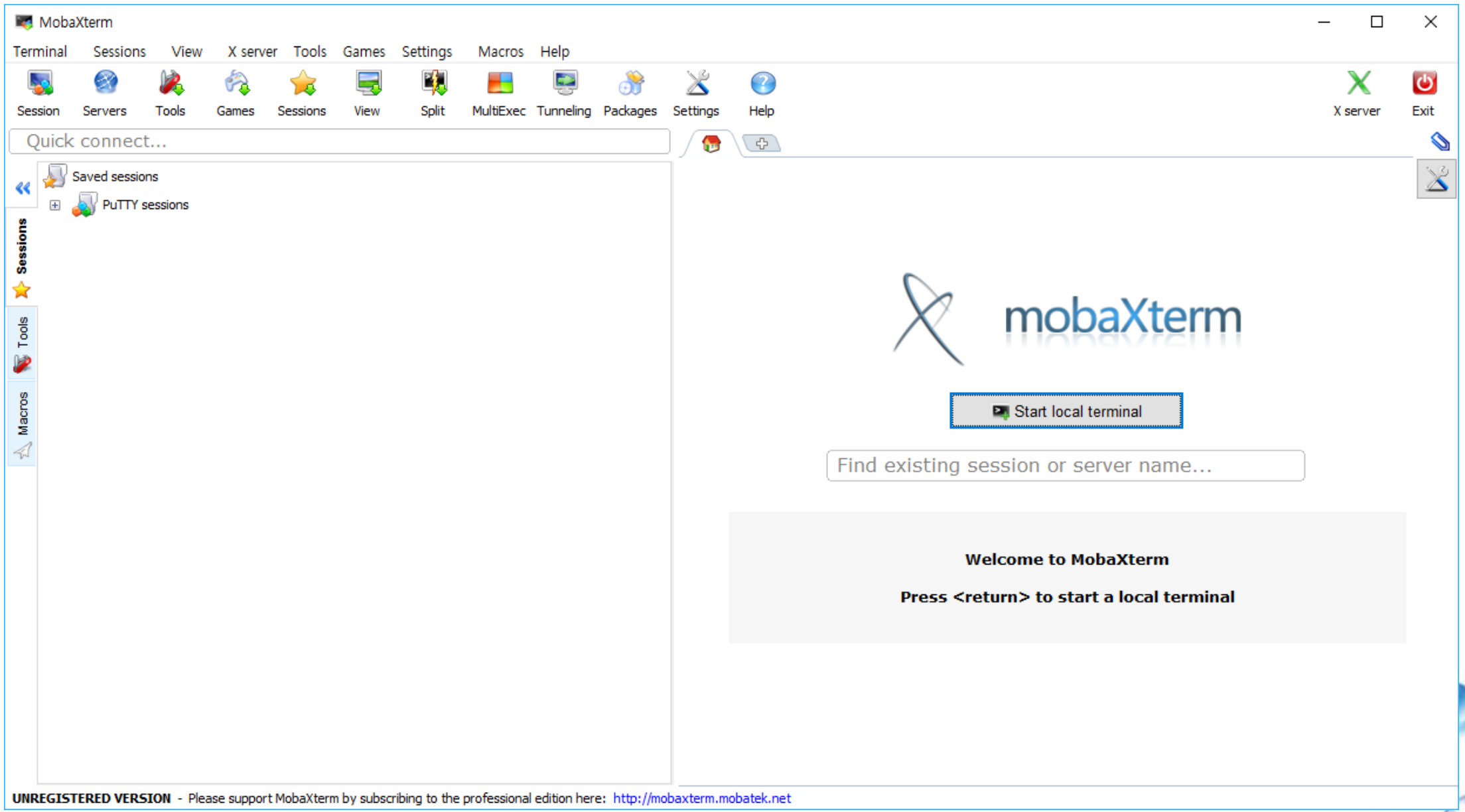




Development Environment Set-up

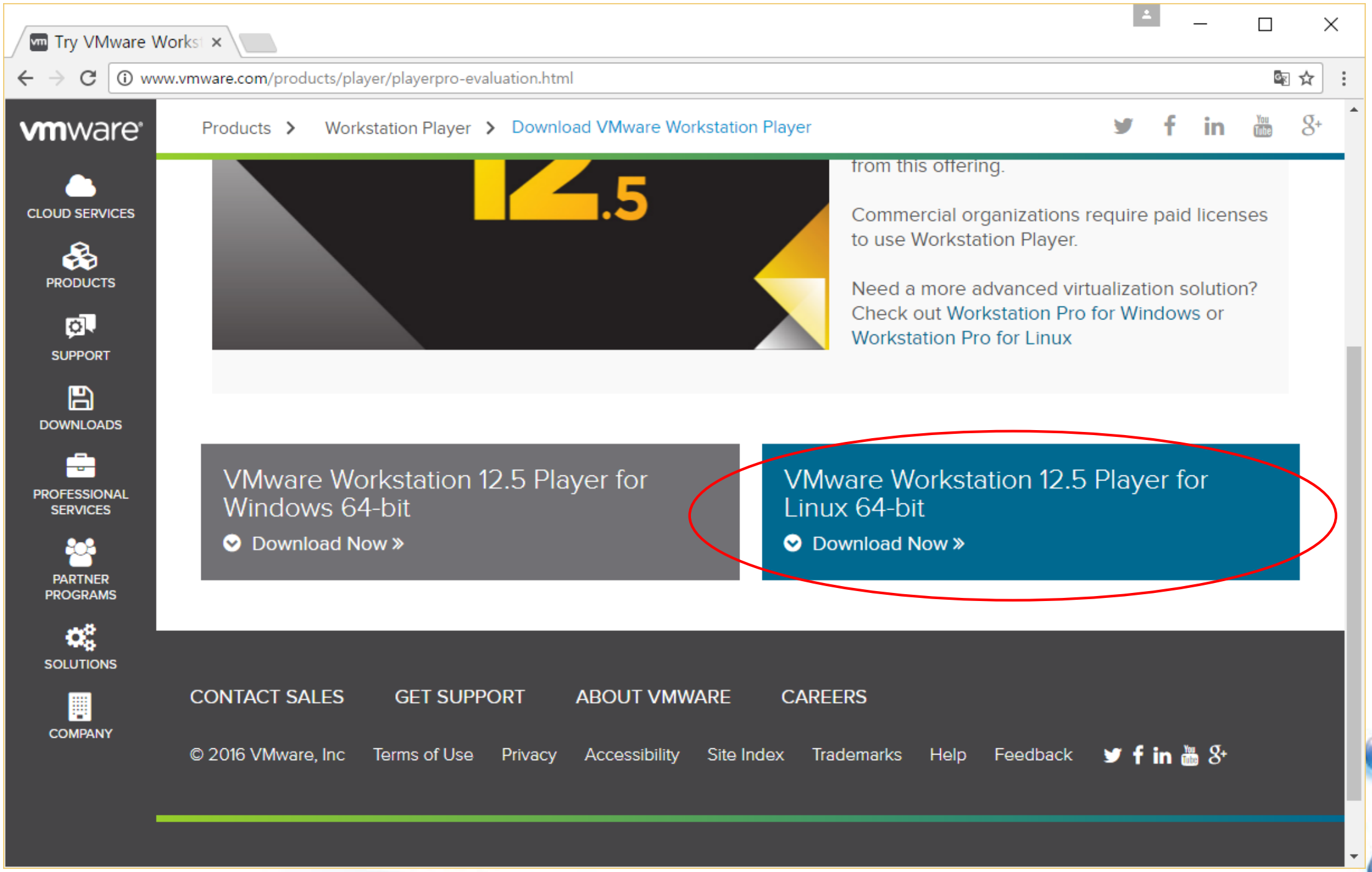
- **Windows ←→ Ubuntu**

  http://mobaxterm.mobatek.net/



Development Environment Set-up

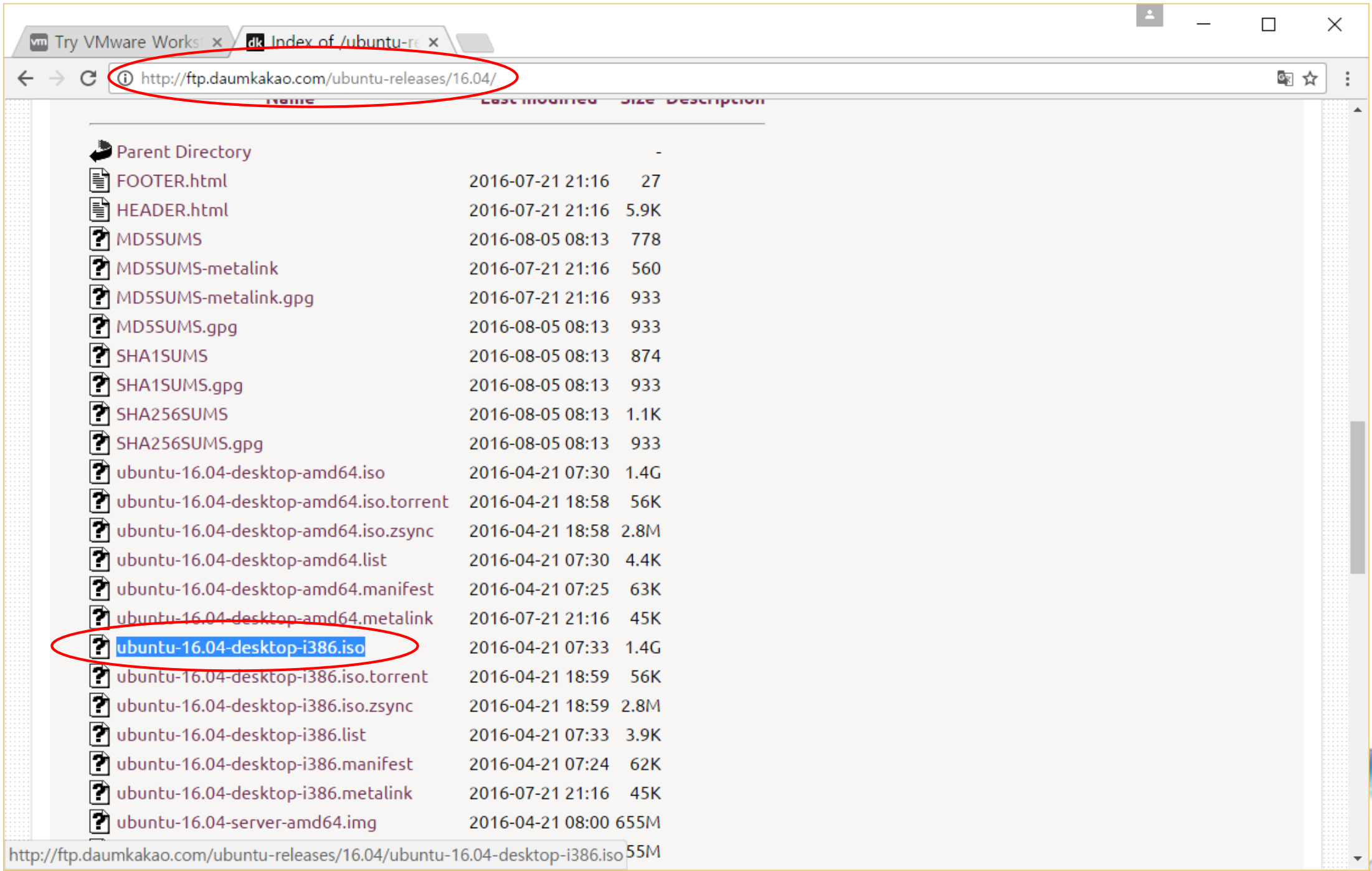
- **Ubuntu on VMware-player**



Development Environment Set-up

INIPRO

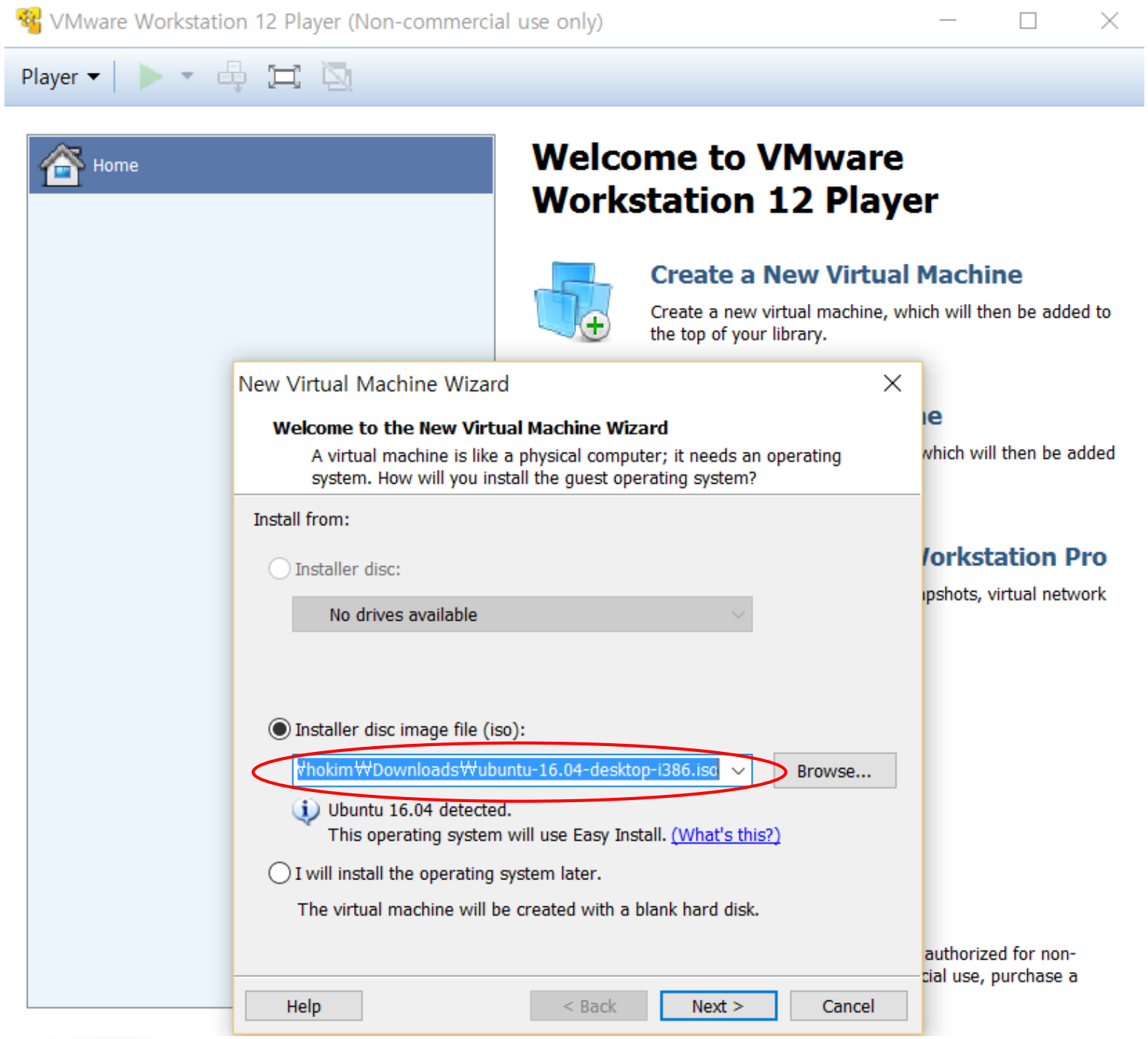
- **Ubuntu on VMware-player**



Development Environment Set-up

INIPRO

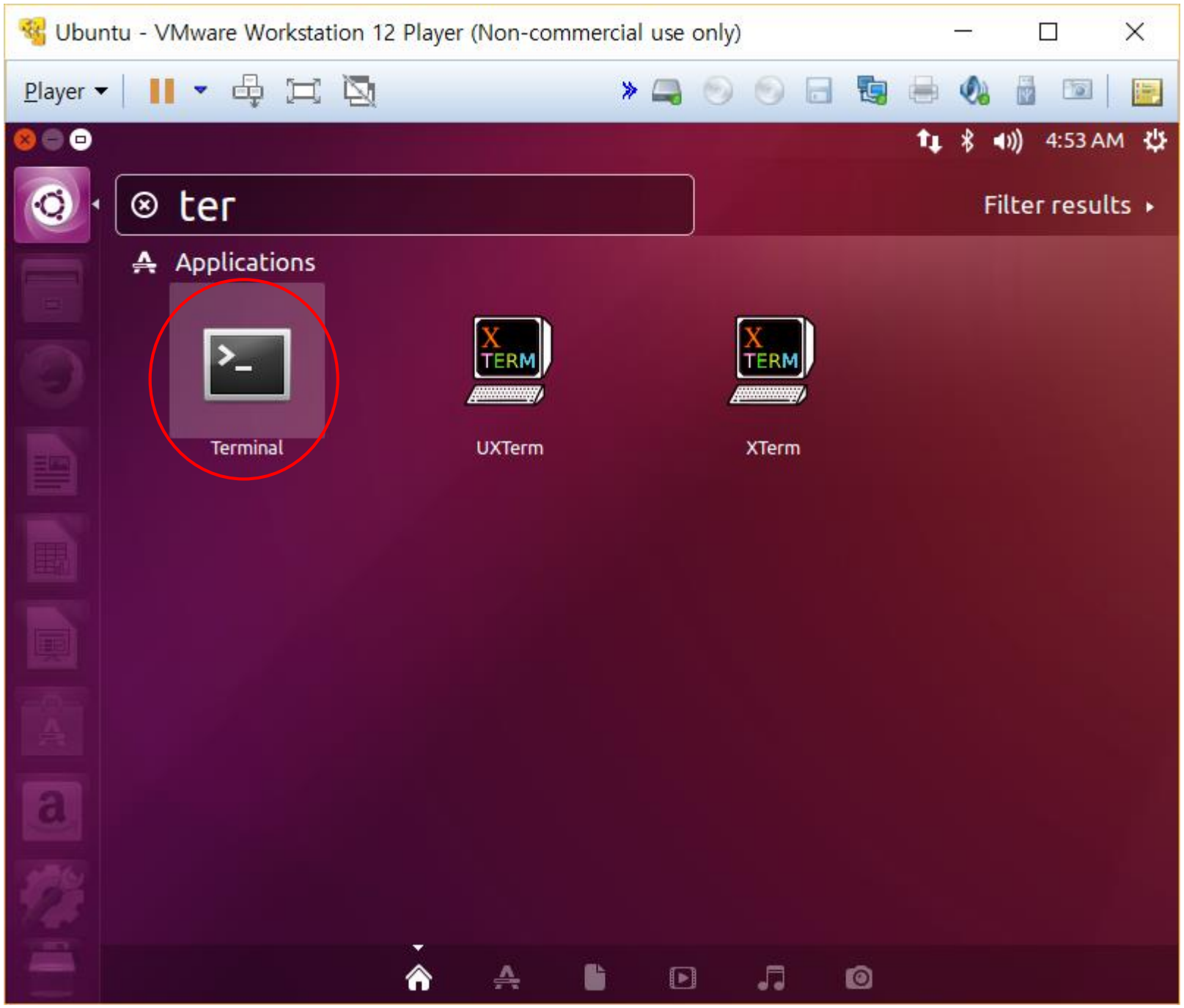
- **Ubuntu on VMware-player**



Development Environment Set-up

INIPRO

- **Ubuntu on VMware-player**

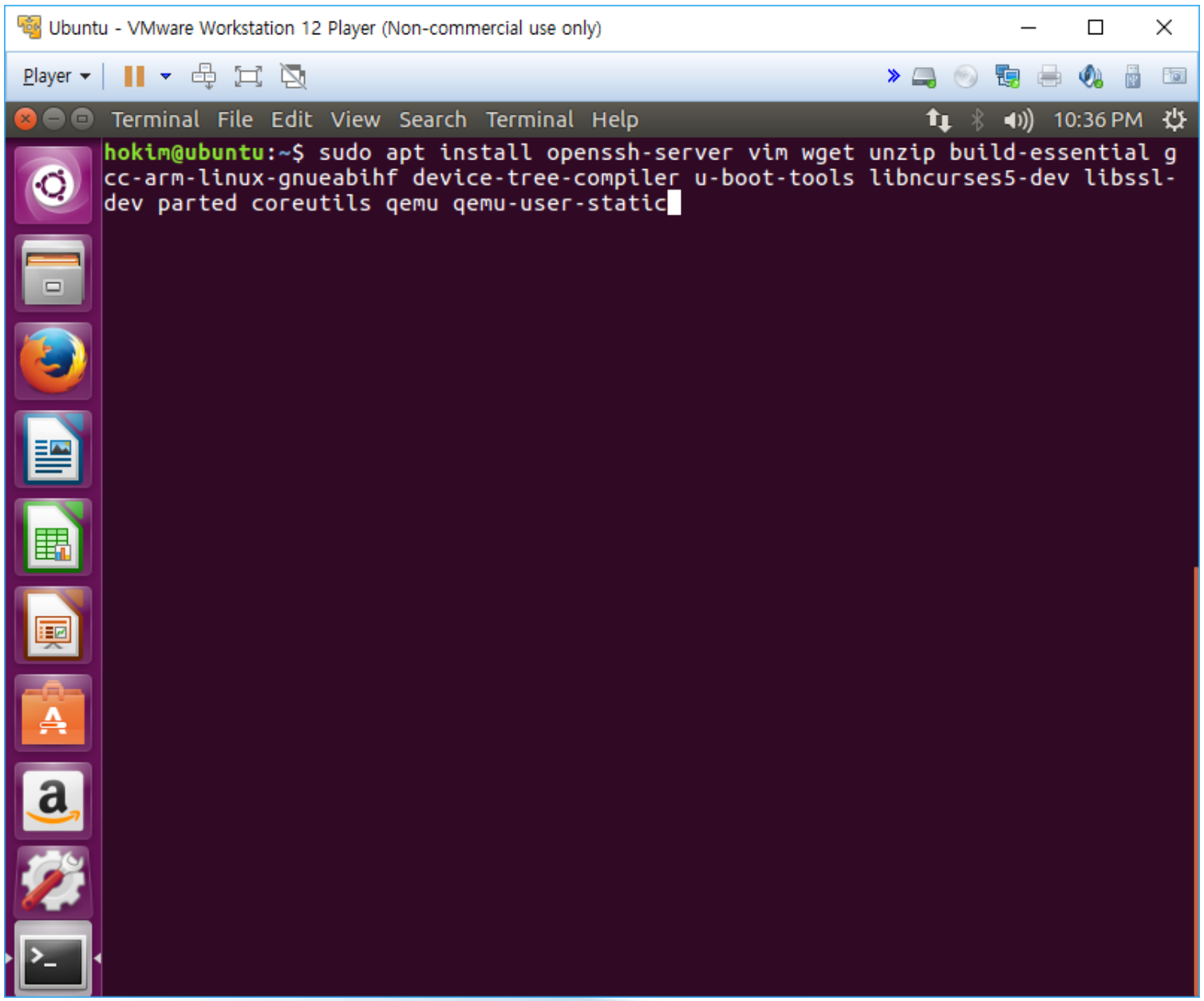


Development Environment Set-up

INIPRO

Ubuntu on VMware-player

$ sudo apt 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

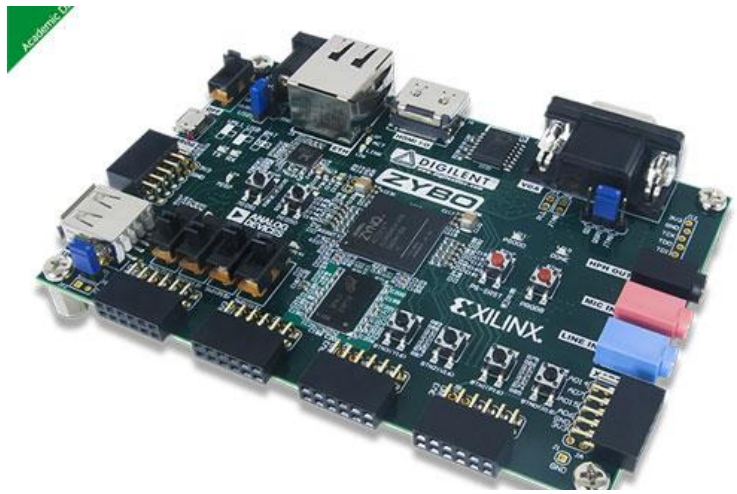


Development Process

- **Target Board** Zybo(Zynq-7010)

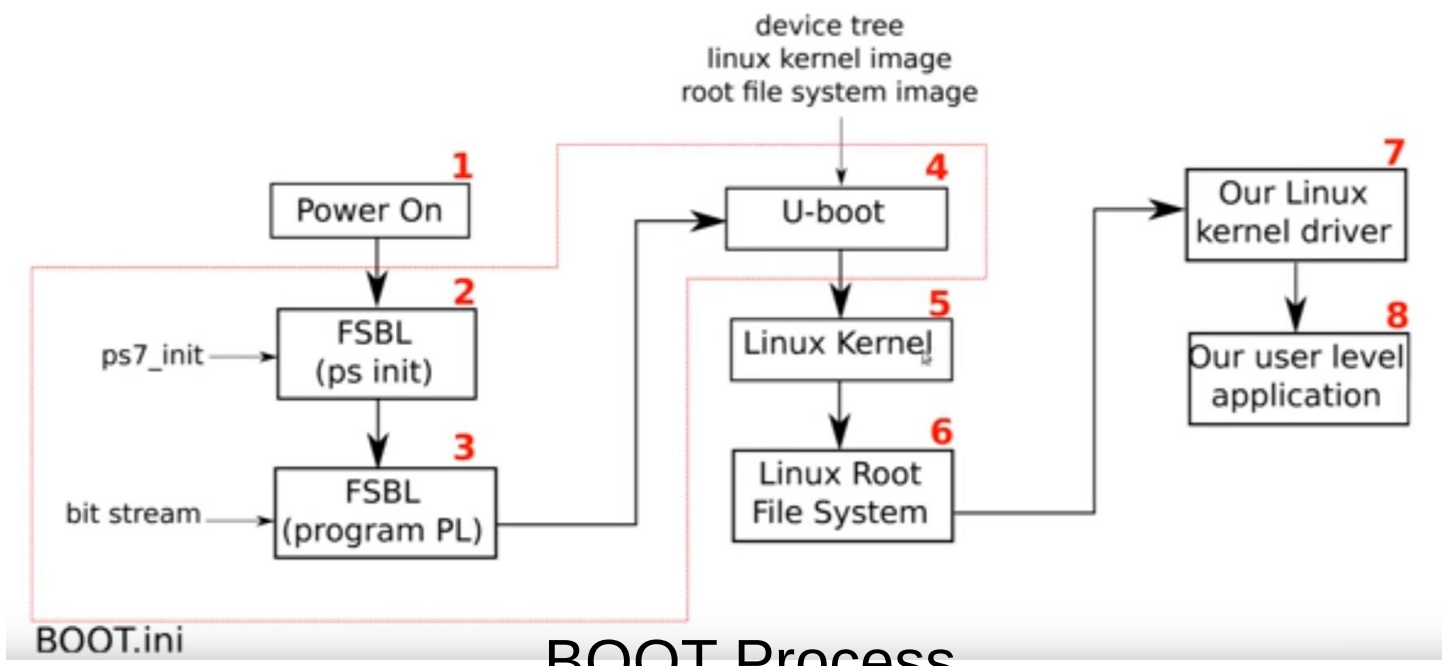
  Gigabit Ethernet, USB, SD, UART

  28,000 logic cells, 240 KB BRAM



- **Development Process**

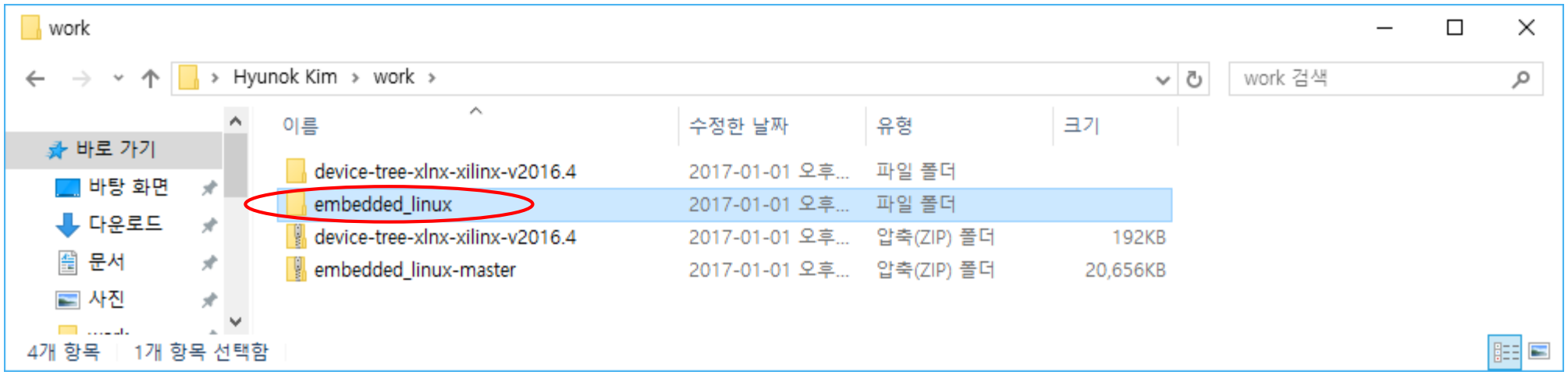
1. bit stream
2. FSBL
3. DT Generation
4. u-boot
5. DT Compile
6. linux kernel image
7. root file system image
8. BOOT.bin(fsbl + bit + u-boot)
9. Update BOOT.bin / DT / kernel
10. (user kernel driver)/user application

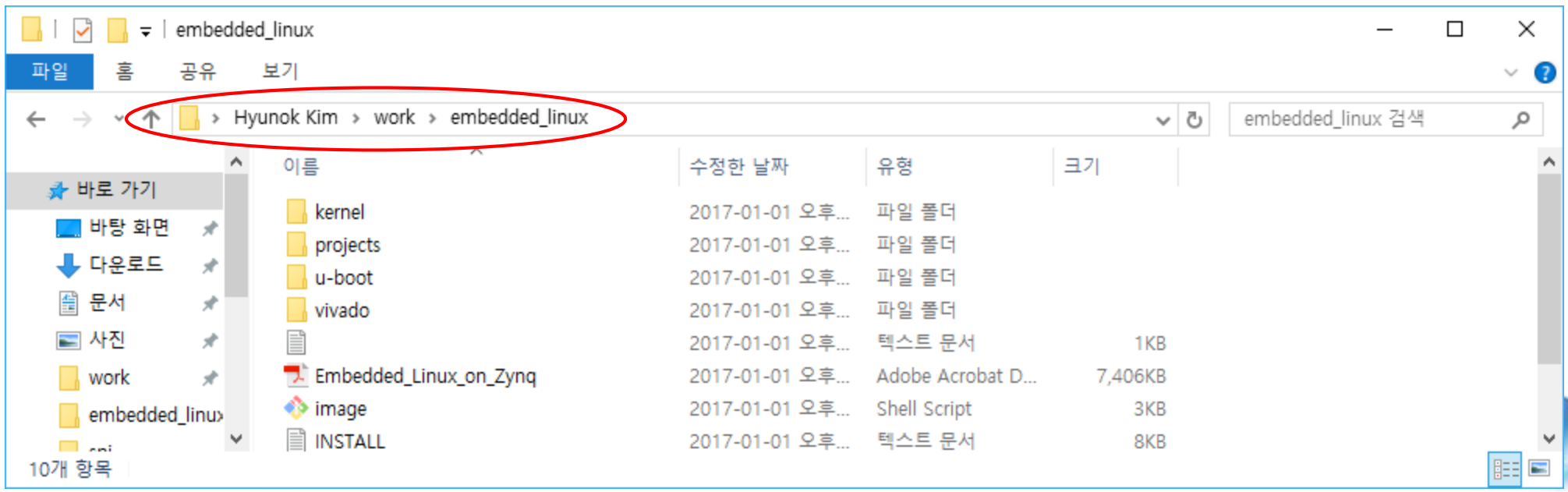


BOOT Process

Development Process

- **Source code**
  https://github.com/inipro/embedded_linux





Design of Zynq PS Hardware

INIPRO

- **Vivado Design**

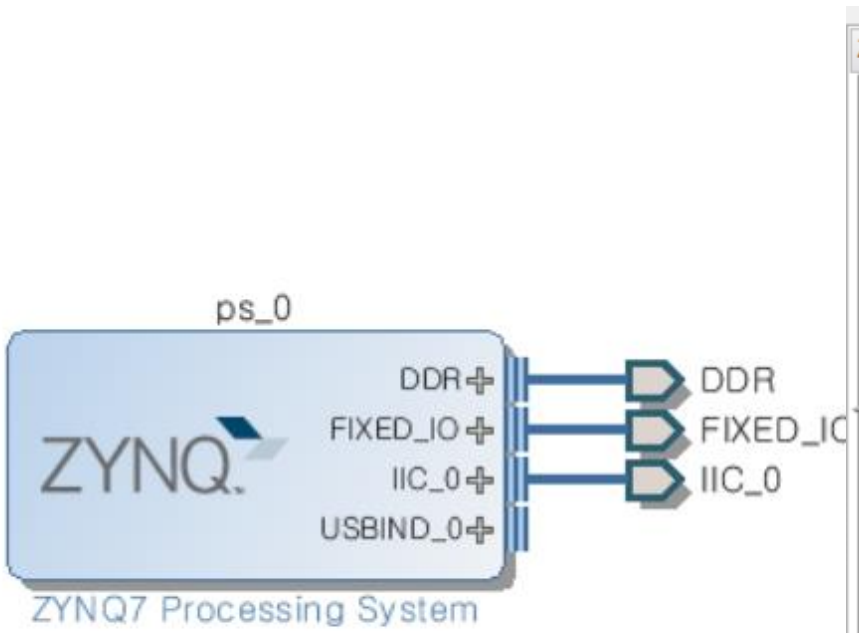
SD0 (boot files, root file system)

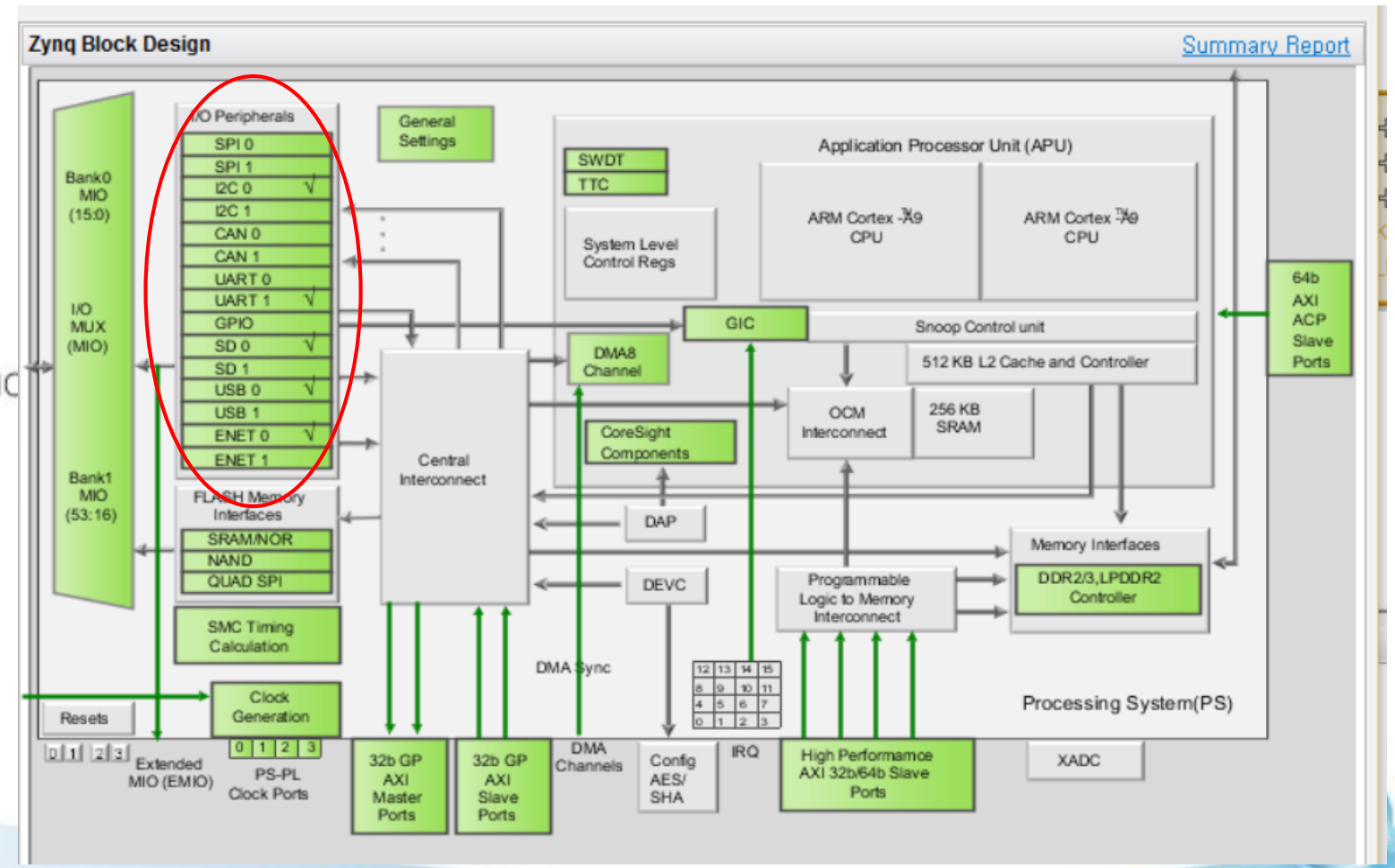
UART1 (console)

I2C0 (eeprom w/ Ethernet mac address; EMIO)

USB0 (usb host)

ENET0 (Gigabit Ethernet)





Vivado Project

INIPRO

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
```

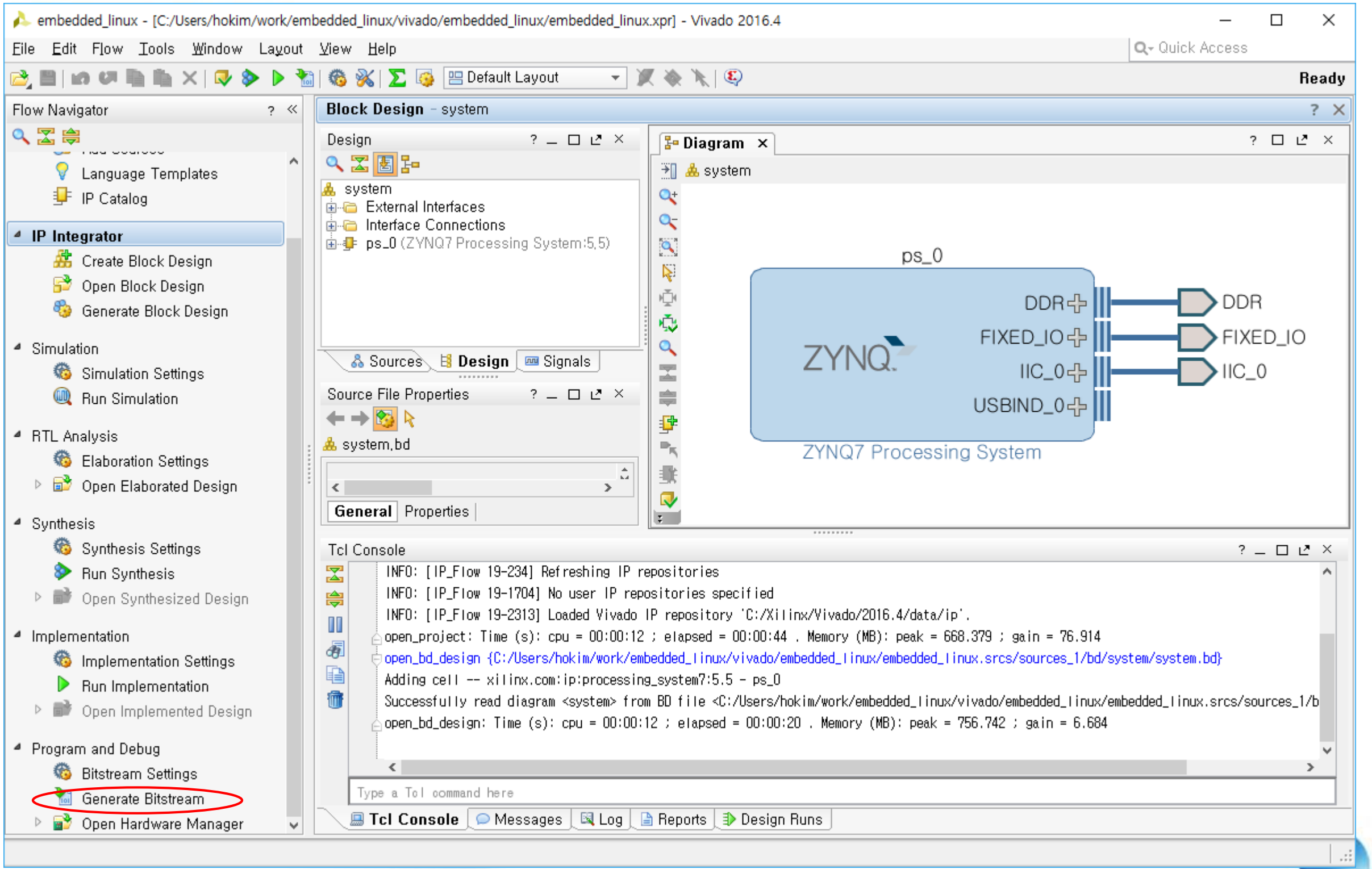
In cmd window for Vivado

```
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

INIPRO



HDF Generation

INIPRO

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

FSB Compile

INIPRO

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

DT Generation

INIPRO

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-v2016.4

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

set_property CONFIG.kernel_version {2016.4} [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]
```

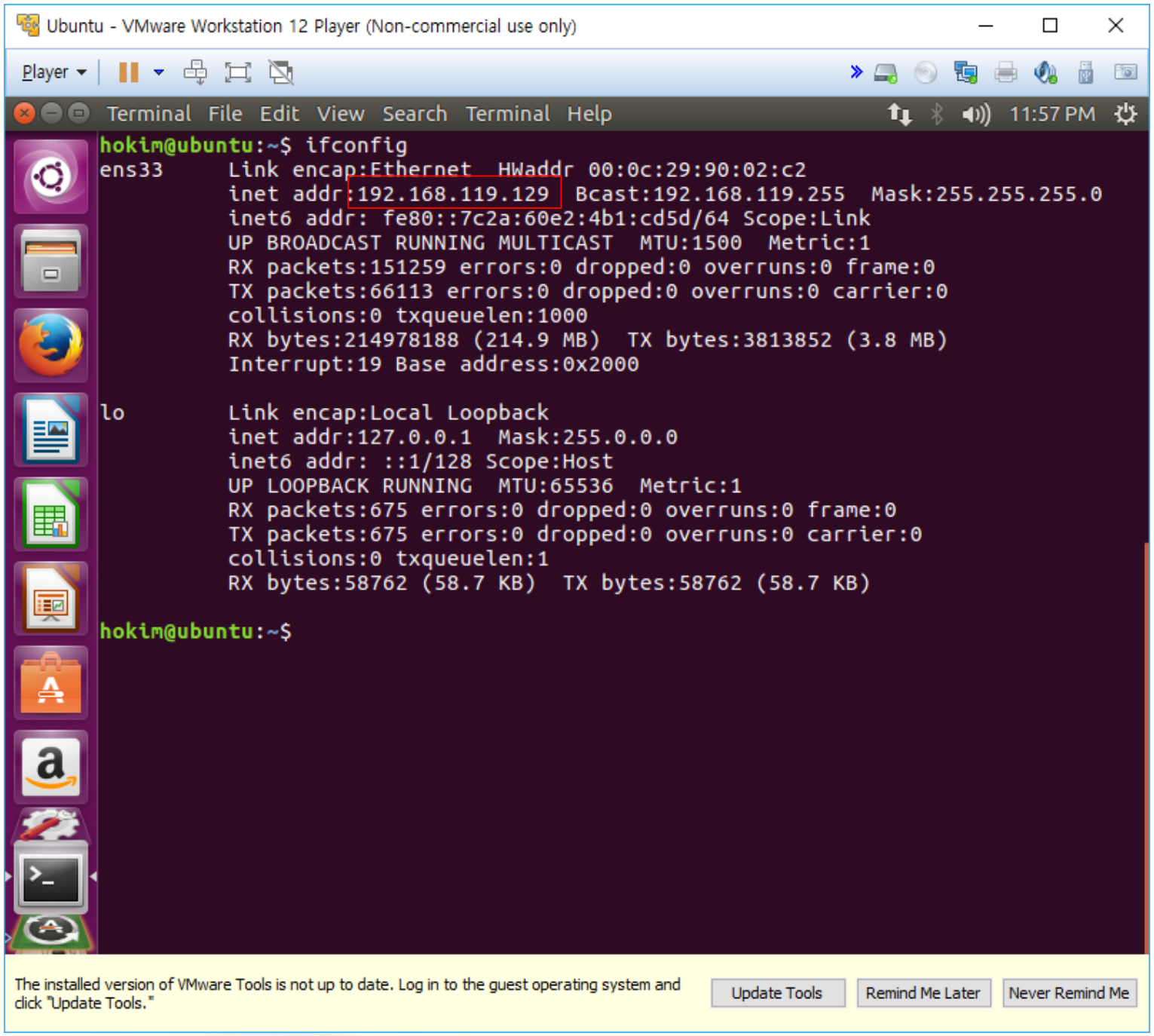
```
C:\> hsi  -nolog -nojournal -mode batch -source devicetree.tcl
```

output : embedded_linux\embedded_linux.tree/*

SFTP File Transfer (Windows → Ubuntu)

INIPRO

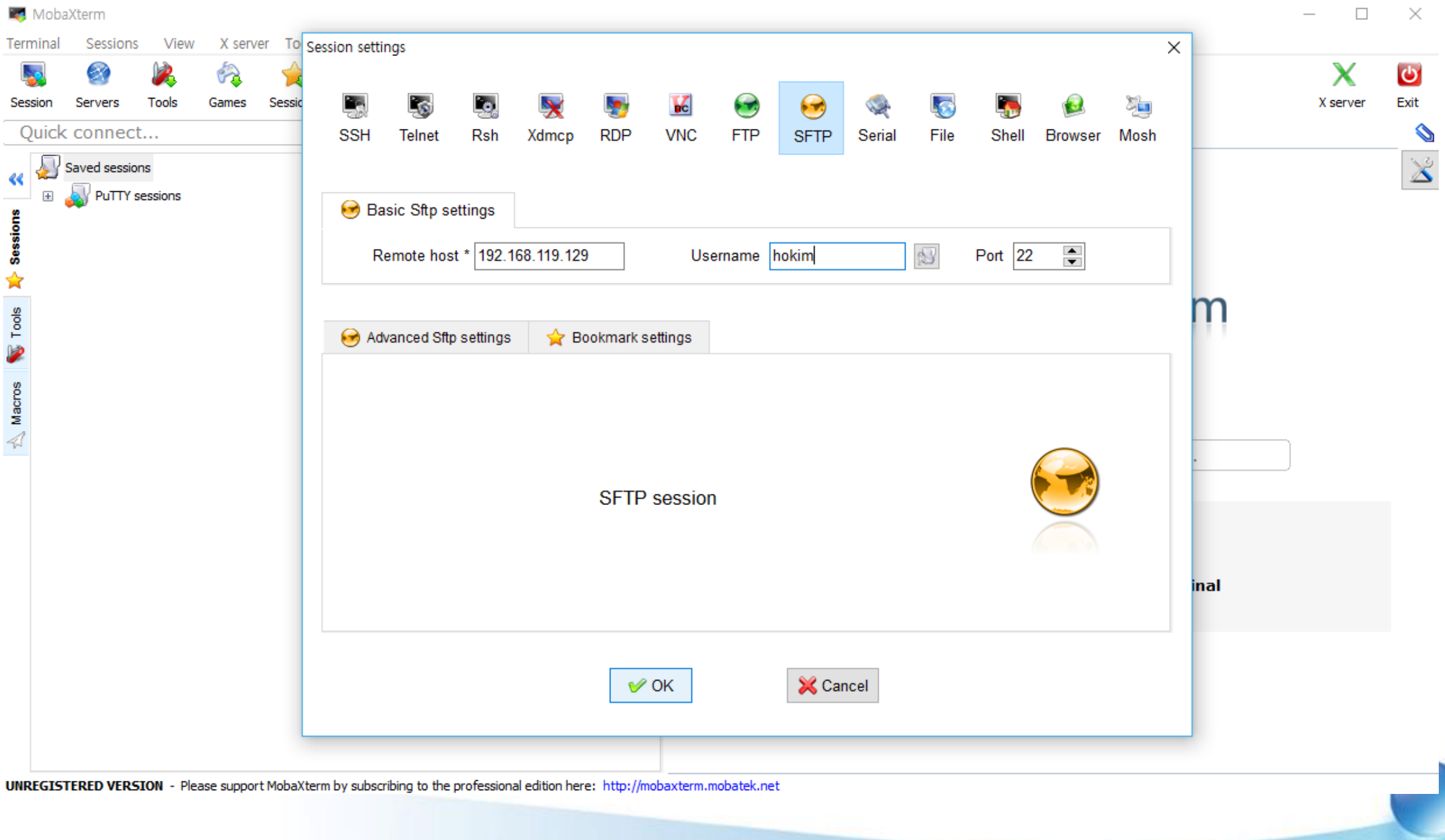
Ubuntu network IP



SFTP File Transfer (Windows → Ubuntu)

INIPRO

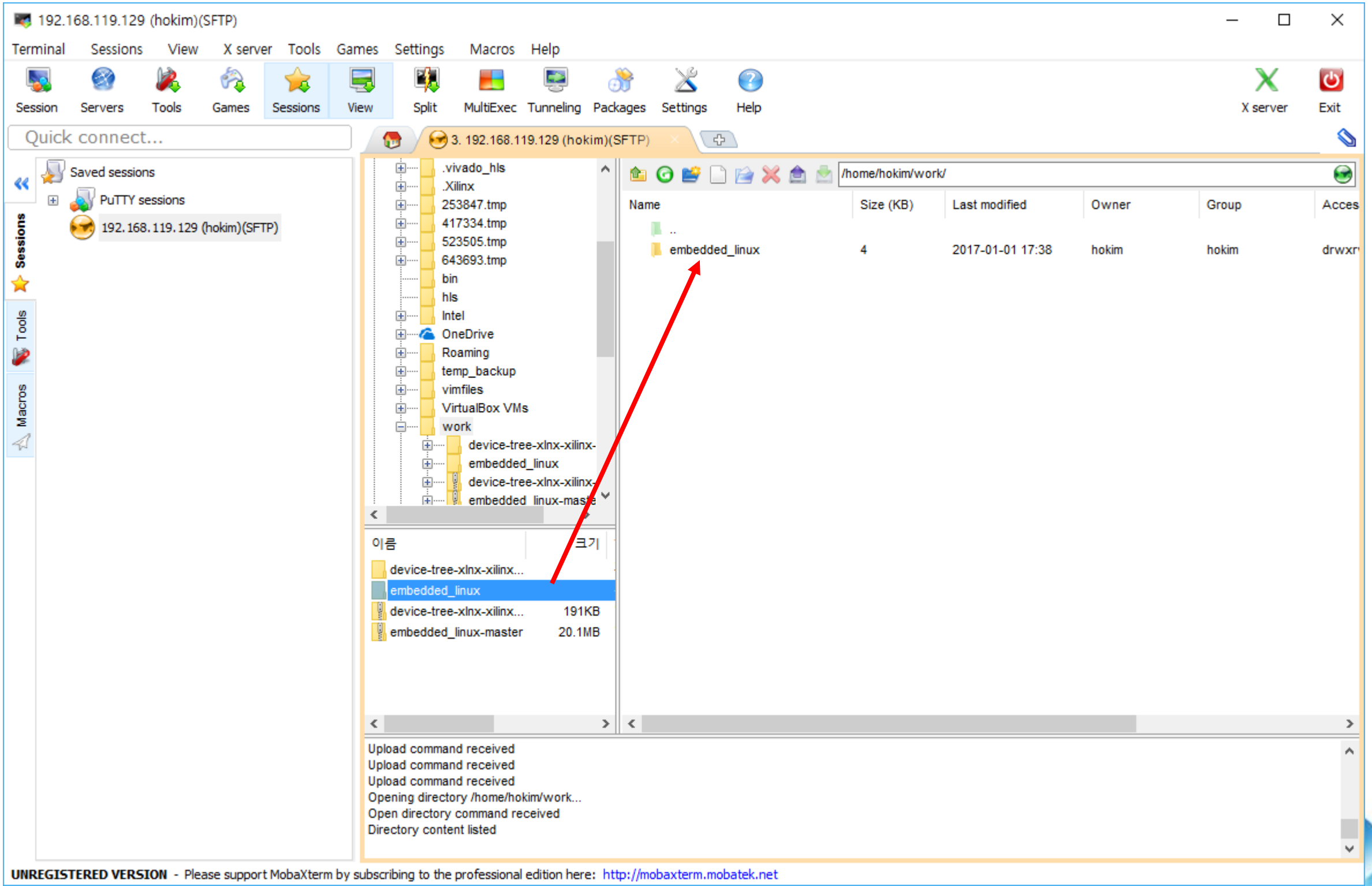
embedded_linux using MobaXterm



SFTP File Transfer (Windows → Ubuntu)

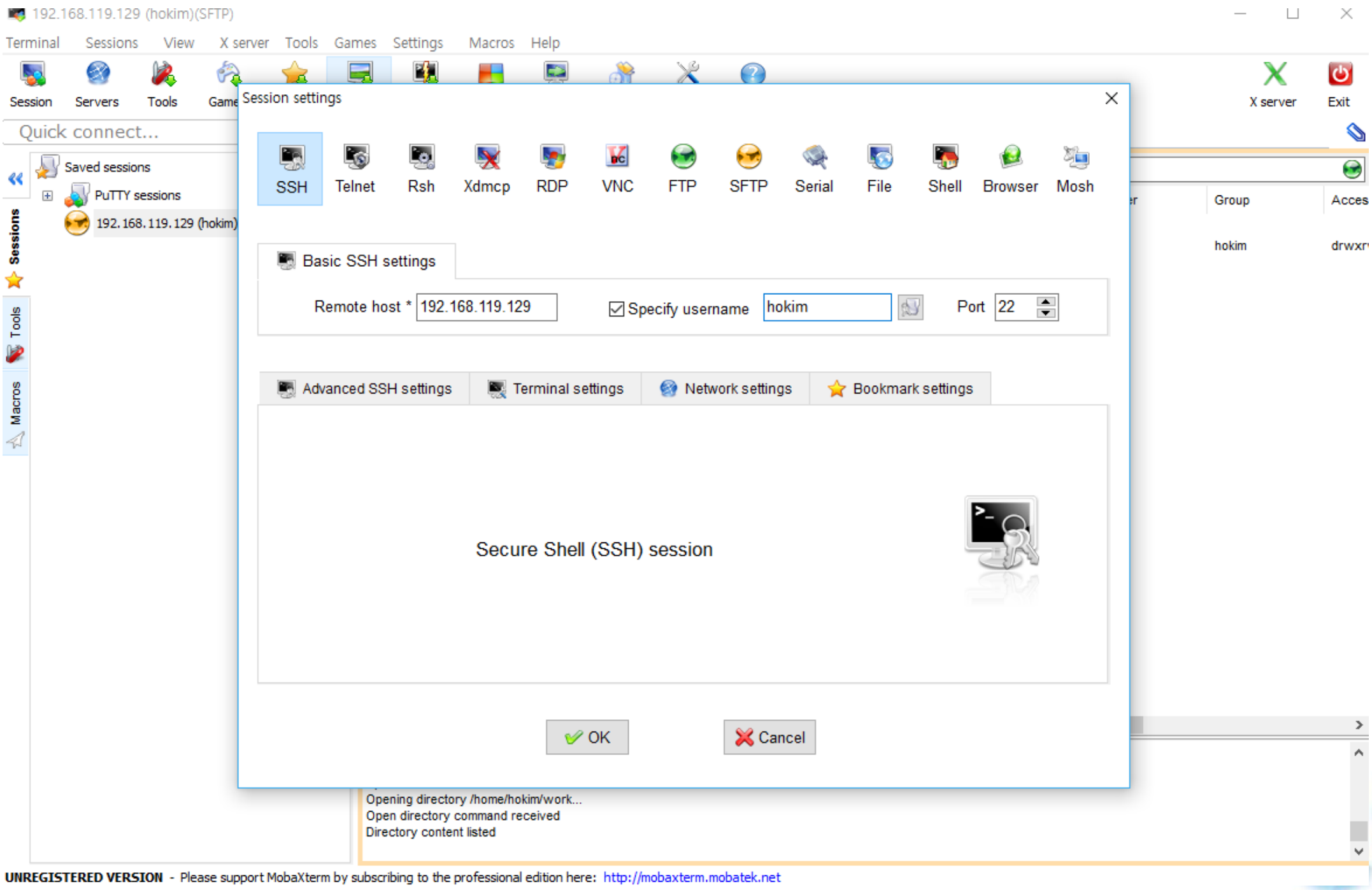
INIPRO

embedded_linux using MobaXterm



SSH Login (Windows → Ubuntu)

INIPRO



SSH Login (Windows → Ubuntu)

INIPRO

192.168.119.129 (hokim)(SSH)

Terminal Sessions View X server Tools Games Settings Macros Help

Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit

Quick connect...

3. 192.168.119.129 (hokim)(SFTP) | 5. 192.168.119.129 (hokim)(SSH)

/home/hokim/

Name	Size (KB)
..	
.cache	4
.config	4
.dbus	4
.gconf	4
.gnupg	4
.local	4
.nano	4
Desktop	4
Documents	4
Downloads	4
Music	4
Pictures	4
Public	4
Templates	4
Videos	4
vmware-tools-distrib	4
work	4
.bash_history	5
.bash_logout	0
.bashrc	3
.dmrc	0
.ICEauthority	4
.profile	0
.sudo_as_admin_successful	0
.viminfo	11
.vimrc	0
.wget-hsts	0

Follow terminal folder

```
                    ? MobaXterm 9.3 ?
            (SSH client, X-server and networking tools)

➤ SSH session to hokim@192.168.119.129
  ? SSH compression : ✔
  ? SSH-browser     : ✔
  ? X11-forwarding  : ✔  (remote display is forwarded through SSH)
  ? DISPLAY         : ✔  (automatically set on remote server)

➤ For more info, ctrl+click on help or visit our website

Welcome to Ubuntu 16.04 LTS (GNU/Linux 4.4.0-47-generic i686)

 * Documentation:  https://help.ubuntu.com/

369 packages can be updated.
66 updates are security updates.

Last login: Sun Jan  1 00:31:20 2017 from 192.168.119.1
hokim@ubuntu:~$
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

- **Boot loader** for linux

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

  Set ethernet using mac address from i2c eeprom.

  Boot uImage.

u-boot Compile

Log into VMWare Ubuntu using Mobaterm
Use nano or leafpad for editing
- : deletion
+ : insertion

```
$ cd ~/work/embedded_linux
$ mkdir -p dl
$ wget https://github.com/Xilinx/u-boot-xlnx/archive/xilinx-v2016.4.tar.gz -O dl/u-boot-xlnx-xilinx-v2016.4.tar.gz
$ cd u-boot
$ tar xvzf ../dl/u-boot-xlnx-xilinx-v2016.4.tar.gz
$ cd u-boot-xlnx-xilinx-v2016.4
$ cp ../zynq-zyboc.dts arch/arm/dts
$ cp ../zynq_zyboc.h include/configs
$ cp ../zynq_zyboc_defconfig configs
```

u-boot Compile

zynq-zyboc.dts

```
.....
/dts-v1/;
#include "zynq-7000.dtsi"

/ {
        model = "Zynq ZYBOC Development Board";
        compatible = "inipro,zynq-zyboc", "xlnx,zynq-7000";

        aliases {
                ethernet0 = &gem0;
                serial0 = &uart1;
                spi0 = &qspi;
                mmc0 = &sdhci0;
        };

        memory {
                device_type = "memory";
                reg = <0x0 0x20000000>;
        };

        chosen {
                bootargs = "";
                stdout-path = "serial0:115200n8";
        };

        usb_phy0: phy0 {
                compatible = "usb-nop-xceiv";
                #phy-cells = <0>;
                reset-gpios = <&gpio0 46 1>;
        };
};
```

```
&clkc {
        ps-clk-frequency = <50000000>;
};

&gem0 {
        status = "okay";
        phy-mode = "rgmii-id";
        phy-handle = <&ethernet_phy>;

        ethernet_phy: ethernet-phy@0 {
                reg = <0>;
        };
};

&qspi {
        u-boot,dm-pre-reloc;
        status = "okay";
};

&sdhci0 {
        u-boot,dm-pre-reloc;
        status = "okay";
};

&uart1 {
        u-boot,dm-pre-reloc;
        status = "okay";
};

&usb0 {
        status = "okay";
        dr_mode = "host";
        usb-phy = <&usb_phy0>;
};
```

u-boot Compile

zynq_zyboc.h

```
.....
#ifndef __CONFIG_ZYNQ_ZYBOC_H
#define __CONFIG_ZYNQ_ZYBOC_H

#define CONFIG_ZYNQ_I2C0
#define CONFIG_ZYNQ_I2C1
#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_DISPLAY
#define CONFIG_I2C_EDID

/* Define ZYBO PS Clock Frequency to 50MHz */
#define CONFIG_ZYNQ_PS_CLK_FREQ       50000000UL
#define CONFIG_EXTRA_ENV_SETTINGS   \
    "loadbootenv_addr=0x2000000\0" \
    "bootenv=uEnv.txt\0" \
    "loadbootenv=load mmc 0 ${loadbootenv_addr} ${bootenv}\0" \
    "importbootenv=echo Importing environment from SD ...; " \
        "env import -t ${loadbootenv_addr} $filesize\0" \
    "sd_uEnvtxt_existence_test=test -e mmc 0 /uEnv.txt\0" \
    "preboot=if test $modeboot = sdboot && env run
sd_uEnvtxt_existence_test; " \
            "then if env run loadbootenv; " \
                "then env run importbootenv; " \
            "fi; " \
        "fi; \0"

#include <configs/zynq-common.h>


#endif /* __CONFIG_ZYNQ_ZYBOC_H */
```

u-boot Compile

zynq_zyboc_defconfig

```
CONFIG_ARM=y
CONFIG_SYS_CONFIG_NAME="zynq_zyboc"
CONFIG_ARCH_ZYNQ=y
CONFIG_DEFAULT_DEVICE_TREE="zynq-zyboc"
CONFIG_SPL=y
CONFIG_FIT=y
CONFIG_FIT_VERBOSE=y
CONFIG_FIT_SIGNATURE=y
CONFIG_SYS_NO_FLASH=y
CONFIG_HUSH_PARSER=y
CONFIG_SYS_PROMPT="Zynq> "
# CONFIG_CMD_IMLS is not set
# CONFIG_CMD_FLASH is not set
CONFIG_CMD_MMC=y
CONFIG_CMD_SF=y
CONFIG_CMD_I2C=y
CONFIG_CMD_USB=y
CONFIG_CMD_DFU=y
CONFIG_CMD_GPIO=y
# CONFIG_CMD_SETEXPR is not set
CONFIG_CMD_TFTPPUT=y
CONFIG_CMD_DHCP=y
CONFIG_CMD_MII=y
CONFIG_CMD_PING=y
CONFIG_CMD_CACHE=y
CONFIG_CMD_EXT2=y
CONFIG_CMD_EXT4=y
CONFIG_CMD_EXT4_WRITE=y
CONFIG_CMD_FAT=y
CONFIG_CMD_FS_GENERIC=y
CONFIG_OF_EMBED=y
CONFIG_NET_RANDOM_ETHADDR=y
CONFIG_SPL_DM_SEQ_ALIAS=y
```

```
CONFIG_ZYNQ_SDHCI=y
CONFIG_SPI_FLASH=y
CONFIG_SPI_FLASH_BAR=y
CONFIG_SPI_FLASH_SPANSION=y
CONFIG_ZYNQ_GEM=y
CONFIG_DEBUG_UART=y
CONFIG_DEBUG_UART_ZYNQ=y
CONFIG_DEBUG_UART_BASE=0xe0001000
CONFIG_DEBUG_UART_CLOCK=50000000
CONFIG_ZYNQ_QSPI=y
CONFIG_USB=y
CONFIG_USB_EHCI_HCD=y
CONFIG_USB_ULPI_VIEWPORT=y
CONFIG_USB_ULPI=y
CONFIG_USB_STORAGE=y
CONFIG_USB_GADGET=y
CONFIG_CI_UDC=y
CONFIG_USB_GADGET_DOWNLOAD=y
CONFIG_G_DNL_MANUFACTURER="Xilinx"
CONFIG_G_DNL_VENDOR_NUM=0x03fd
CONFIG_G_DNL_PRODUCT_NUM=0x0300
```

u-boot Compile

INIPRO

$ 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-picozed.dtb \
    zynq-zc770-xm010.dtb \
    zynq-zc770-xm011.dtb \
    zynq-zc770-xm012.dtb \
-   zynq-zc770-xm013.dtb
+   zynq-zc770-xm013.dtb \
+   zynq-zyboc.dtb
```

```
$ 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/vivado
$ cp embedded_linux/embedded_linux.tree/system.dts .
$ nano system.dts
```

system.dts

```
&clkc {
    fclk-enable = <0x0>;
    ps-clk-frequency = <50000000>;
};

+&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 embedded_linux/embedded_linux.tree/ -o devicetree.dtb
system.dts
```

Linux Kernel Compile

INIPRO

```
$ cd ~/work/embedded_linux
$ wget https://github.com/Xilinx/linux-xlnx/archive/xilinx-v2016.4.tar.gz -O
dl/linux-xlnx-xilinx-v2016.4.tar.gz
$ cd kernel
$ tar xvzf ../dl/linux-xlnx-xilinx-v2016.4.tar.gz
$ cd linux-xlnx-xilinx-v2016.4
$ make ARCH=arm xilinx_zynq_defconfig
$ make ARCH=arm menuconfig
```

Linux Kernel Compile

INIPRO

hokim@ubuntu: ~/work/embedded_linux/kernel/linux-xlnx-xilinx-v2016.4

File Edit Tabs Help

```
.config - Linux/arm 4.6.0 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 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)
    [*]   Minstrel (NEW)
          v(+)

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

hokim@ubuntu: ~/work/embedded_linux/kernel/linux-xlnx-xilinx-v2016.4

File Edit Tabs Help

```
.config - Linux/arm 4.6.0 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 and Open Firmware support
    [ ]   Device Tree runtime unit tests
    [*]   Device Tree overlays

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

Linux Kernel Compile

INIPRO

hokim@ubuntu: ~/work/embedded_linux/kernel/linux-xlnx-xilinx-v2016.4

File Edit Tabs Help

```
.config - Linux/arm 4.6.0 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  [ ]
  ┌──────────────────────────────────────────────────────────────────┐
  │                    ^(-)                                          │
  │     < >     Marvell 88W8xxx PCI/PCIe Wireless support (NEW)     │
  │     [*]   MediaTek devices                                       │
  │     < >     MediaTek MT7601U (USB) support (NEW)                 │
  │     [*]   Ralink devices                                         │
  │     < >     Ralink driver support (NEW)  ----                    │
  │     [*]   Realtek devices                                        │
  │     < >     Realtek 8180/8185/8187SE PCI support (NEW)           │
  │     < >     Realtek 8187 and 8187B USB support (NEW)             │
  │     <*>     Realtek rtlwifi family of devices  ----               │
  │     < >     RTL8723AU/RTL8188[CR]U/RTL819[12]CU (mac80211) support (N
  │                    v(+)                                          │
  └──────────────────────────────────────────────────────────────────┘
        <Select>    < Exit >    < Help >    < Save >    < Load >
```

hokim@ubuntu: ~/work/embedded_linux/kernel/linux-xlnx-xilinx-v2016.4

File Edit Tabs Help

```
.config - Linux/arm 4.6.0 Kernel Configuration
> Device Drivers
──────────────────────────── Device 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  [ ]
  ┌──────────────────────────────────────────────────────────────────┐
  │                    ^(-)                                          │
  │     [*] DMA Engine support  --->                                 │
  │     [ ] Auxiliary Display support  ----                          │
  │     <*> Userspace I/O drivers  --->                              │
  │     [ ] Virtualization drivers  ----                             │
  │         Virtio drivers  --->                                     │
  │         Microsoft Hyper-V guest support  ----                    │
  │     [*] Staging drivers  --->                                    │
  │     [ ] Platform support for Goldfish virtual devices  ----      │
  │     [ ] Platform support for Chrome hardware  ----               │
  │         Common Clock Framework  --->                             │
  │                    v(+)                                          │
  └──────────────────────────────────────────────────────────────────┘
        <Select>    < Exit >    < Help >    < Save >    < Load >
```

Linux Kernel Compile

INIPRO

hokim@ubuntu: ~/work/embedded_linux/kernel/linux-xlnx-xilinx-v2016.4

File Edit Tabs Help

```
.config - Linux/arm 4.6.0 Kernel Configuration
> Device Drivers > Staging drivers
                        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  [ ]

      --- Staging drivers
      < >   Prism2.5/3 USB driver (NEW)
      < >   Data acquisition support (comedi) (NEW)
      < >   RealTek RTL8192U Wireless LAN NIC driver (NEW)
      < >   Support for rtllib wireless devices (NEW)
      < >   RealTek RTL8712U (RTL8192SU) Wireless LAN NIC driver (NEW)
      <*>   Realtek RTL8188EU Wireless LAN NIC driver
      [*]     Realtek RTL8188EU AP mode (NEW)
      < >   Realtek PCI-E Card Reader RTS5208/5288 support (NEW)
      < >   VIA Technologies VT6655 support (NEW)
      v(+)

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

hokim@ubuntu: ~/work/embedded_linux/kernel/linux-xlnx-xilinx-v2016.4

File Edit Tabs Help

```
.config - Linux/arm 4.6.0 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  [ ]
      ^(-)
      < >   Rockchip SPI controller driver
      < >   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
            *** SPI Protocol Masters ***
      <*>   User mode SPI device driver support
      < >   spi loopback test framework support
      v(+)

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

Linux Kernel Compile

INIPRO

hokim@ubuntu: ~/work/embedded_linux/kernel/linux-xlnx-xilinx-v2016.4

File Edit Tabs Help

```
.config - Linux/arm 4.6.0 Kernel Configuration
> Device Drivers > GPIO Support > Memory mapped GPIO drivers
                    Memory mapped GPIO 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  [ ]
      -*- Generic memory-mapped GPIO controller support (MMIO platform
      < > Aeroflex Gaisler GRGPIO support
      [ ] MPC512x/MPC8xxx/QorIQ GPIO support
      [ ] PrimeCell PL061 GPIO support
      < > GPIO based on SYSCON
      < > VIA VX855/VX875 GPIO
      <*> Xilinx GPIO support
      [ ] LSI ZEVIO SoC memory mapped GPIOs
      <*> Xilinx Zynq GPIO support
      [ ] ZTE ZX GPIO support
          <Select>    < Exit >    < Help >    < Save >    < Load >
```

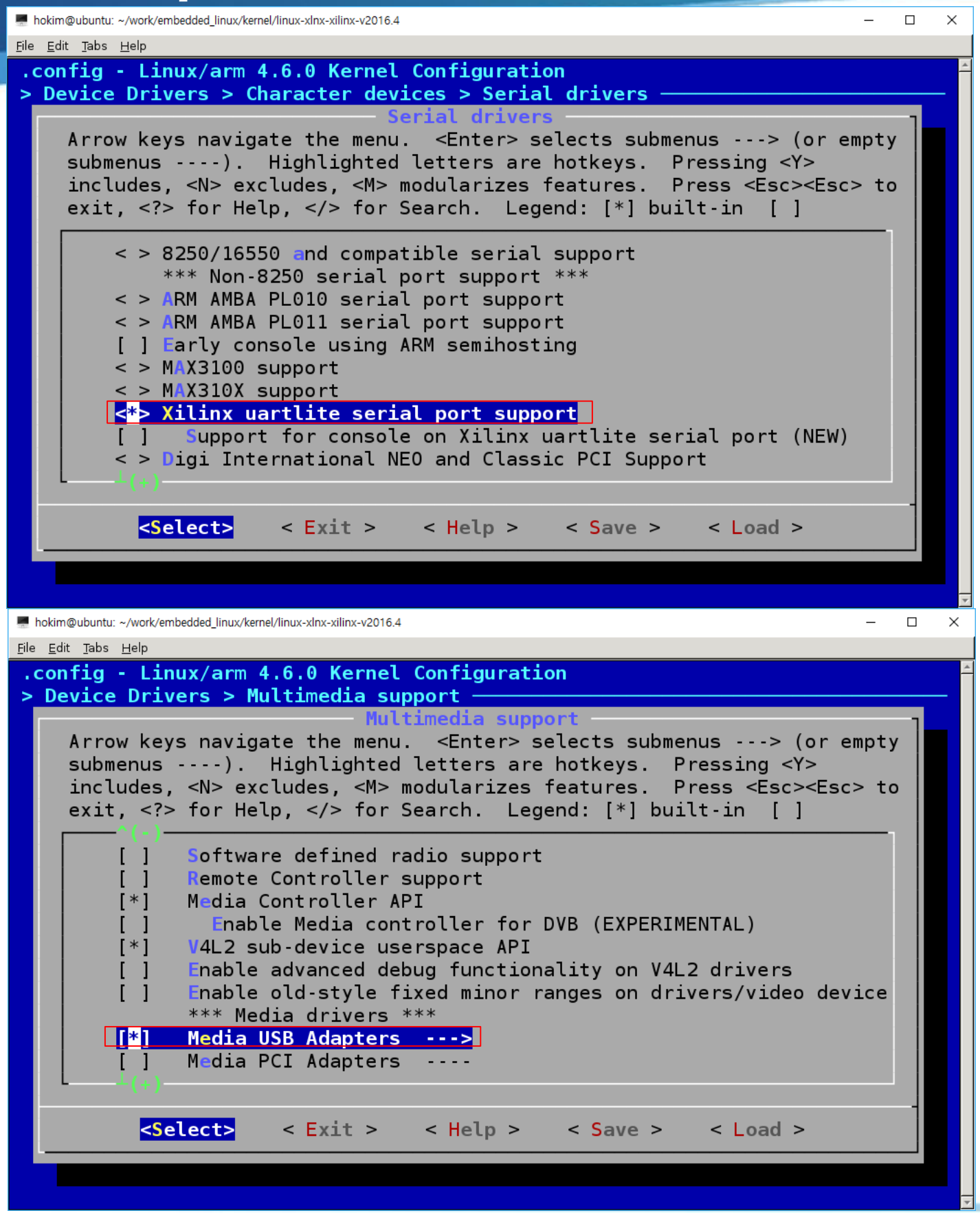
hokim@ubuntu: ~/work/embedded_linux/kernel/linux-xlnx-xilinx-v2016.4

File Edit Tabs Help

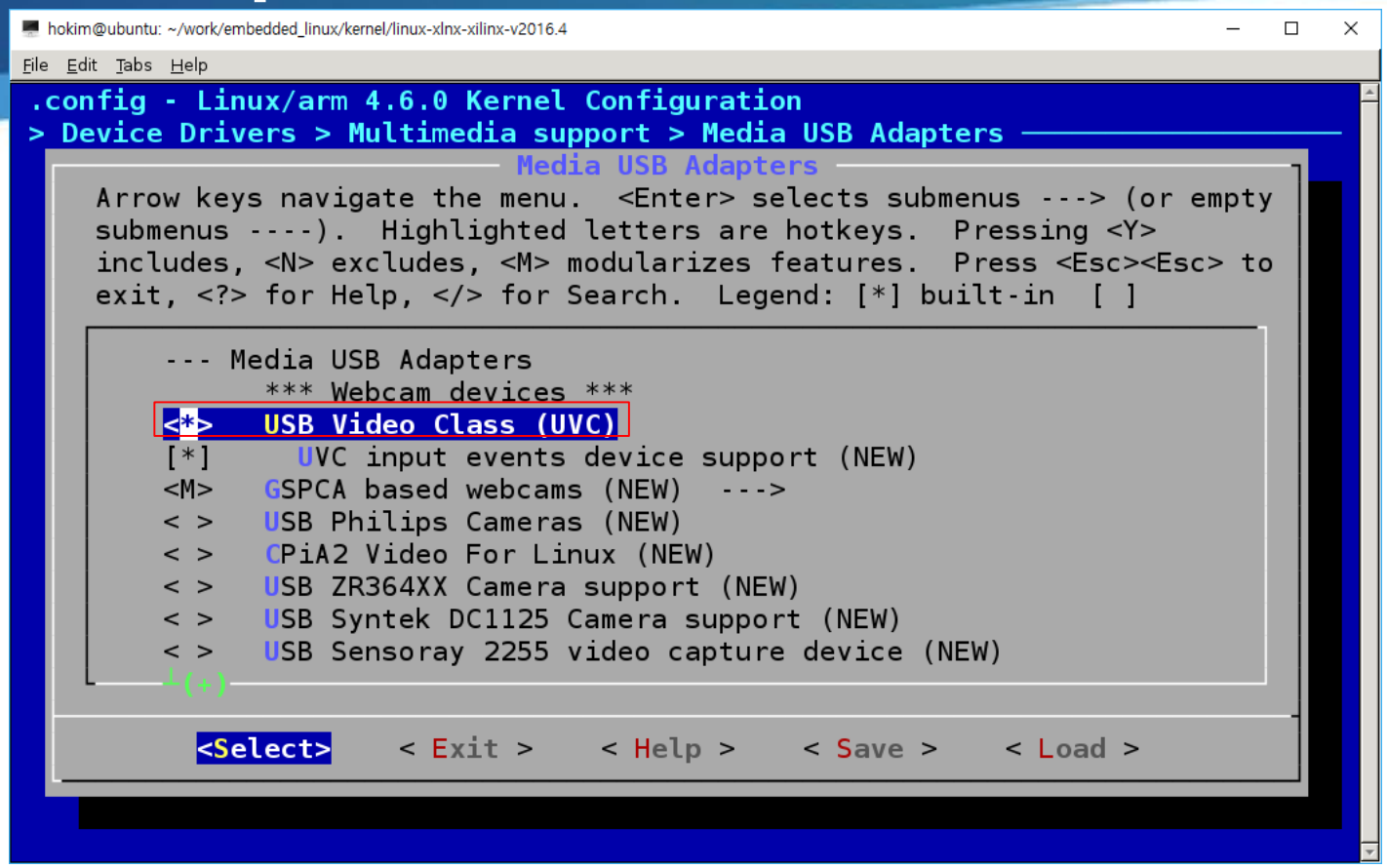
```
.config - Linux/arm 4.6.0 Kernel Configuration
> Device Drivers > I2C support > I2C Hardware Bus support
                    I2C Hardware Bus support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]
      < > GPIO-based bitbanging I2C
      < > ST-Ericsson Nomadik/Ux500 I2C Controller
      < > OpenCores I2C Controller
      < > PCA9564/PCA9665 as platform device
      < > Rockchip RK3xxx I2C adapter
      < > Simtec Generic I2C interface
      < > ARM Versatile/Realview I2C bus support
      <*> Xilinx I2C Controller
          *** External I2C/SMBus adapter drivers ***
      < > Diolan U2C-12 USB adapter
          <Select>    < Exit >    < Help >    < Save >    < Load >
```

Linux Kernel Compile

INIPRO



Linux Kernel Compile




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

Root File System Build

INIPRO

```
$ 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

INIPRO

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 lsb-release vim nano sudo openssh-server udev usbutils u-boot-tools
device-tree-compiler kmod net-tools wpasupplicant parted rfkill lshw wireless-tools gcc g++ cmake python git i2c-tools iputils-
ping lxterminal leafpad linux-firmware
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
```

SFTP File Transfer (Ubuntu → Windows)

INIPRO

using MobaXterm

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

ubuntu-core_armhf_16.04.img
(https://1drv.ms/u/s!AugbD-Nk6q_jhTrgintgtHWlbwI0)

Boot.bin Build

INIPRO

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
```

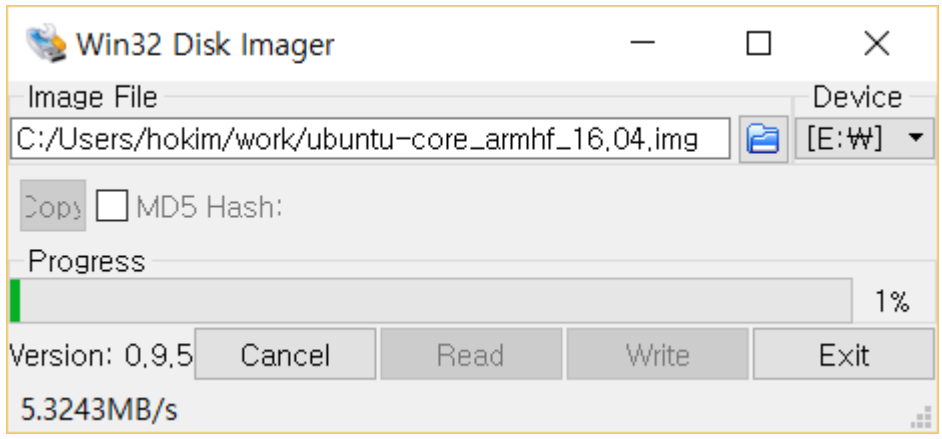
In cmd window for Vivado

```
C:\> cd c:\Users\hokim\work\embedded_linux\vivado
C:\> tclsh bootbin.tcl
```

output : boot.bin

SD Card

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

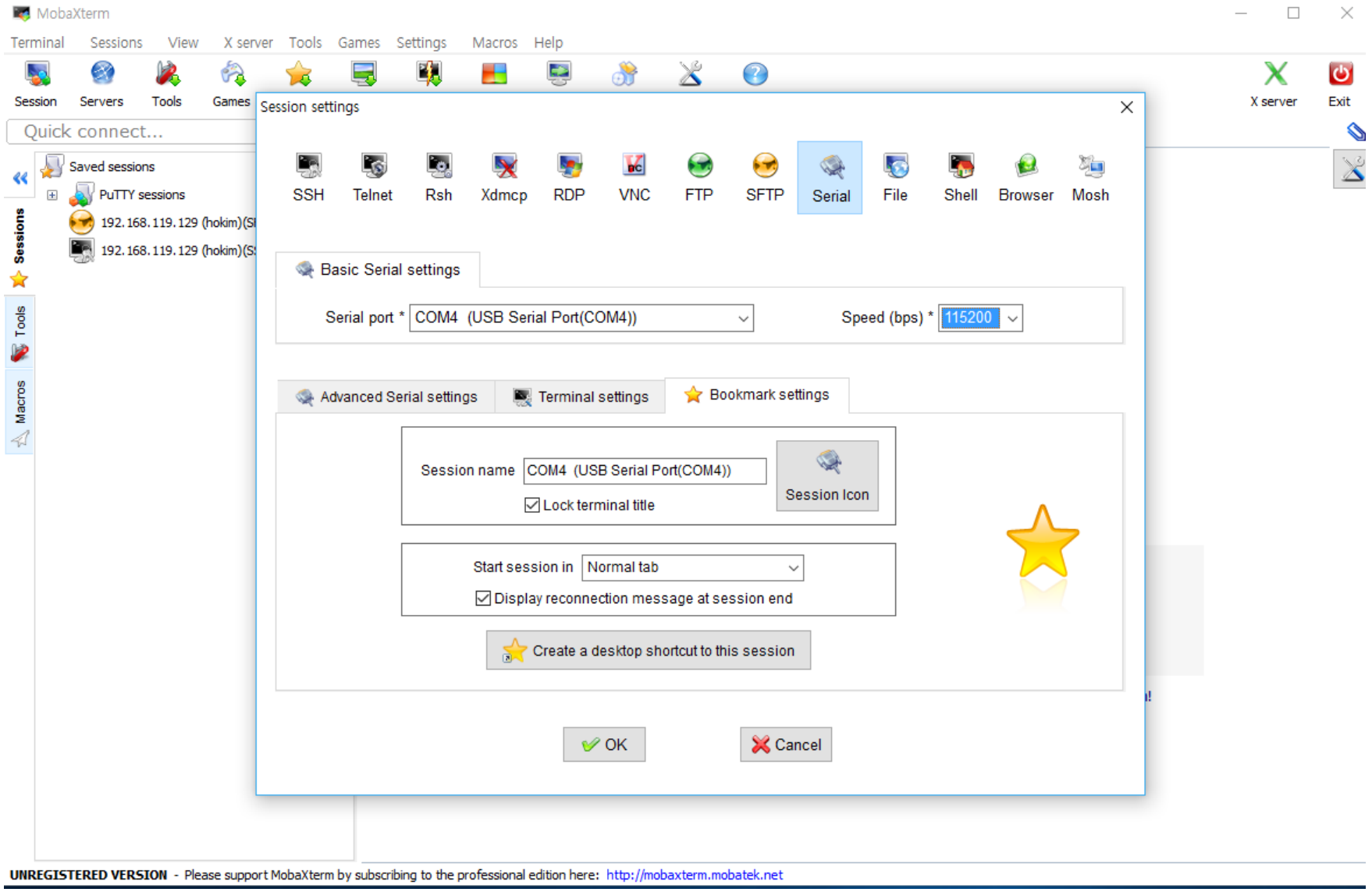


Copy
uImage, uEnv.txt,
vivado\boot.bin, vivado\devicetree.dtb
To E:

Boot zybo with SD card

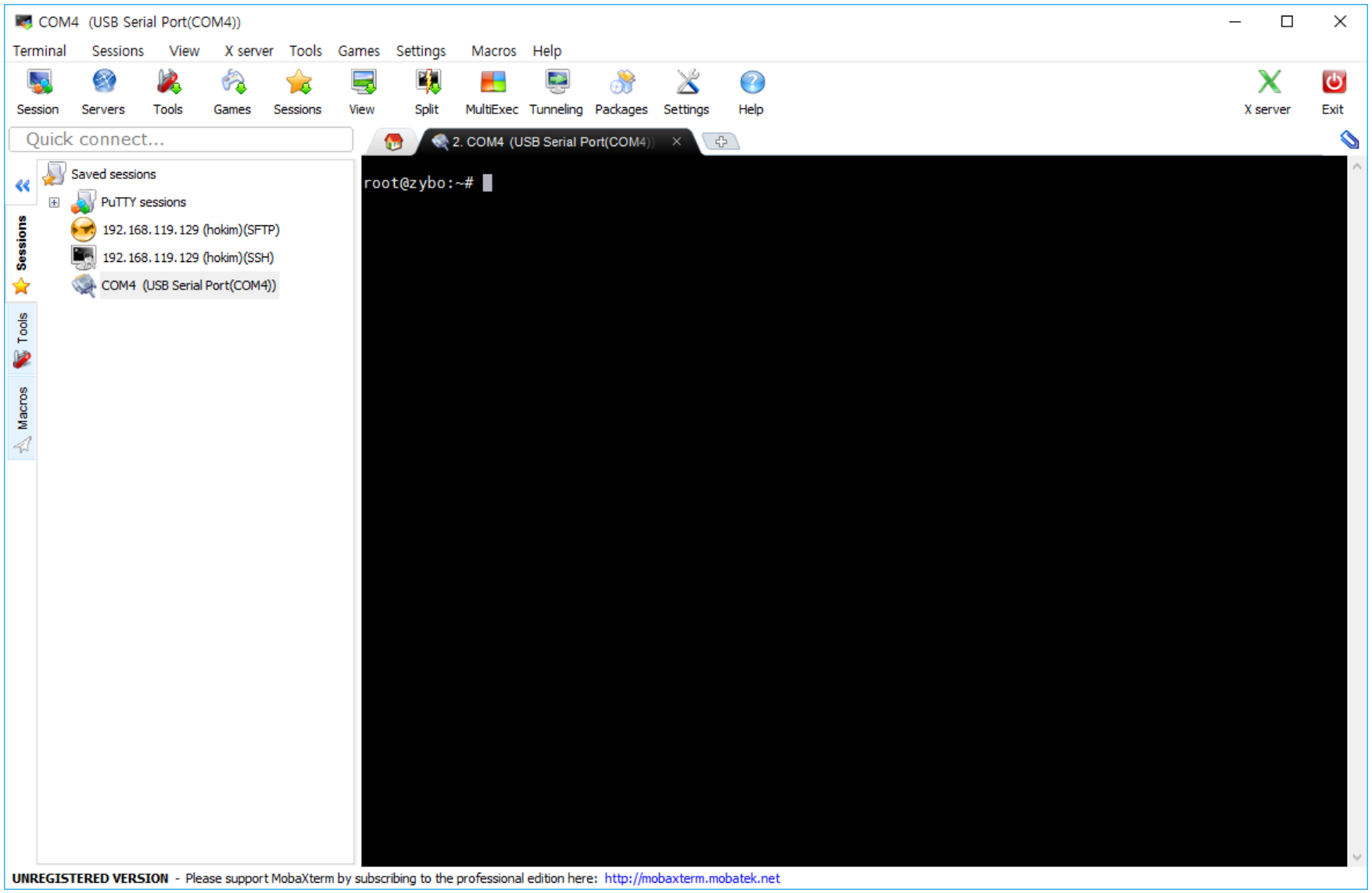
Post install

Log in through UART console using MobaXterm



Post install

Log in through UART console using MobaXterm



Post install

```
# 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

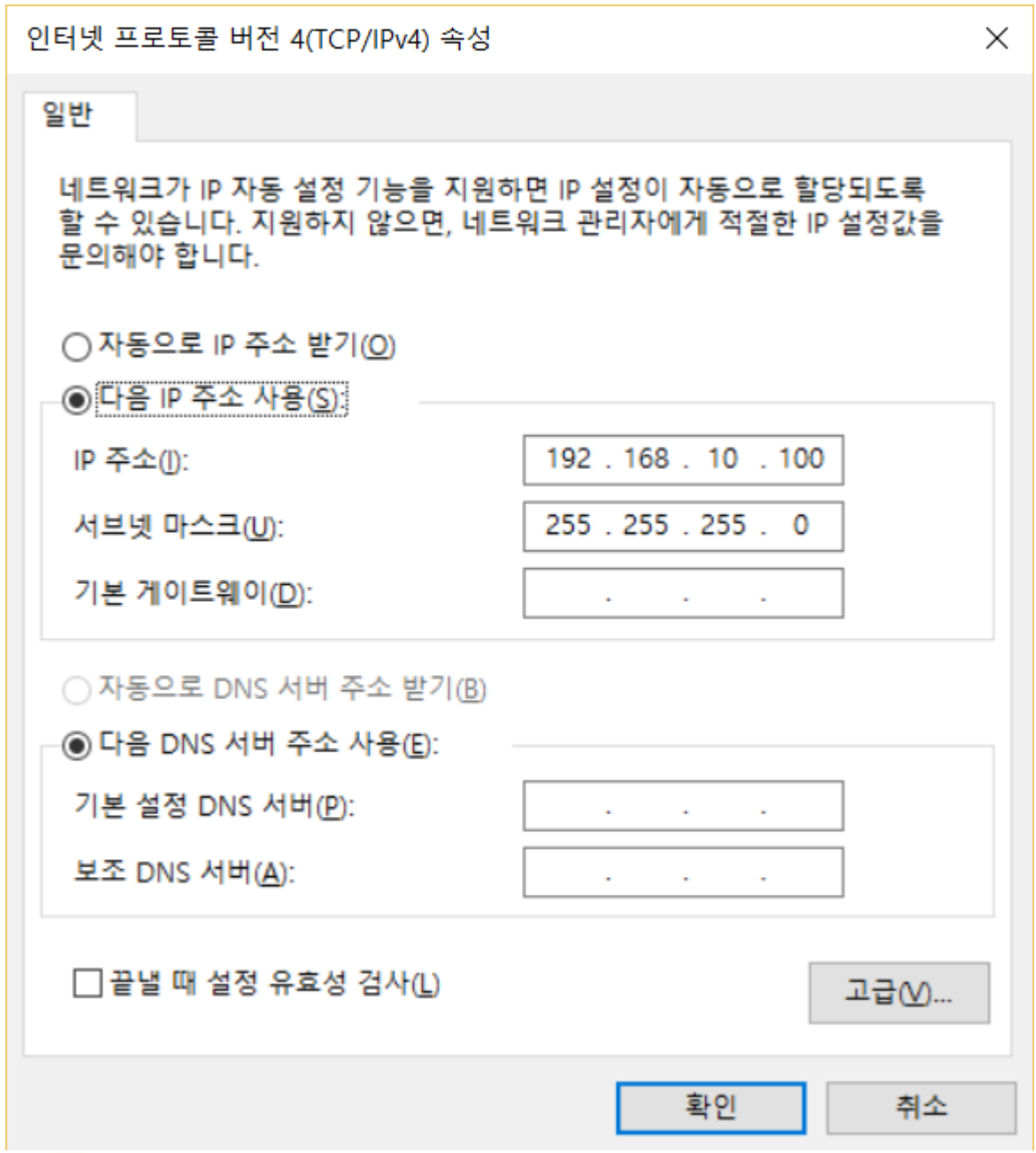
```
allow-hotplug eth0
iface eth0 inet static
address 192.168.10.10
netmask 255.255.255.0
```

```
# halt
```

Turn off/on zybo

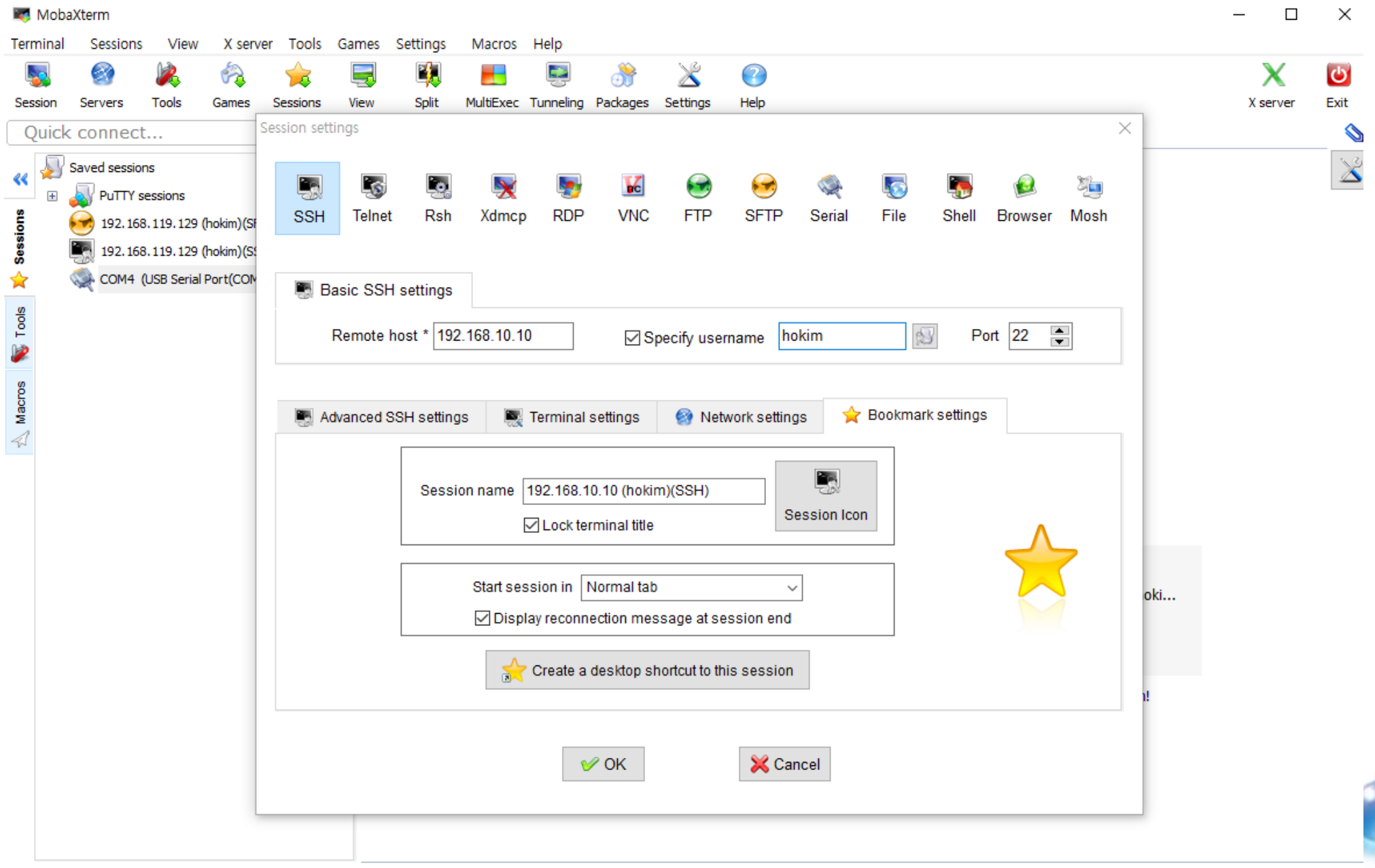
Post install

Log in through Ethernet using MobaXterm



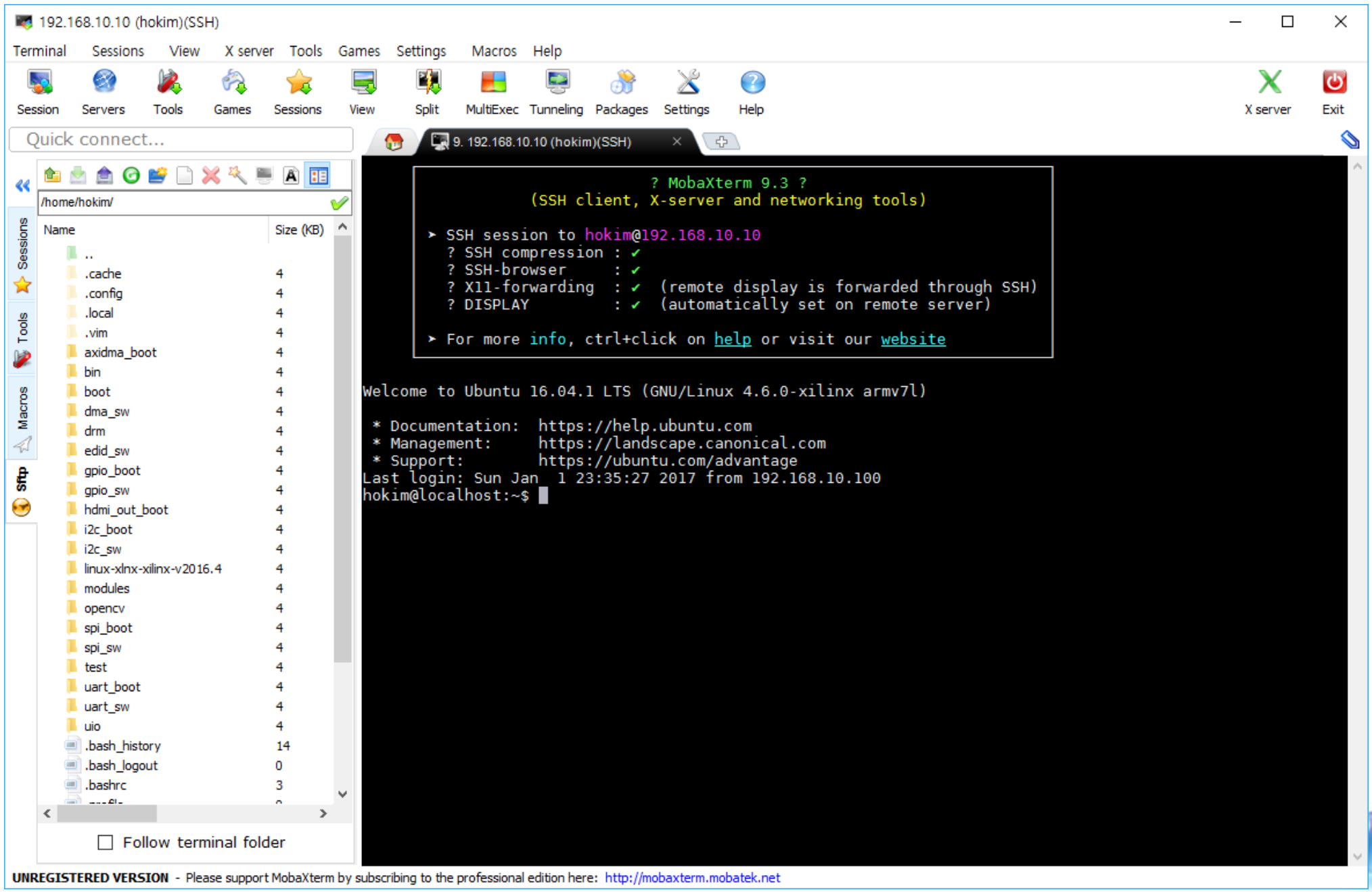
Post install

Log in through Ethernet using MobaXterm



Post install

Log in through Ethernet using MobaXterm



Post install

INIPRO

$ 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
```

$ lsusb

```
Bus 001 Device 002: ID 0bda:8179 Realtek Semiconductor Corp. RTL8188EUS 802.11n Wireless Network Adapter
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Post install

INIPRO

```
$ sudo nano /etc/udev/rules.d/10-network.rules
```

/etc/udev/rules.d/10-network.rules

```
ACTION=="add", SUBSYSTEM=="net", ATTRS{idVendor}=="0bda", ATTRS{idProduct}=="8179", NAME="wlan0"
```

```
$ sudo nano /etc/network/interfaces.d/wlan0
```

/etc/network/interfaces.d/wlan0

```
allow-hotplug wlan0
Iface wlan0 inet dhcp
        pre-up wpa_supplicant -B -D wext -i wlan0 -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"
}
```

Post install

```
$ sudo halt
```

Turn off/on zybo

```
$ 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
```

Post install

INIPRO

```
$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0a:35:00:01:22
          inet addr:192.168.10.10  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20a:35ff:fe00:122/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:463 errors:0 dropped:0 overruns:0 frame:0
          TX packets:300 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:37295 (37.2 KB)  TX bytes:39886 (39.8 KB)
          Interrupt:145 Base address:0xb000

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:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

wlan0     Link encap:Ethernet  HWaddr 00:e0:4c:15:44:d8
          inet addr:192.168.0.5  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::2e0:4cff:fe15:44d8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:960 errors:0 dropped:1808 overruns:0 frame:0
          TX packets:26 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:308669 (308.6 KB)  TX bytes:3036 (3.0 KB)
```

Linux Architecture

- **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

Linux Architecture

- **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

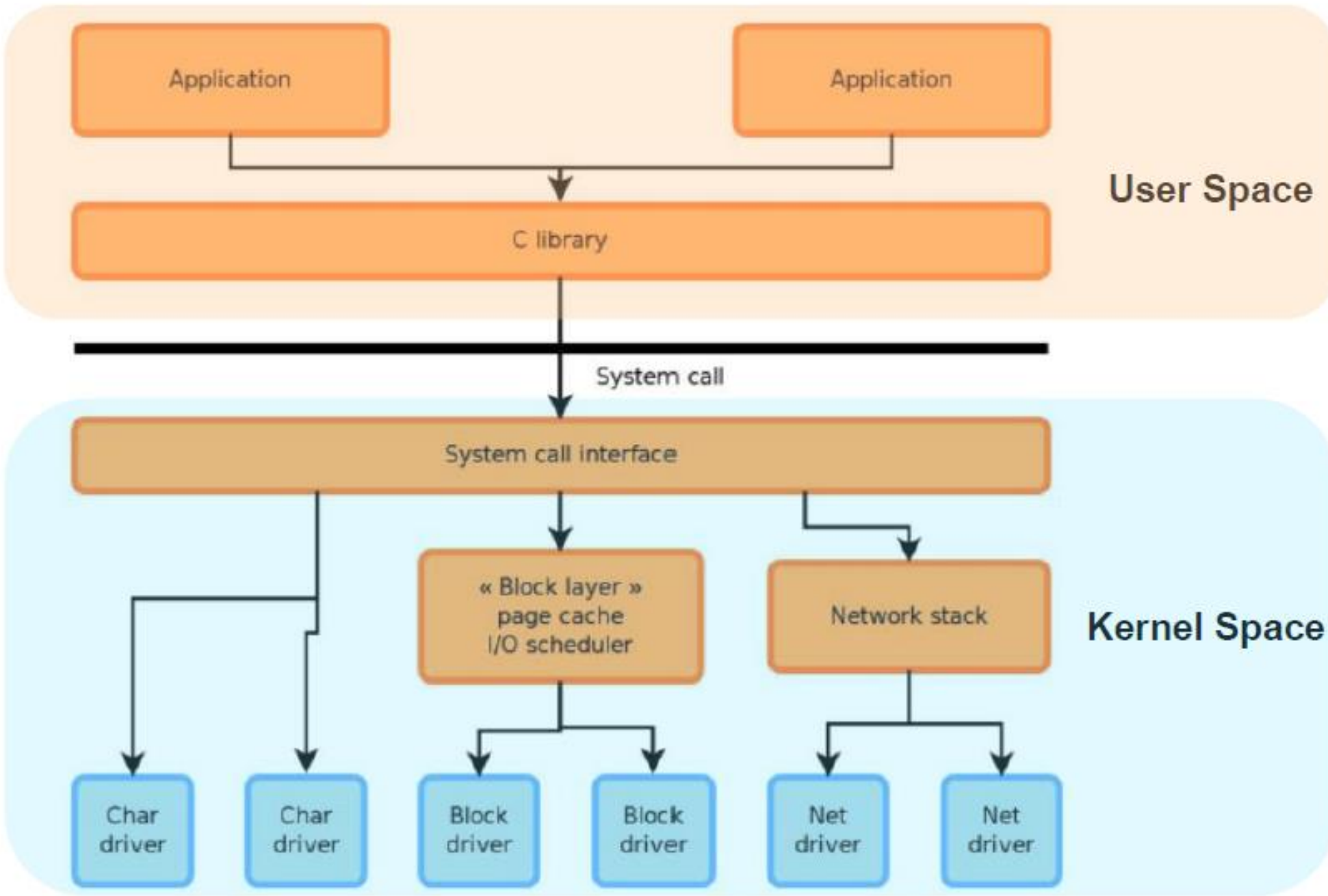
- **System calls run in kernel mode on the user's behalf and are provided by the kernel itself**

Linux Architecture

INIPRO

- 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 an 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 proper, or with specific drivers and frameworks of the kernel

Linux Architecture



Linux Device Model

- 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

Linux Device Types

- **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

Linux Kernel Frameworks

INIPRO

- **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

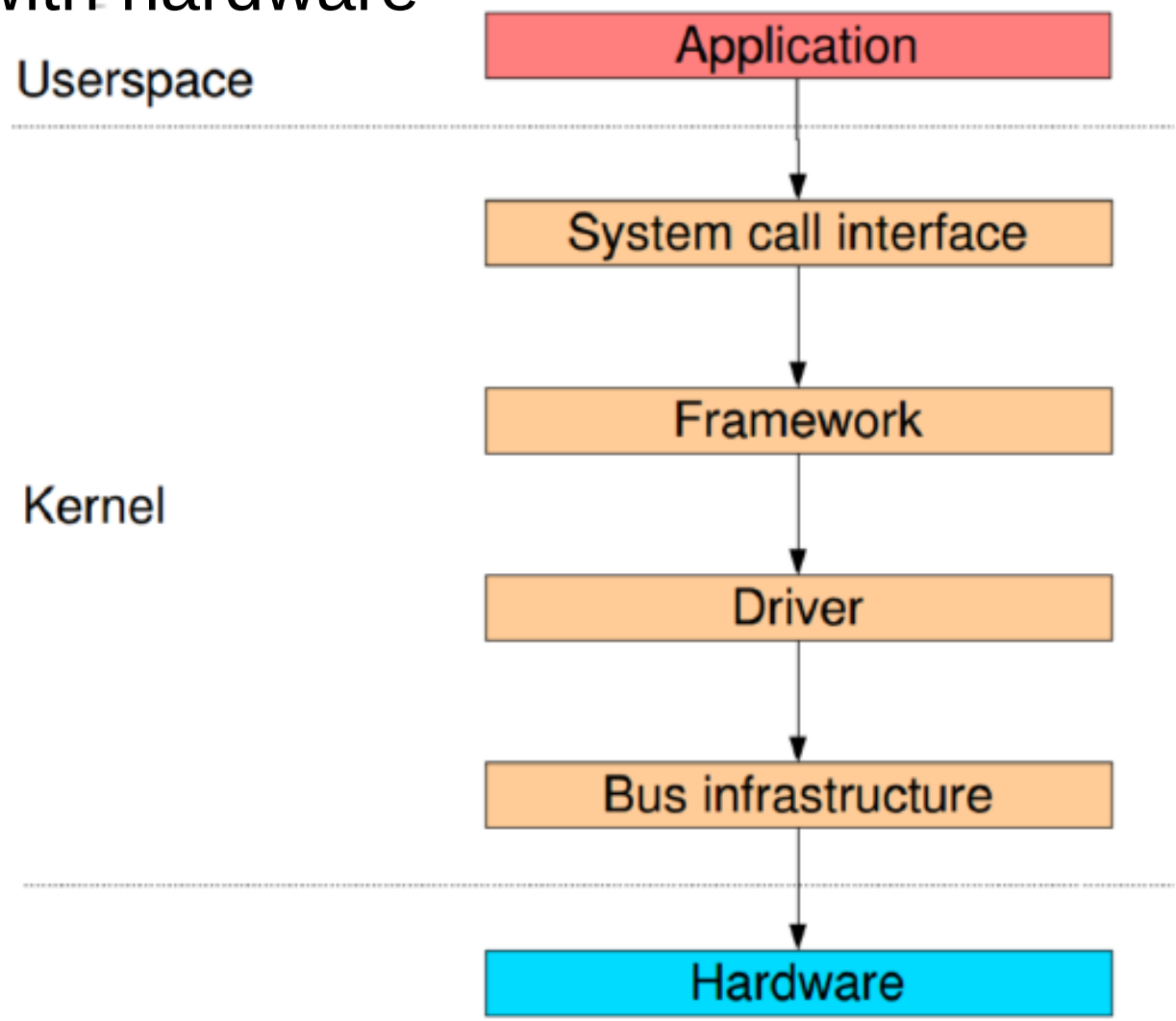
- sockfd = socket(AF_INET, SOCK_STREAM, 0)

Linux Kernel Layers Focused on Frameworks

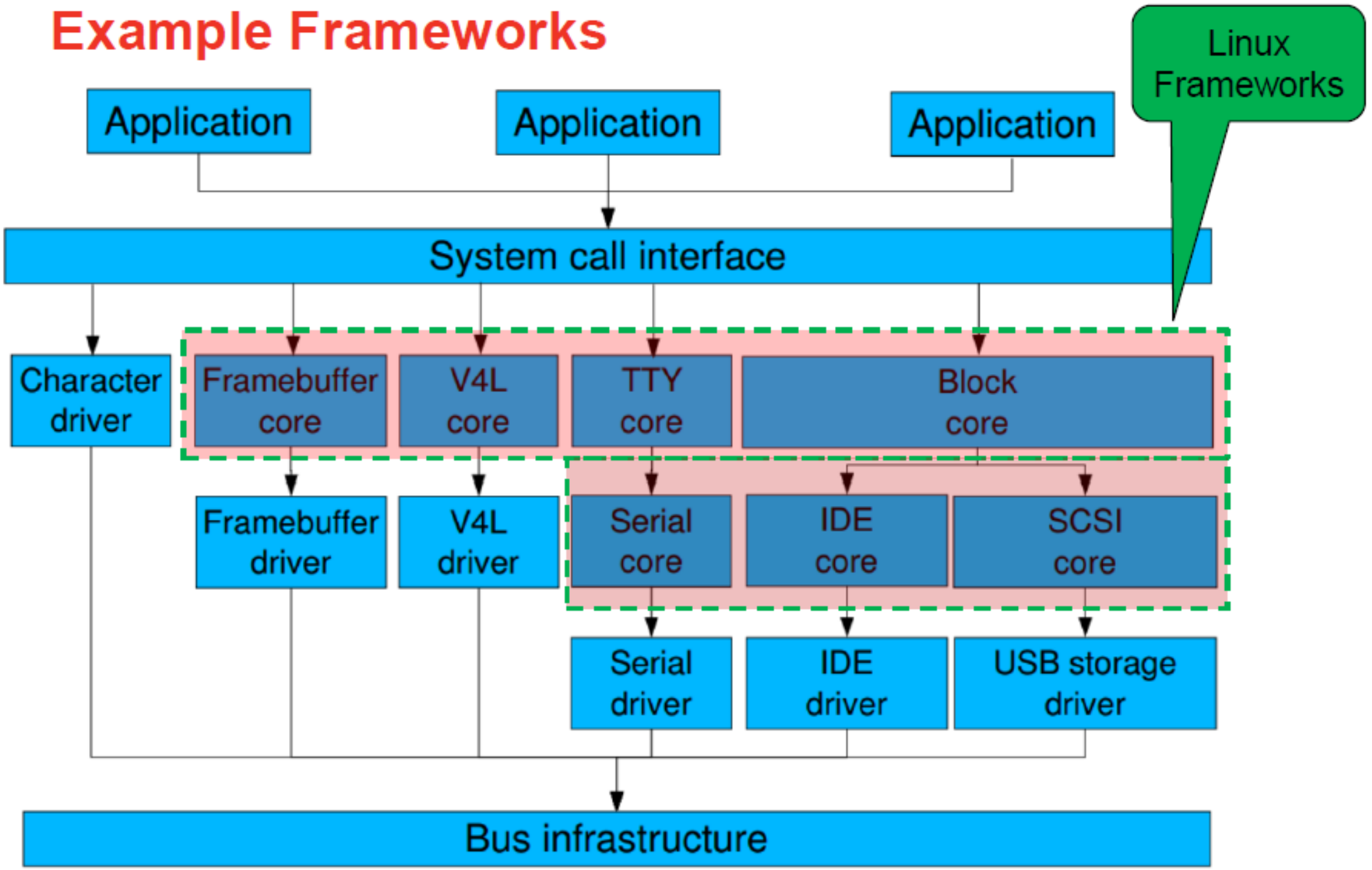
- **A driver is always interfacing with:**

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



Virtual File Systems - Overview

- **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

Virtual File Systems - sysfs

- **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.**

Device Tree In A Nutshell

INIPRO

- 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)

Device Tree In A Nutshell

INIPRO

- **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>;
        } ;
};
```

Device Tree In A Nutshell

- 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";
};
```

Note the "&" used to reference an existing node (rather than creating a new node

=

```
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";
} ;
```

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

Linux GPIO

INIPRO

- **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

Linux GPIO Userspace Interface

INIPRO

- **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

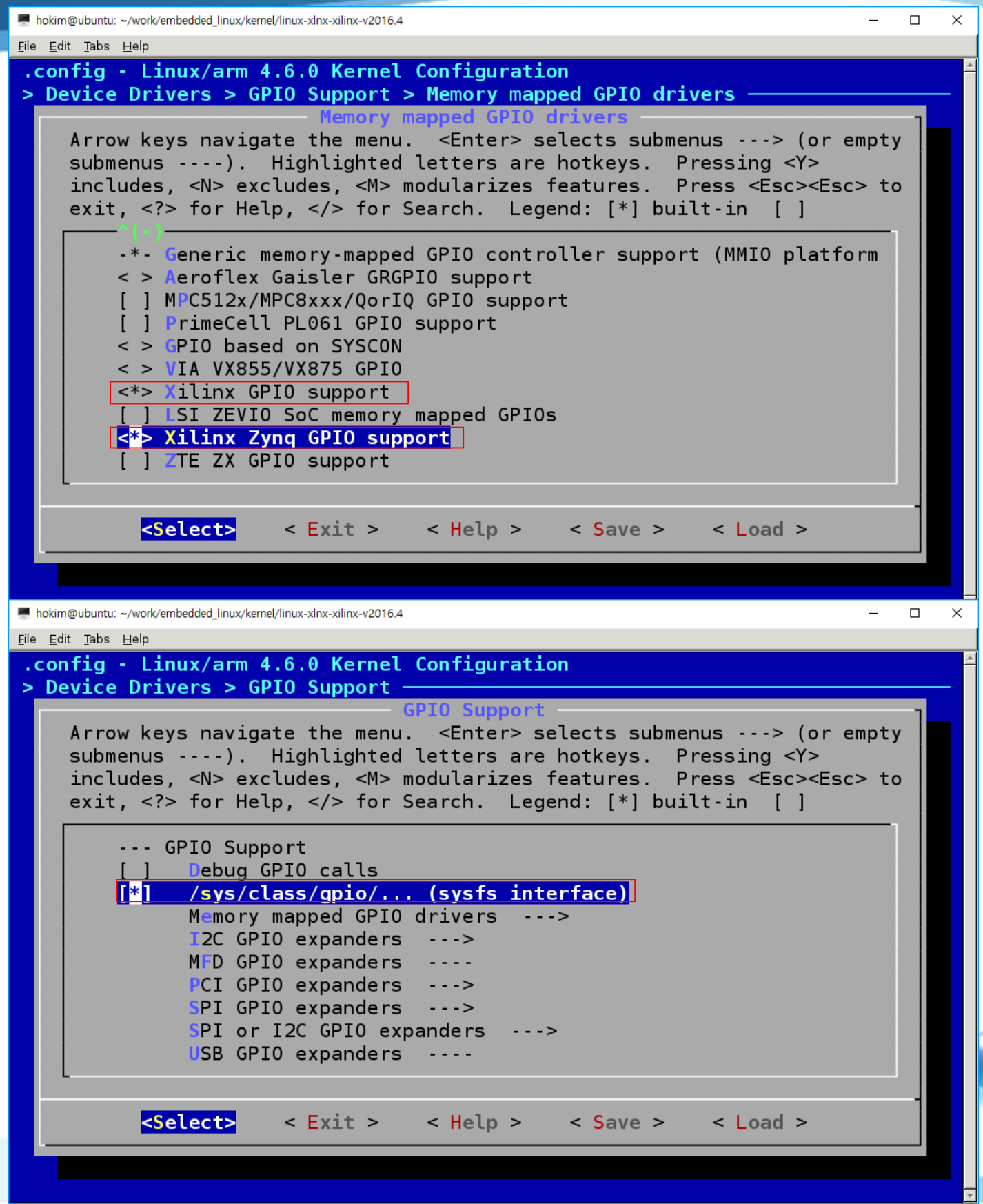
gpioN/direction Write “in” or “out” to set pin direction

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

Linux GPIO Userspace Interface

INIPRO

Kernel Configuration



Linux GPIO Userspace Interface

INIPRO

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)))
        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);
}
```

Legacy UserSpace Driver Methods (/dev/mem)

INIPRO

- 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

INIPRO

gpio_sw/mmap/*main.c*

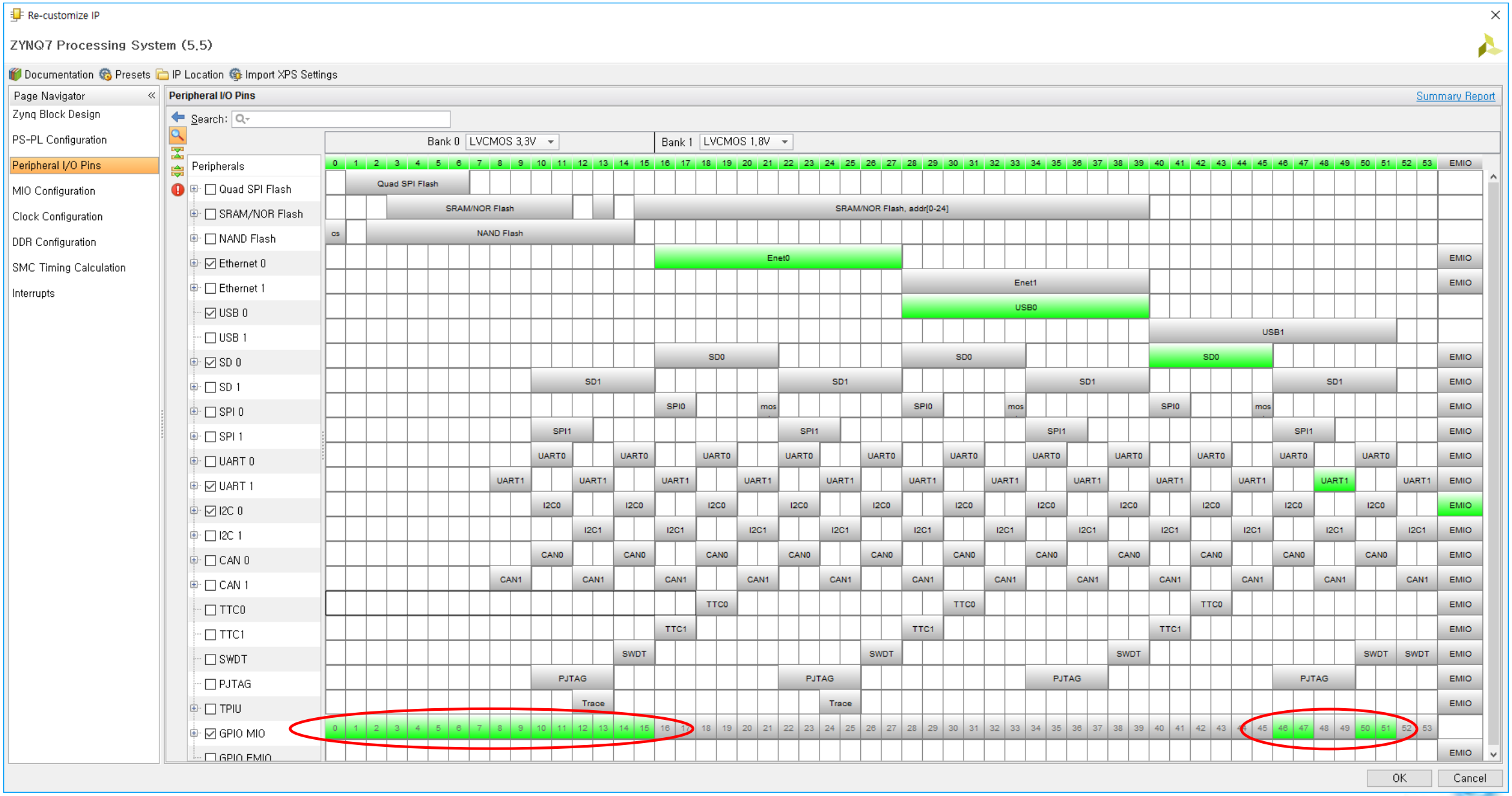
```
#include <stdio.h>
#include <stdint.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <signal.h>
#include "xgpio.h"

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

INIPRO

Based on Zynq7 PS Re-customize IP of embedded_linux project



Zynq7 PS Re-customize IP for GPIO

INIPRO

Based on Zynq7 PS Re-customize IP of embedded_linux project

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 — Summary Report

Bank 0 I/O Voltage: LVCMOS 3.3V — Bank 1 I/O Voltage: LVCMOS 1.8V

Search:

Peripheral	IO	Signal	IO Type	Speed	Pullup	Direction	Polarity
GPIO							
GPIO MIO	MIO						
GPIO	MIO 0	gpio[0]	LVCMOS 3.3V	slow	disabled	inout	
GPIO	MIO 1	gpio[1]	LVCMOS 3.3V	slow	disabled	inout	
GPIO	MIO 2	gpio[2]	LVCMOS 3.3V	slow	disabled	inout	
GPIO	MIO 3	gpio[3]	LVCMOS 3.3V	slow	disabled	inout	
GPIO	MIO 4	gpio[4]	LVCMOS 3.3V	slow	disabled	inout	
GPIO	MIO 5	gpio[5]	LVCMOS 3.3V	slow	disabled	inout	
GPIO	MIO 6	gpio[6]	LVCMOS 3.3V	slow	disabled	inout	
GPIO	MIO 7	gpio[7]	LVCMOS 3.3V	slow	disabled	out	
GPIO	MIO 8	gpio[8]	LVCMOS 3.3V	slow	disabled	out	
GPIO	MIO 9	gpio[9]	LVCMOS 3.3V	slow	disabled	inout	
GPIO	MIO 10	gpio[10]	LVCMOS 3.3V	slow	disabled	inout	
GPIO	MIO 11	gpio[11]	LVCMOS 3.3V	slow	disabled	inout	
GPIO	MIO 12	gpio[12]	LVCMOS 3.3V	slow	disabled	inout	
GPIO	MIO 13	gpio[13]	LVCMOS 3.3V	slow	disabled	inout	
GPIO	MIO 14	gpio[14]	LVCMOS 3.3V	slow	disabled	inout	
GPIO	MIO 15	gpio[15]	LVCMOS 3.3V	slow	disabled	inout	
GPIO	MIO 46	gpio[46]	LVCMOS 1.8V	slow	disabled	inout	
GPIO	MIO 47	gpio[47]	LVCMOS 1.8V	slow	disabled	inout	
GPIO	MIO 50	gpio[50]	LVCMOS 1.8V	slow	disabled	inout	
GPIO	MIO 51	gpio[51]	LVCMOS 1.8V	slow	disabled	inout	
EMIO GPIO (Width)							

OK | Cancel

Zynq7 PS Re-customize IP for GPIO

INIPRO

Based on Zynq7 PS Re-customize IP of embedded_linux project

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

PS-PL Configuration — Summary Report

Search:

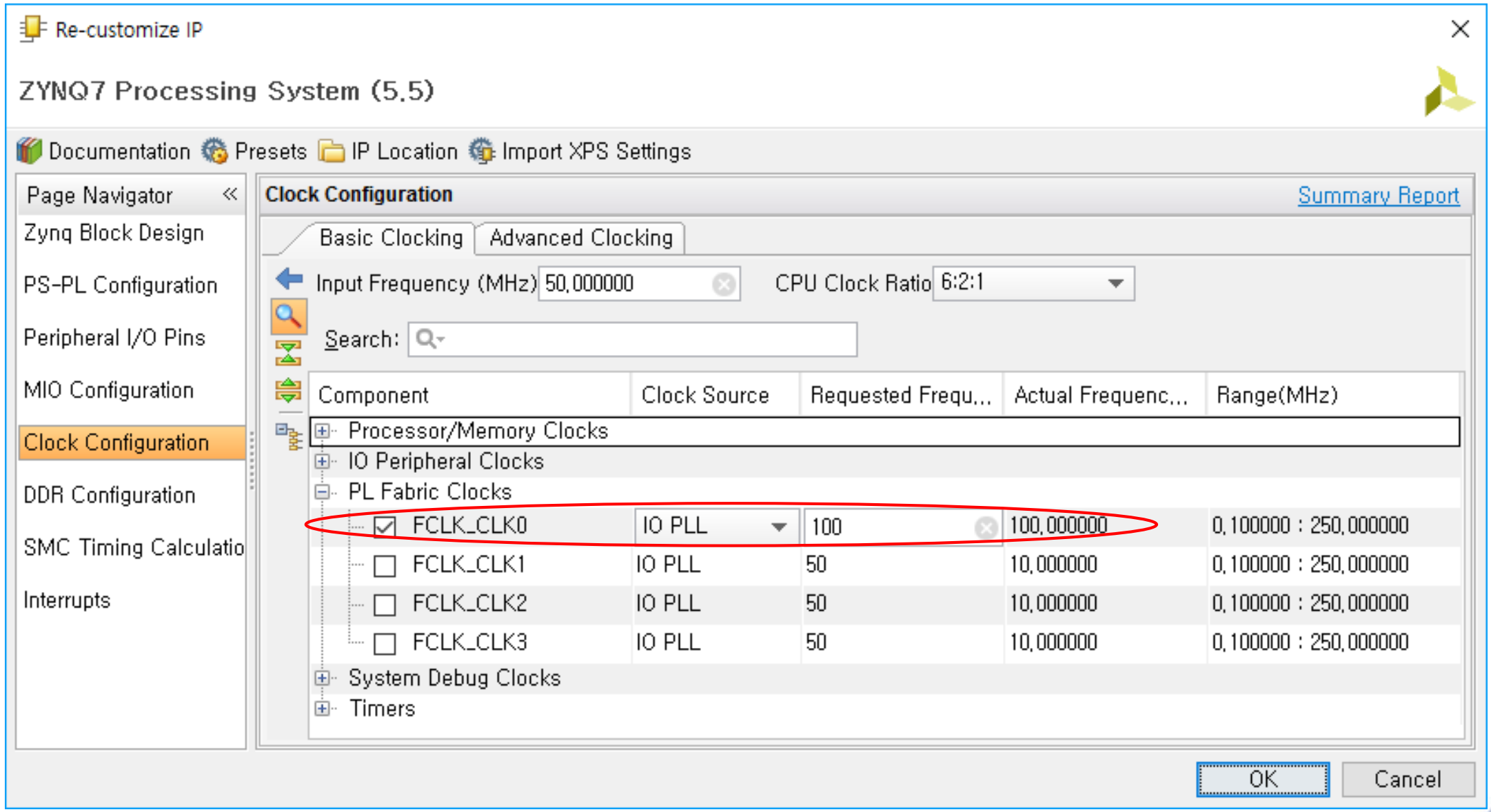
Name	Select	Description
General		
UART0 Baud Rate	115200	Baud rate is generated with internally fixed UART Ref Clock Freq=10...
UART1 Baud Rate	115200	Baud rate is generated with internally fixed UART Ref Clock Freq=10...
PL AXI idle Port	☐	Enables idle AXI signal to the PS used to indicate that there are no ...
DDR ARB bypass Port	☐	Enables DDR urgent/arb signal used to signal a critical memory sta...
PS-PL Debug interface	☐	Enables PL debug signals to PS and vice-versa
FTM Trace data interface	☐	Enables FTM Trace AXI stream interface used to capture data from PL to PS debug system
FTM Trace buffer	0	
FTM Data edge detector	0	Stores trace data in the FIFO when the data changes as marked by ...
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 be...
Include ACP transaction checker	☐	Enables ACP transaction checker.
Trace data/control signal pipeline width	8	Enables configurable number of pipeline stages on the TRACE DAT...
Power-on reset(POR) 4k timer	☐	Enables power-on reset(POR) 4k timer. By default, 64k timer is used.
Processor event interface	☐	Enables event bus which provides a low-latency and direct mecha...
Address Editor		
Enable Clock Triggers		
Enable Clock Resets		
FCLK_RESET0_N	☑	Enables general purpose reset signal 0 for PL logic
FCLK_RESET1_N	☐	Enables general purpose reset signal 1 for PL logic
FCLK_RESET2_N	☐	Enables general purpose reset signal 2 for PL logic
FCLK_RESET3_N	☐	Enables general purpose reset signal 3 for PL logic
AXI Non Secure Enablement	0	Enable AXI Non Secure Transaction
GP Master AXI Interface		
M AXI GP0 interface	☑	Enables General purpose AXI master interface 0
M AXI GP1 interface	☐	Enables General purpose AXI master interface 1
GP Slave AXI Interface		

OK | Cancel

Zynq7 PS Re-customize IP for GPIO

INIPRO

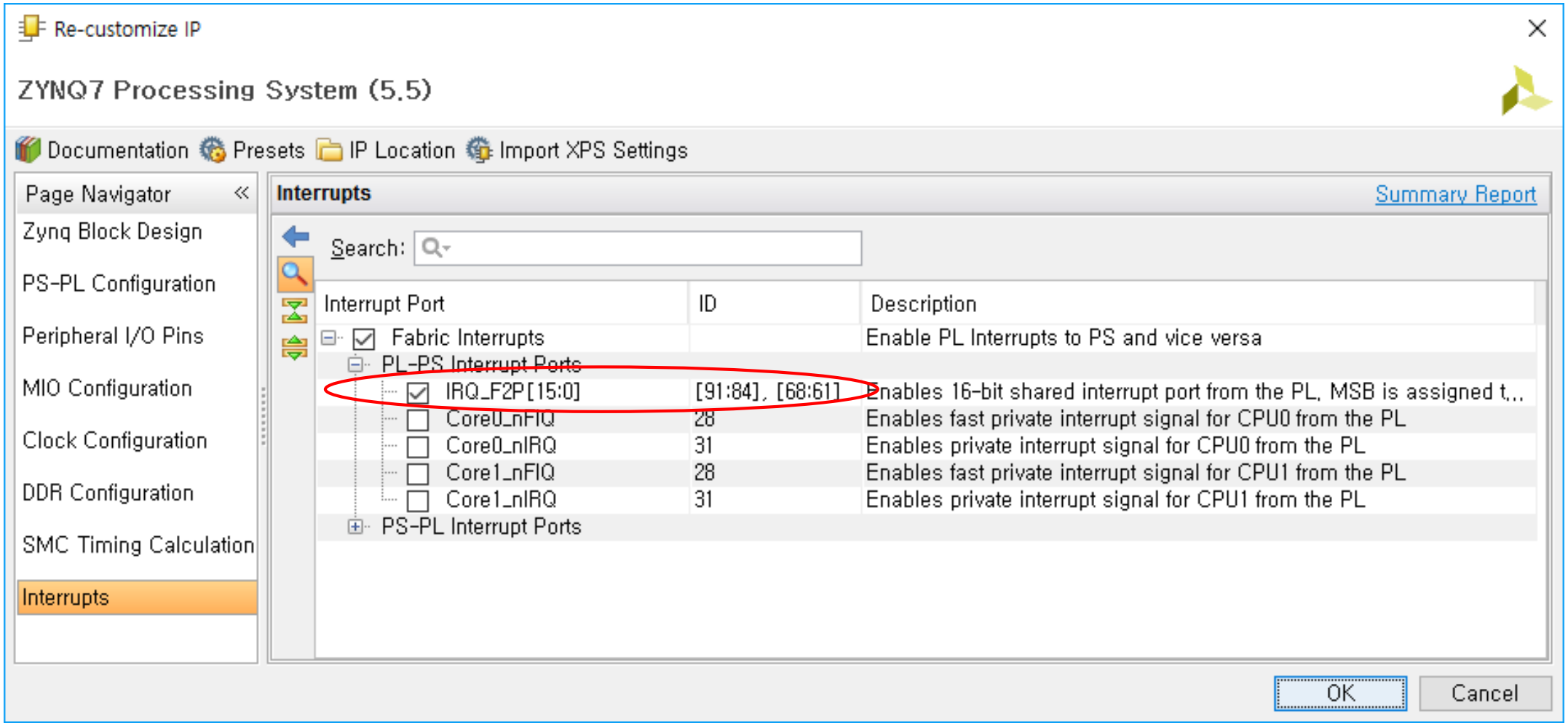
Based on Zynq7 PS Re-customize IP of embedded_linux project



Zynq7 PS Re-customize IP for GPIO

INIPRO

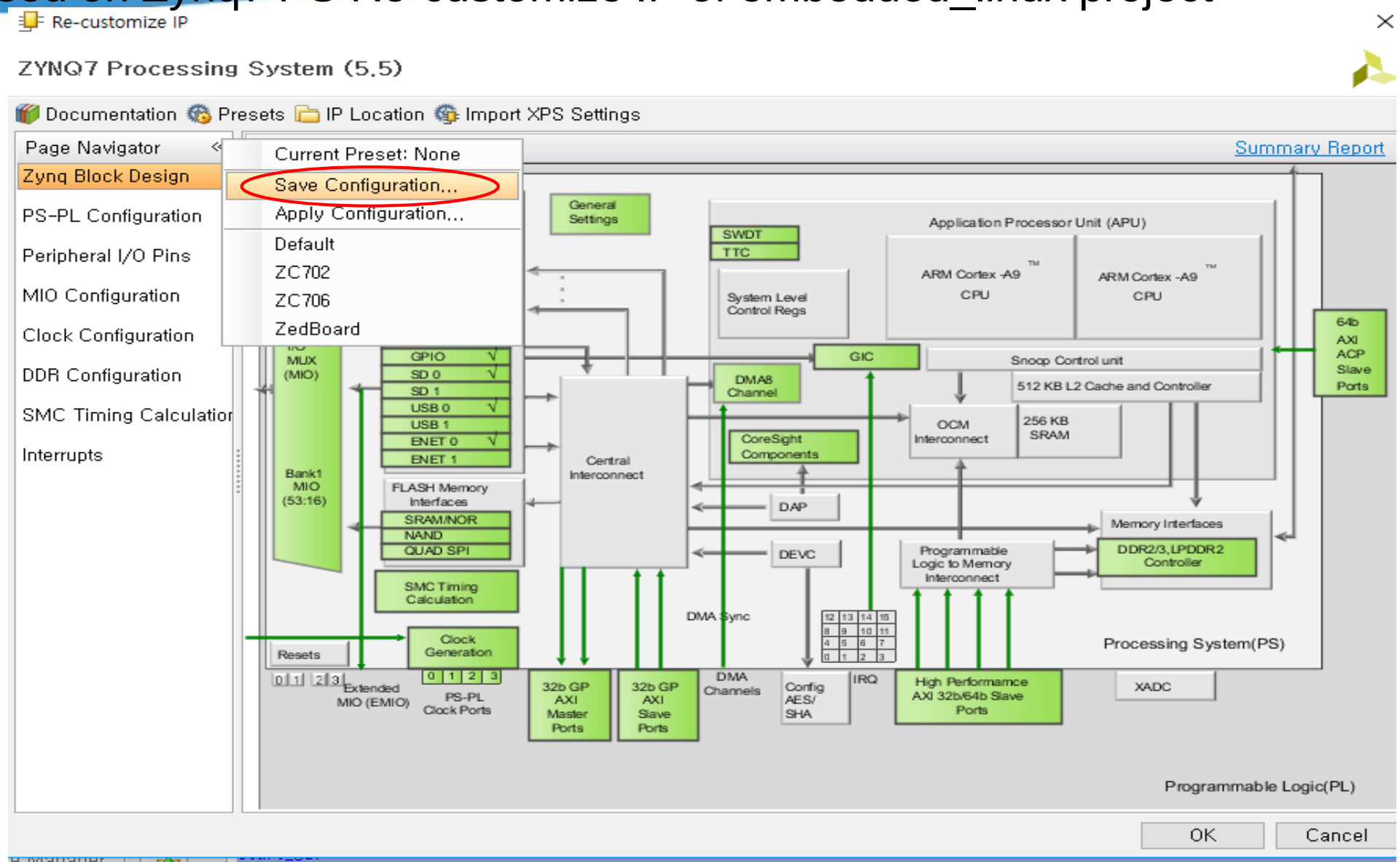
Based on Zynq7 PS Re-customize IP of embedded_linux project

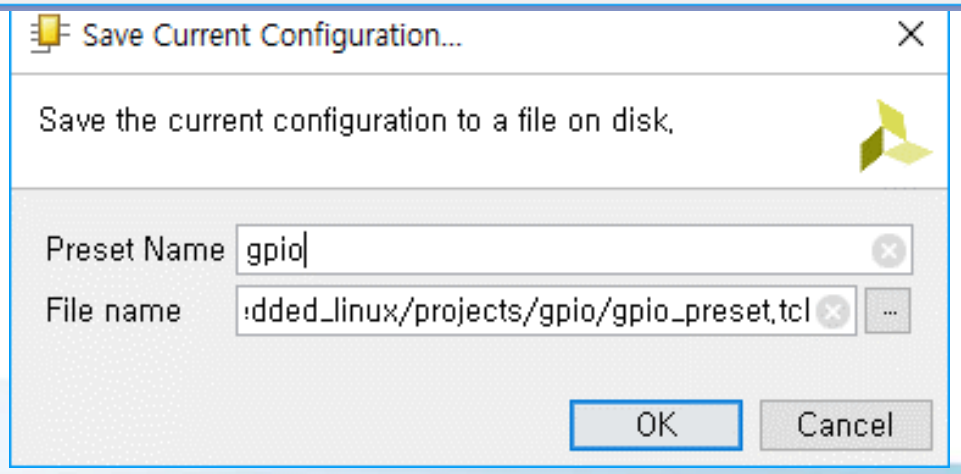


Zynq7 PS Re-customize IP for GPIO

INIPRO

Based on Zynq7 PS Re-customize IP of embedded_linux project





Project for GPIO

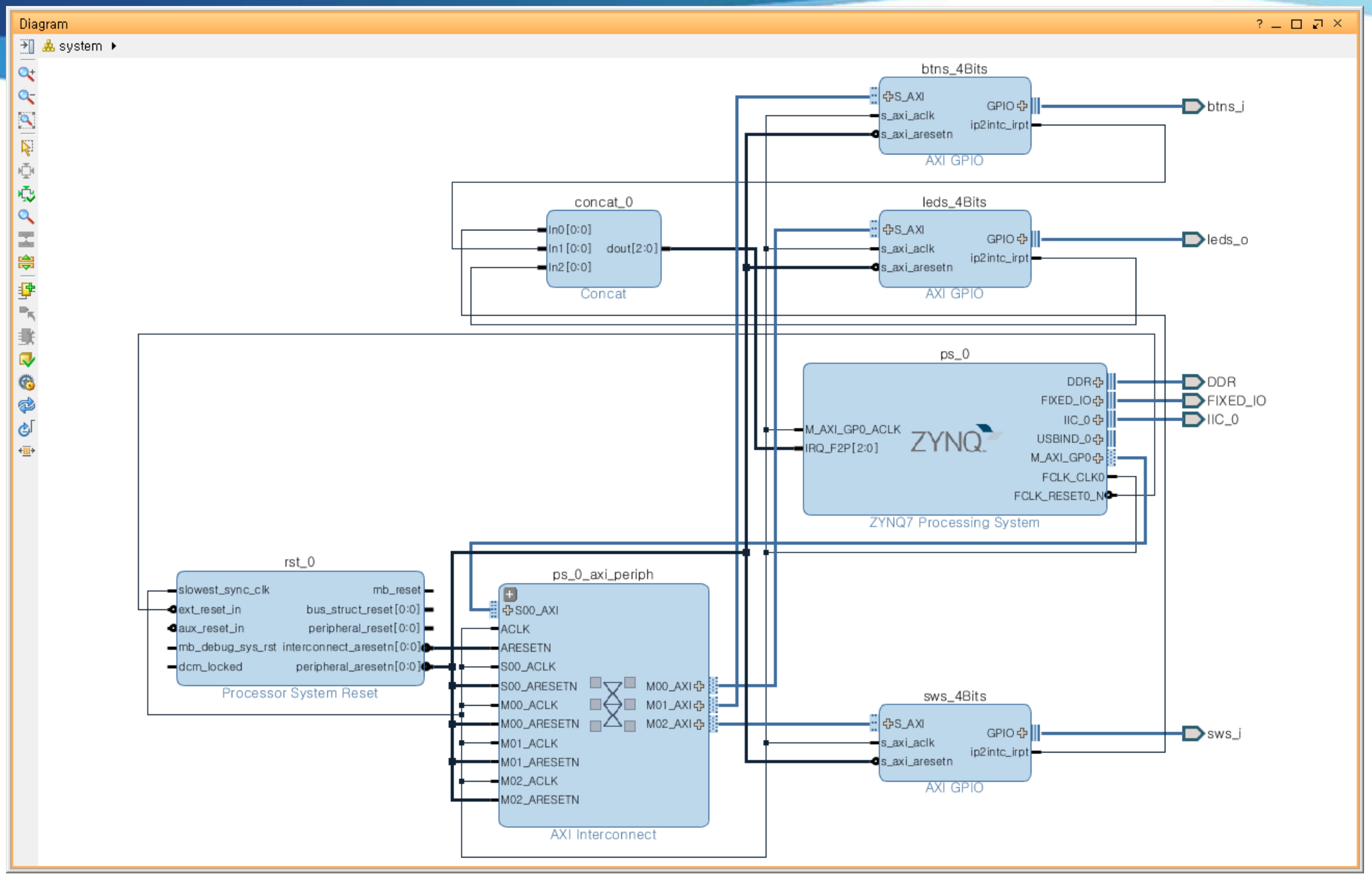
https://github.com/inipro/embedded_linux/projects/gpio

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…

Project for GPIO



Address Editor

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
ps_0					
Data (32 address bits : 0x40000000 [1G])					
btns_4Bits	S_AXI	Reg	0x4000_1000	4K	0x4000_1FFF
leds_4Bits	S_AXI	Reg	0x4000_0000	4K	0x4000_0FFF
sws_4Bits	S_AXI	Reg	0x4000_2000	4K	0x4000_2FFF

Bit Generation for GPIO

INIPRO

gpio - [C:/Users/hokim/work/embedded_linux/projects/gpio/gpio/gpio.xpr] - Vivado 2016.4

File Edit Flow Tools Window Layout View Help

Quick Access

Default Layout

Ready

Flow Navigator

- Language Templates
- IP Catalog

IP Integrator
- Create Block Design
- Open Block Design
- Generate Block Desig

Simulation
- Simulation Settings
- Run Simulation

RTL Analysis
- Elaboration Settings
- Open Elaborated Desi

Synthesis
- Synthesis Settings
- Run Synthesis
- Open Synthesized De

Implementation
- Implementation Settin
- Run Implementation
- Open Implemented De

Program and Debug
- Bitstream Settings
- Generate Bitstream
- Open Hardware Mana

Block Design - system

Design

- system
  - External Interfaces
  - Interface Connections
  - Nets
  - btns_4Bits (AXI GPIO:2,0)
  - concat_0 (Concat:2,1)
  - leds_4Bits (AXI GPIO:2,0)

Sources | Design | Signals

Source File Properties

system.bd

Enabled

General | Properties

Diagram | Address Editor

system

Close

Tcl Console

```
Adding cell -- xilinx.com:ip:axi_gpio:2.0 - leds_4Bits
Adding cell -- xilinx.com:ip:axi_gpio:2.0 - btns_4Bits
Adding cell -- xilinx.com:ip:axi_gpio:2.0 - sws_4Bits
Adding cell -- xilinx.com:ip:xlconcat:2.1 - concat_0
Adding cell -- xilinx.com:ip:axi_crossbar:2.1 - xbar
Adding cell -- xilinx.com:ip:axi_protocol_converter:2.1 - auto_pc
Successfully read diagram <system> from BD file <C:/Users/hokim/work/embedded_linux/projects/gpio/gpio/gpio.srcs/sources_1/bd/system/
open_bd_design: Time (s): cpu = 00:00:15 ; elapsed = 00:00:10 . Memory (MB): peak = 853.527 ; gain = 71.754
```

Type a Tcl command here

Tcl Console | Messages | Log | Reports | Design Runs

System Net: ps_0_FCLK_RESET0_N

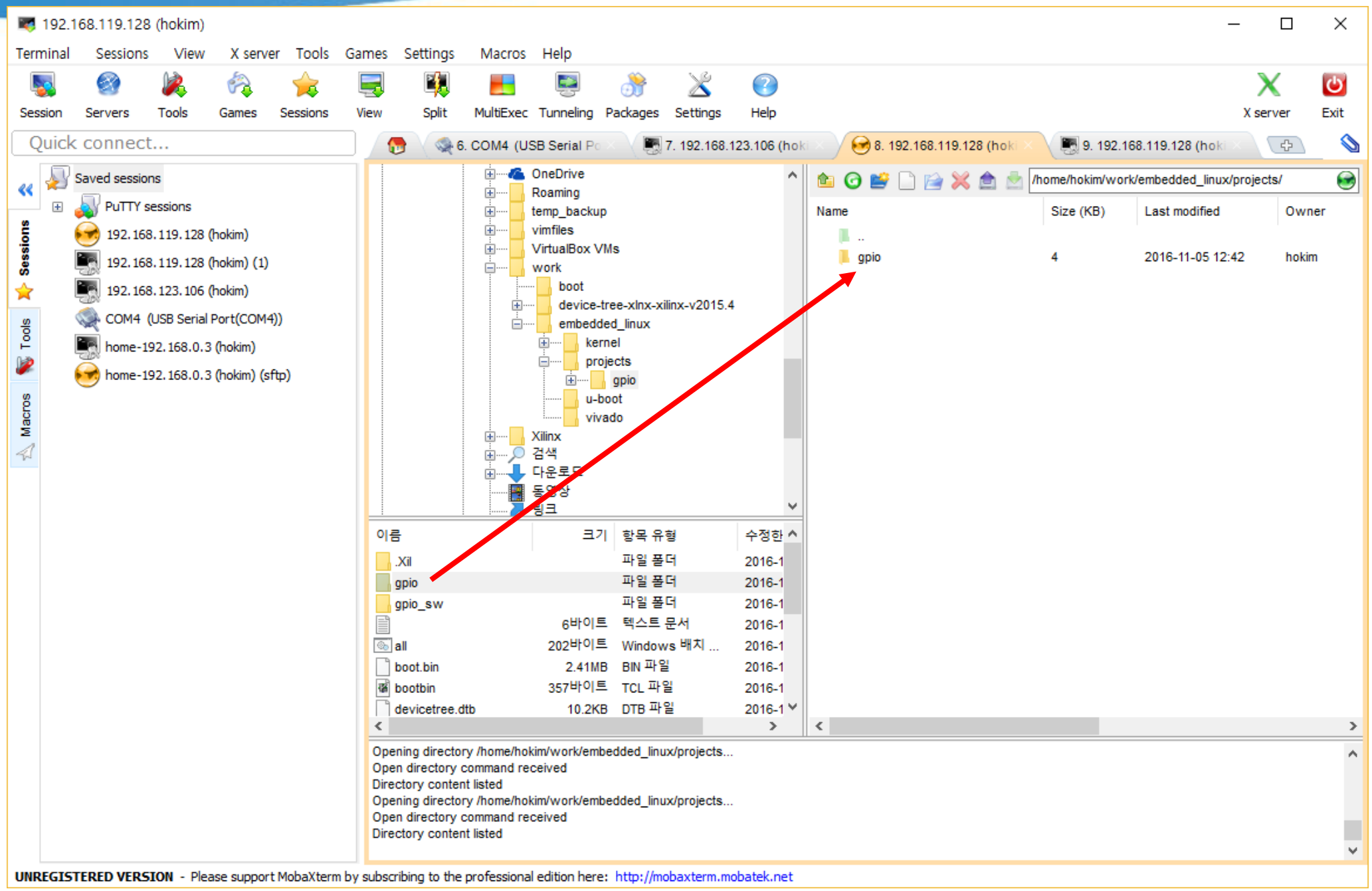
Boot.bin Build/DT Generation for GPIO

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/gpio
$ 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.dtsi"
/include/ "pl.dtsi“
…….
…….
&clkc {
    fclk-enable = <0x1>;
    ps-clk-frequency = <50000000>;
};
+&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

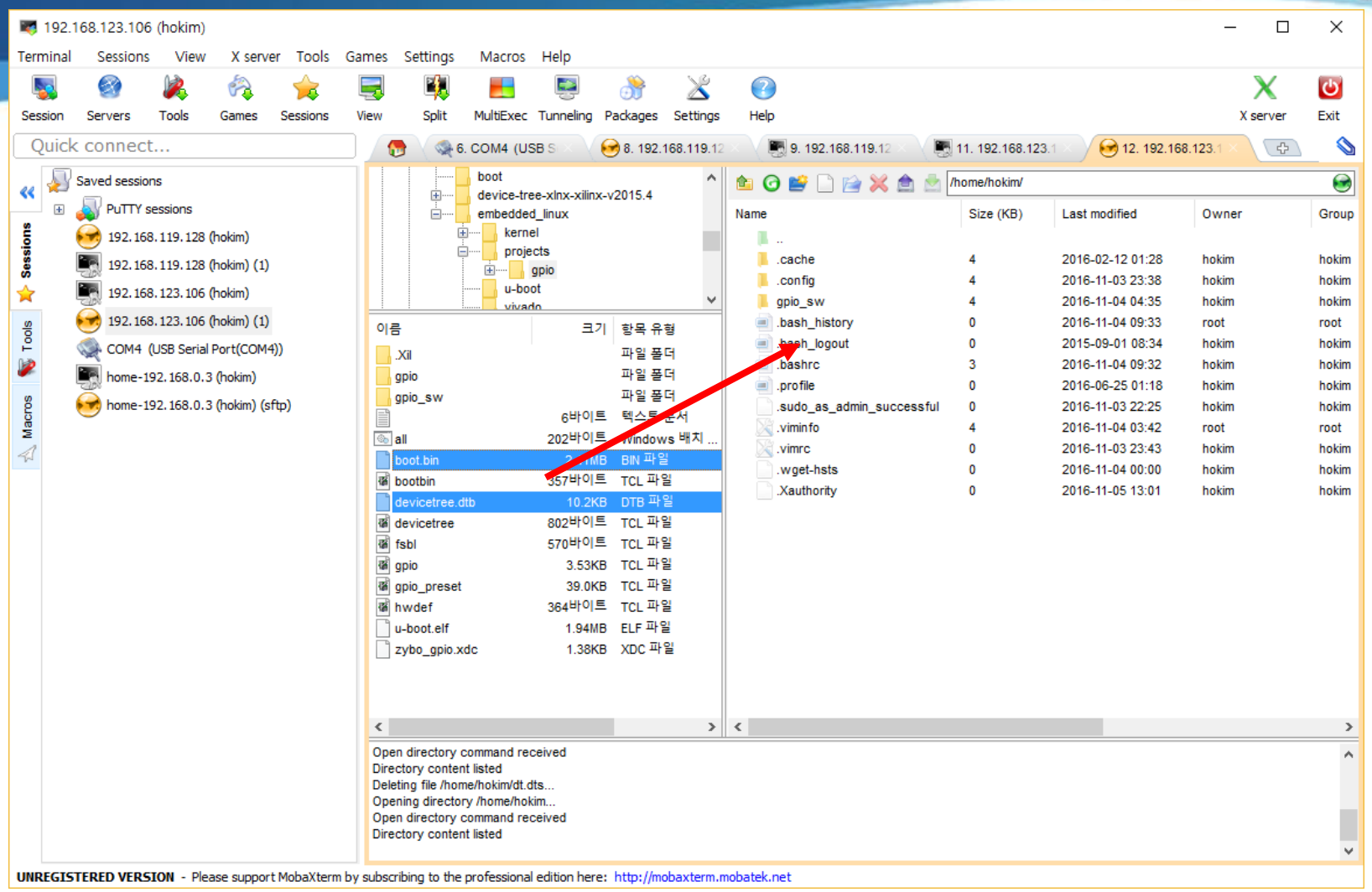
gpio/gpio.tree/*pl.dtsi*

```
/ {
        amba_pl: amba_pl {
                #address-cells = <1>;
                #size-cells = <1>;
                compatible = "simple-bus";
                ranges ;
                btns_4Bits: gpio@40001000 {
                        #gpio-cells = <2>;
                        compatible = "xlnx,xps-
gpio-1.00.a";
                        gpio-controller ;
                        interrupt-parent = <&intc>;
                        interrupts = <0 30 4>;
                        reg = <0x40001000
0x1000>;
.....
……
                };
                leds_4Bits: gpio@40000000 {
                        #gpio-cells = <2>;
                        compatible = "xlnx,xps-
gpio-1.00.a";
                        gpio-controller ;
                        interrupt-parent = <&intc>;
                        interrupts = <0 31 4>;
                        reg = <0x40000000
0x1000>;
.....
……
                };
```

```
                sws_4Bits: gpio@40002000 {
                        #gpio-cells = <2>;
                        compatible = "xlnx,xps-
gpio-1.00.a";
                        gpio-controller ;
                        interrupt-parent = <&intc>;
                        interrupts = <0 29 4>;
                        reg = <0x40002000
0x1000>;
.....
……
                };
        };
};
```

```
$ dtc -O dtb -I 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
```

GPIO investigation

```
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
```

GPIO sw & test

```
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
```

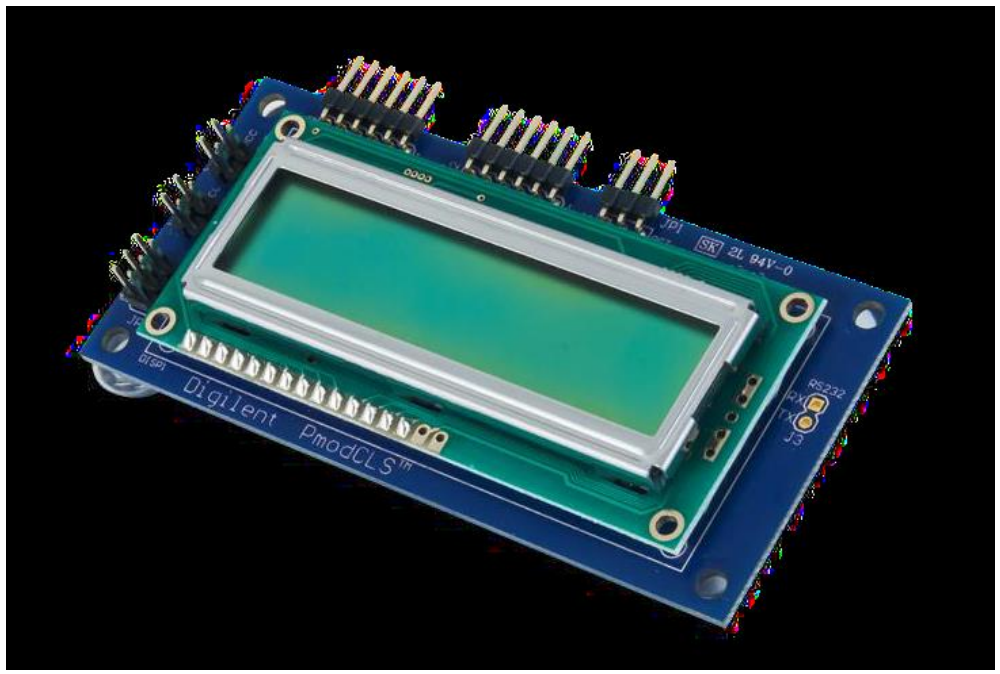
```
hokim@zybo:~$ cd gpio_sw/driver/python
hokim@zybo:~/gpio_sw/driver/python$ ls
gpio.py  gpio_test.py
hokim@zybo:~/gpio_sw/driver/python$ sudo python gpio_test.py
```

```
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
```

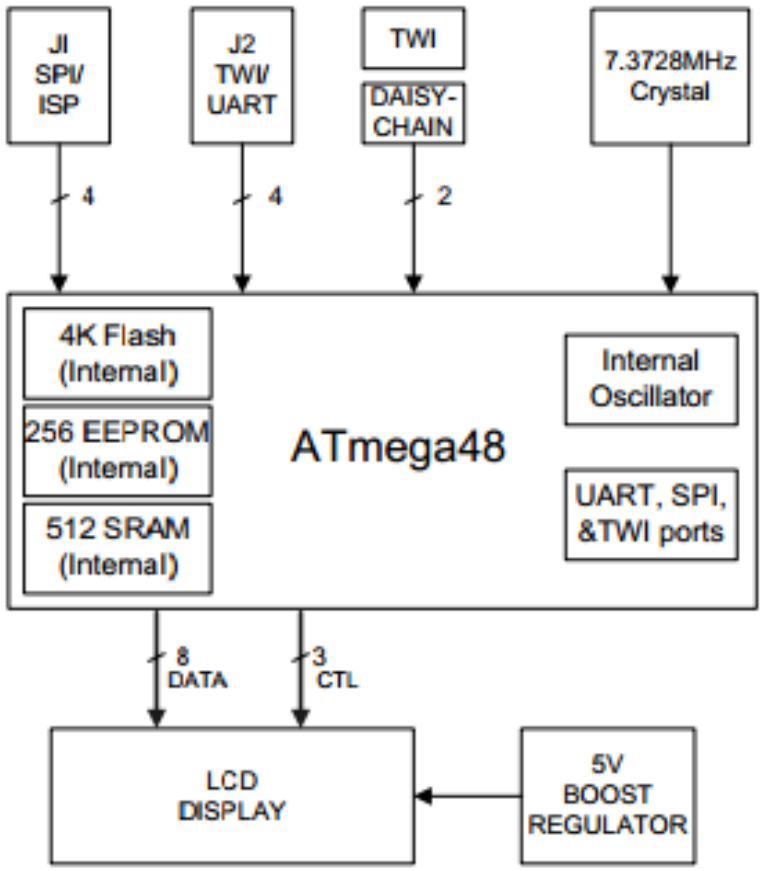
PmodCLS

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

https://github.com/Xilinx/linux-xlnx/blob/master/Documentation/pmods/pmodcls.txt



PmodCLS



MD2, MD1, MD0	Protocol	Details
0,0,0	UART	2400 baud
0,0,1	UART	4800 baud
0,1,0	UART	9600 baud
0,1,1	UART	Baud rate in EEPROM
1,0,0	TWI	Address: 0x48
1,0,1	TWI	Address in EEPROM
1,1,0	SPI	
1,1,1	Specified in EEPROM	Specified in EEPROM

Header J1		
Pin	**Signal**	**Description**
1	SS	Slave Select
2	MOSI	Master-Out-Slave-In
3	MISO	Master-In-Slave-Out
4	SCK	Serial Clock
5	GND	Power Supply Ground
6	VCC	Positive Power Supply (3.3V)

Header J2		
Pin	**Signal**	**Description**
1	SCL	Serial Clock
2	SDA	Serial Data
3	TXD	Transmit Data
4	RXD	Receive Data
5	GND	Power Supply Ground
6	VCC	Positive Power Supply (3.3V)

Linux SPI

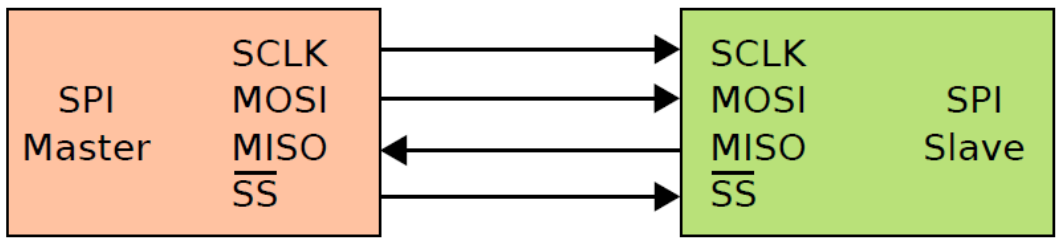
- **Synchronous serial digital data link by Motorola**
- **SPI master controls**

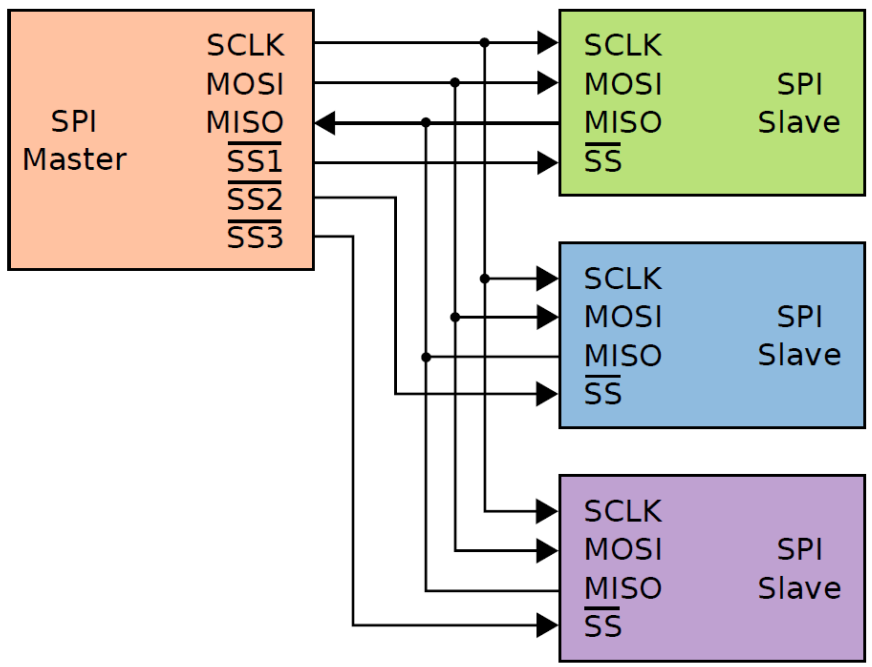
CLK : synchronization clock

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

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

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





Linux SPI Userspace Interface

INIPRO

- Documented in Documentation/spi/spidev
- SPI kernel framework
- SPI master drivers : drivers/spi/(spi-cadence.c, spi-zynq-qspi.c)
- Kernel platform code registers the “spidev” platform device
- Create character device nodes at /dev/spidevB.C where:

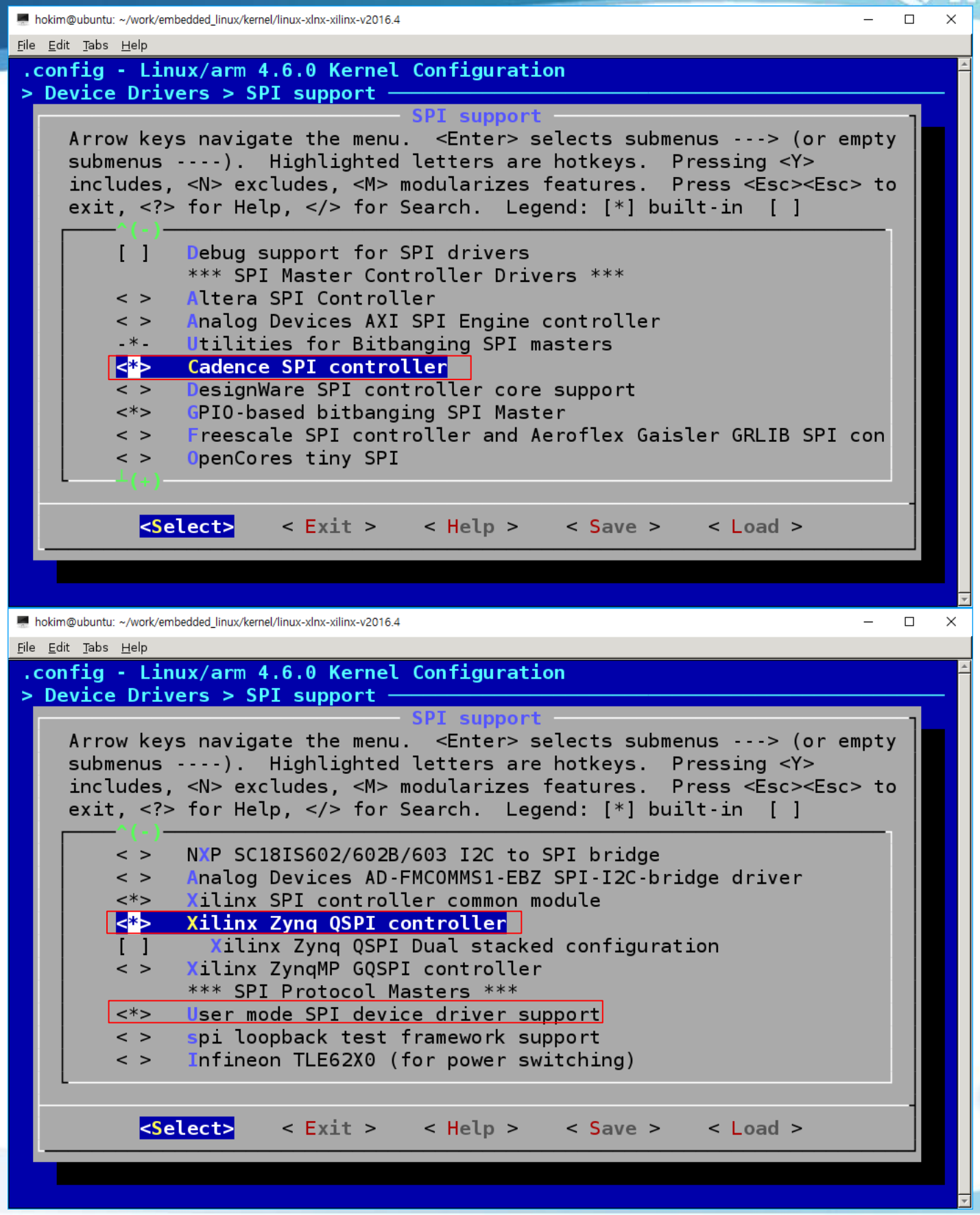
B is the SPI bus (master) number

C is the chip-select number of specific SPI slave.

Linux SPI Userspace Interface

INIPRO

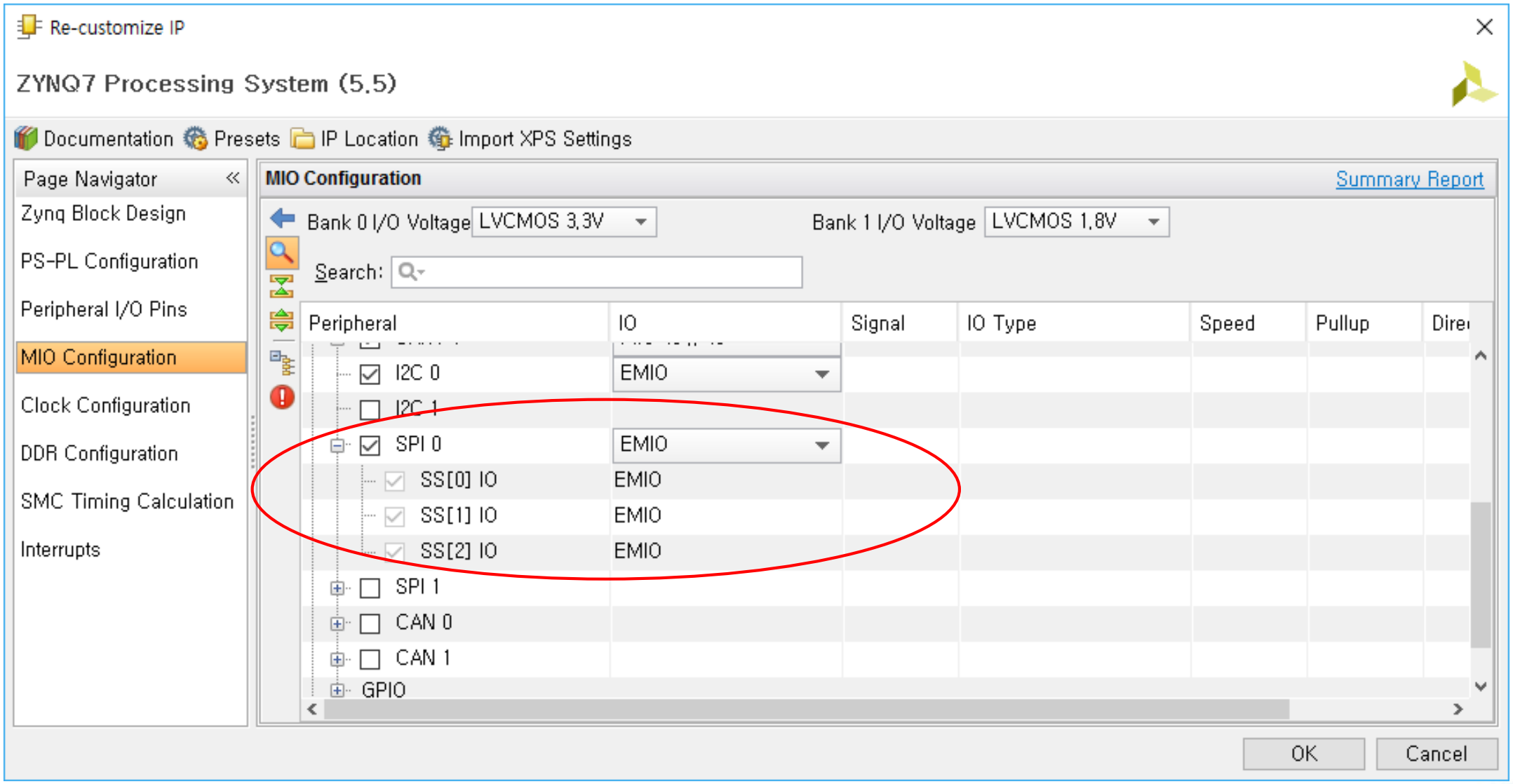
Kernel Configuration



Zynq7 PS Re-customize IP for SPI

INIPRO

Based on Zynq7 PS Re-customize IP of embedded_linux project



Zynq7 PS Re-customize IP for SPI

INIPRO

Based on Zynq7 PS Re-customize IP of embedded_linux project

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

PS-PL Configuration — Summary Report

Search:

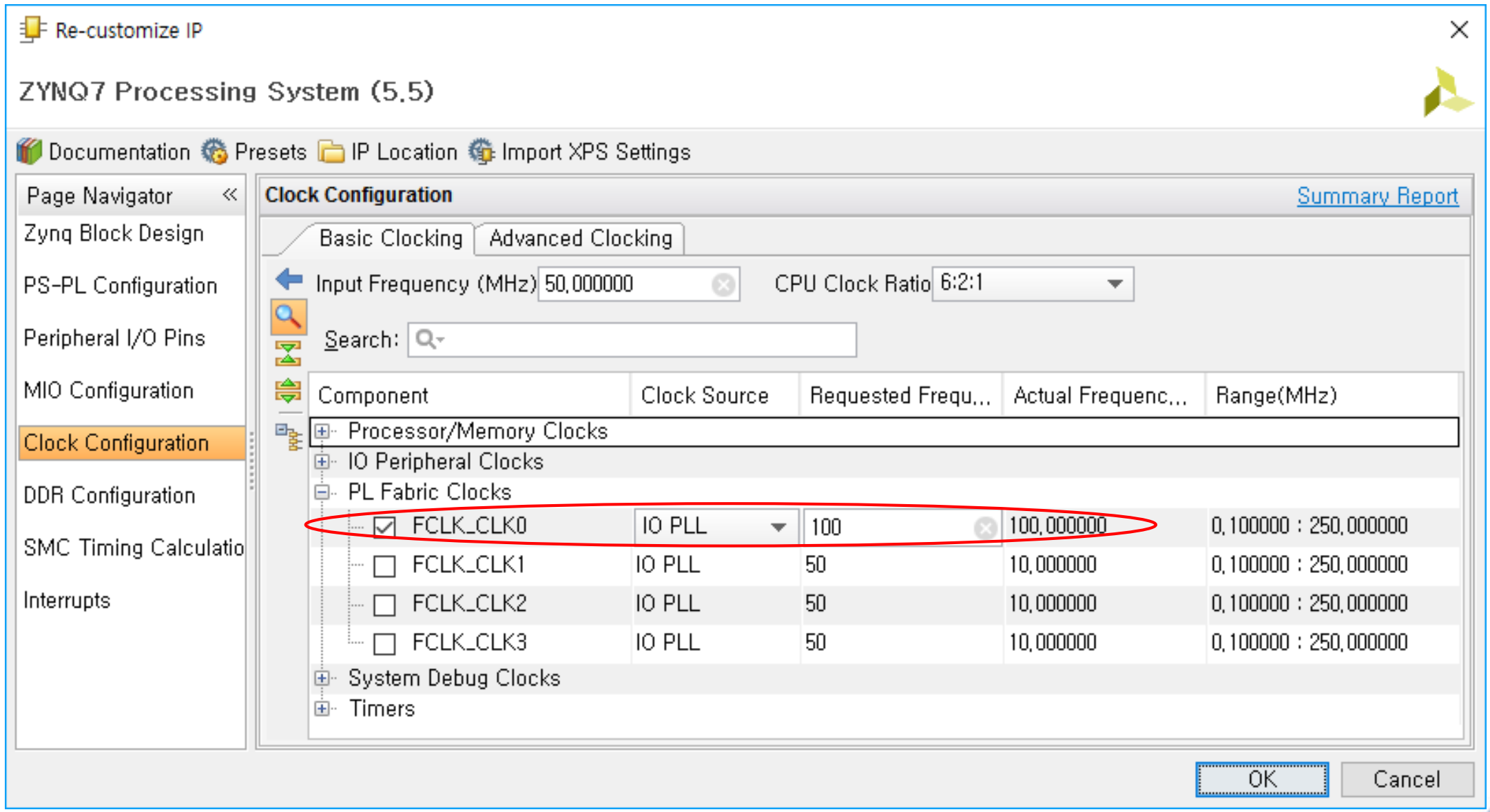
Name	Select	Description
General		
UART0 Baud Rate	115200	Baud rate is generated with internally fixed UART Ref Clock Freq=10...
UART1 Baud Rate	115200	Baud rate is generated with internally fixed UART Ref Clock Freq=10...
PL AXI idle Port	☐	Enables idle AXI signal to the PS used to indicate that there are no ...
DDR ARB bypass Port	☐	Enables DDR urgent/arb signal used to signal a critical memory sta...
PS-PL Debug interface	☐	Enables PL debug signals to PS and vice-versa
FTM Trace data interface	☐	Enables FTM Trace AXI stream interface used to capture data from PL to PS debug system
FTM Trace buffer	0	
FTM Data edge detector	0	Stores trace data in the FIFO when the data changes as marked by ...
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 be...
Include ACP transaction checker	☐	Enables ACP transaction checker.
Trace data/control signal pipeline width	8	Enables configurable number of pipeline stages on the TRACE DAT...
Power-on reset(POR) 4k timer	☐	Enables power-on reset(POR) 4k timer. By default, 64k timer is used.
Processor event interface	☐	Enables event bus which provides a low-latency and direct mecha...
Address Editor		
Enable Clock Triggers		
Enable Clock Resets		
FCLK_RESET0_N	☑	Enables general purpose reset signal 0 for PL logic
FCLK_RESET1_N	☐	Enables general purpose reset signal 1 for PL logic
FCLK_RESET2_N	☐	Enables general purpose reset signal 2 for PL logic
FCLK_RESET3_N	☐	Enables general purpose reset signal 3 for PL logic
AXI Non Secure Enablement	0	Enable AXI Non Secure Transaction
GP Master AXI Interface		
M AXI GP0 interface	☑	Enables General purpose AXI master interface 0
M AXI GP1 interface	☐	Enables General purpose AXI master interface 1
GP Slave AXI Interface		

OK | Cancel

Zynq7 PS Re-customize IP for SPI

INIPRO

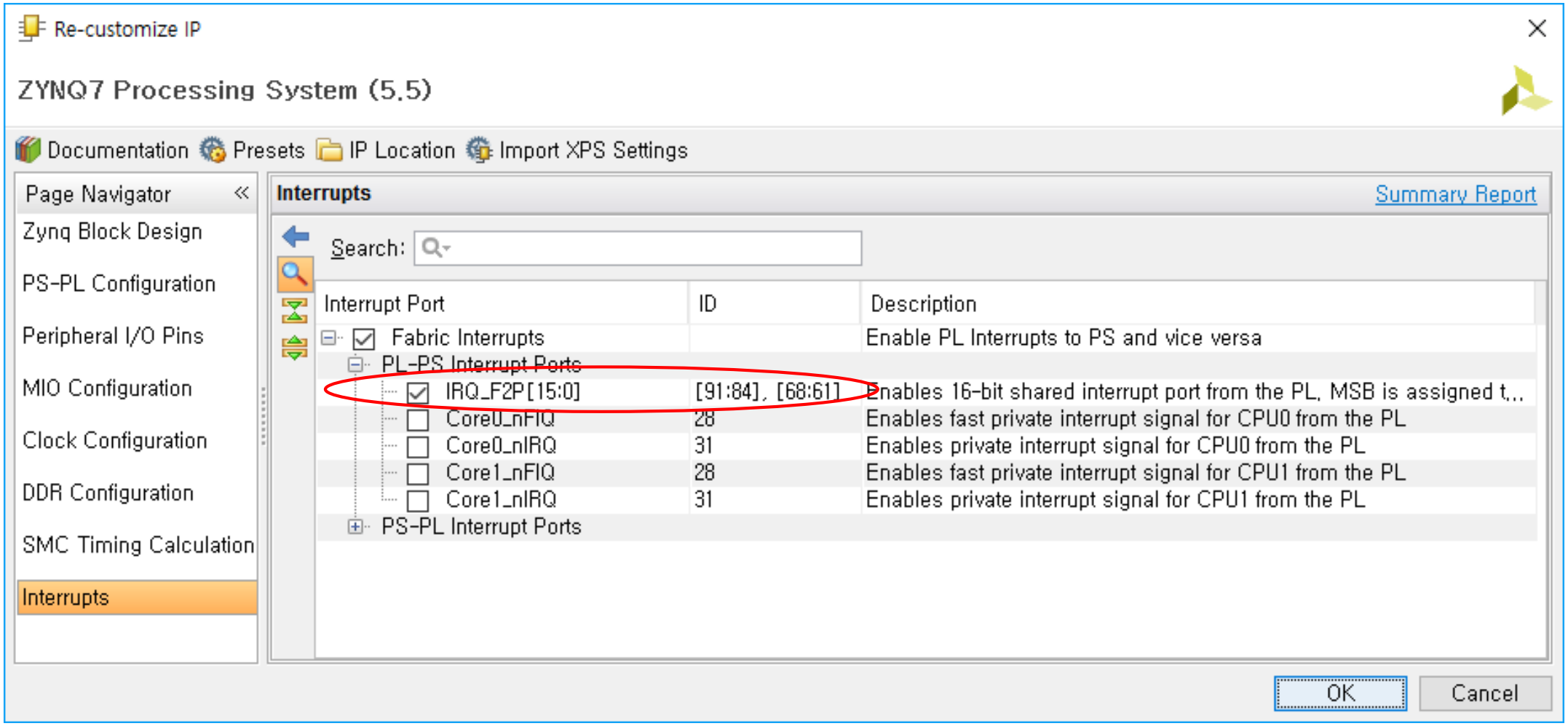
Based on Zynq7 PS Re-customize IP of embedded_linux project



Zynq7 PS Re-customize IP for SPI

INIPRO

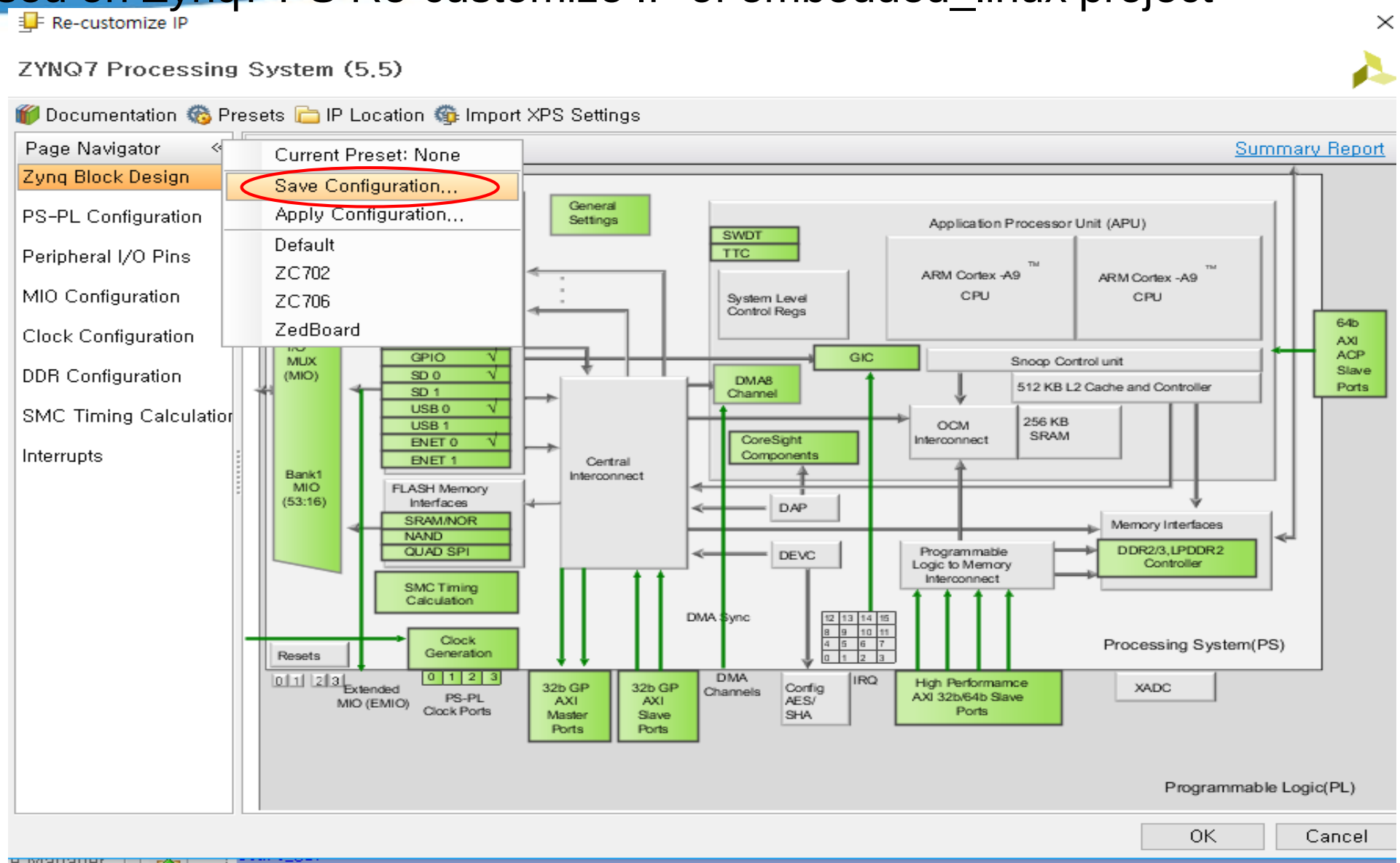
Based on Zynq7 PS Re-customize IP of embedded_linux project

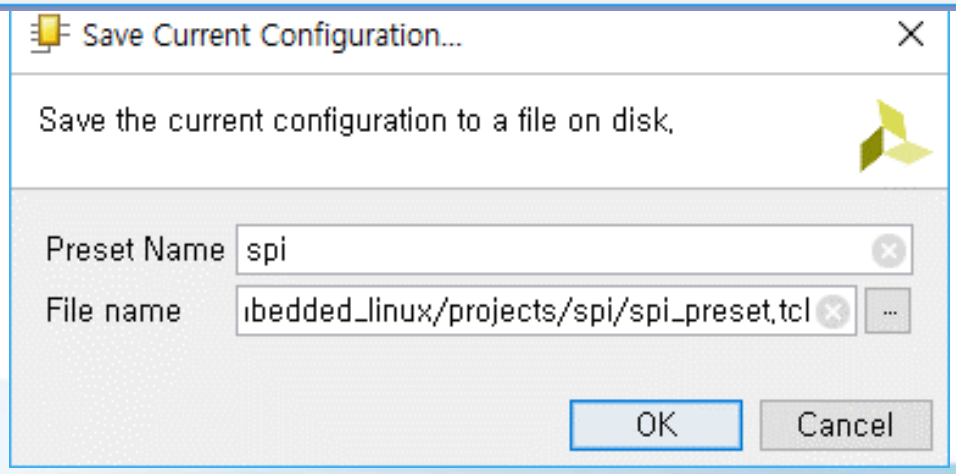


Zynq7 PS Re-customize IP for SPI

INIPRO

Based on Zynq7 PS Re-customize IP of embedded_linux project





Project for SPI

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

In cmd window for Vivado

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

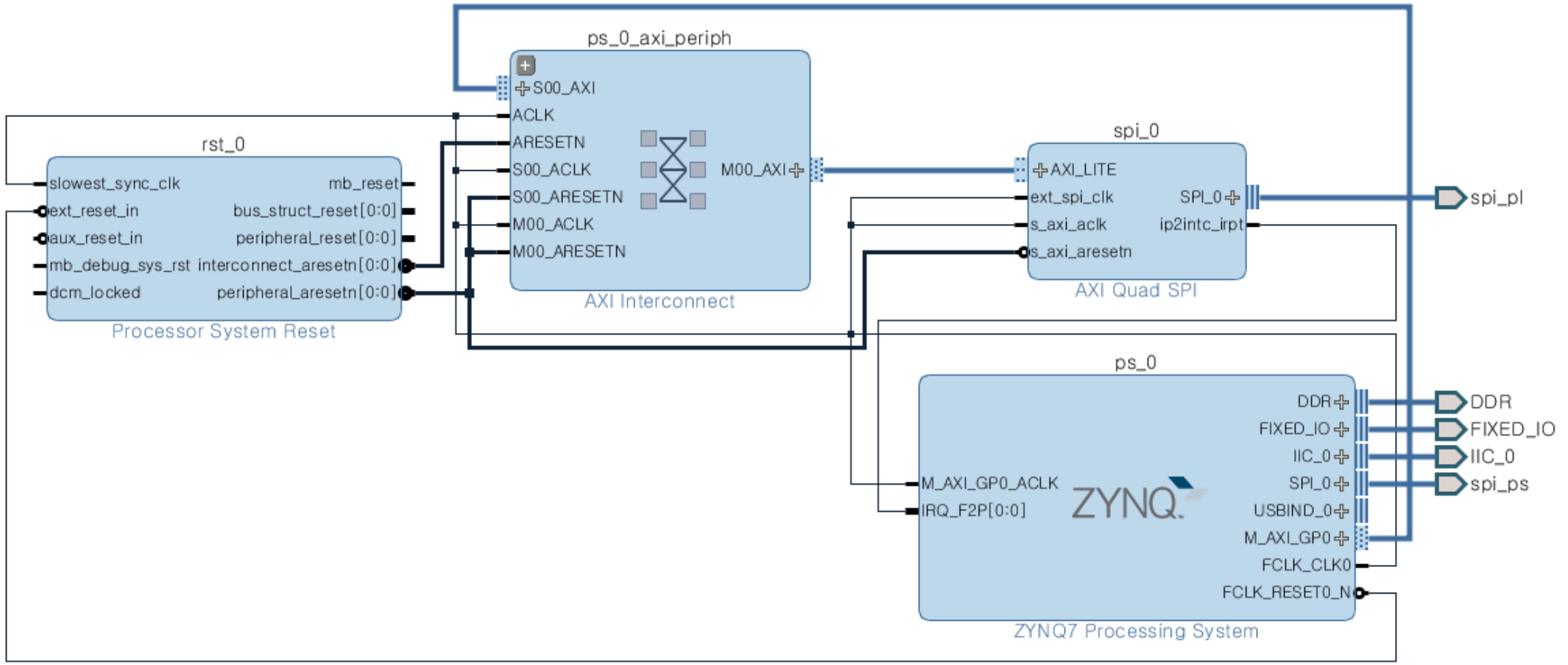
output : spi\spi.xpr…

Project for SPI

INIPRO

Diagram

system



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

Bit Generation for SPI

INIPRO

spi - [C:/Users/hokim/work/embedded_linux/projects/spi/spi/spi.xpr] - Vivado 2016.4

File Edit Flow Tools Window Layout View Help

Quick Access

Default Layout

Ready

Flow Navigator

- Language Templates
- IP Catalog

IP Integrator

- Create Block Design
- Open Block Design
- Generate Block Design

Simulation

- Simulation Settings
- Run Simulation

RTL Analysis

- Elaboration Settings
- Open Elaborated Design

Synthesis

- Synthesis Settings
- Run Synthesis
- Open Synthesized Desig

Implementation

- Implementation Settings
- Run Implementation
- Open Implemented Desig

Program and Debug

- Bitstream Settings
- Generate Bitstream
- Open Hardware Manager

Block Design - system

Design

- system
  - External Interfaces
  - Interface Connections
  - Nets
  - ps_0 (ZYNQ7 Processing S
  - ps_0_axi_periph
  - rst_0 (Processor System Re
  - spi_0 (AXI Quad SPI:3,2)

Sou.. De.. Sig..

Block Propert...

ps_0

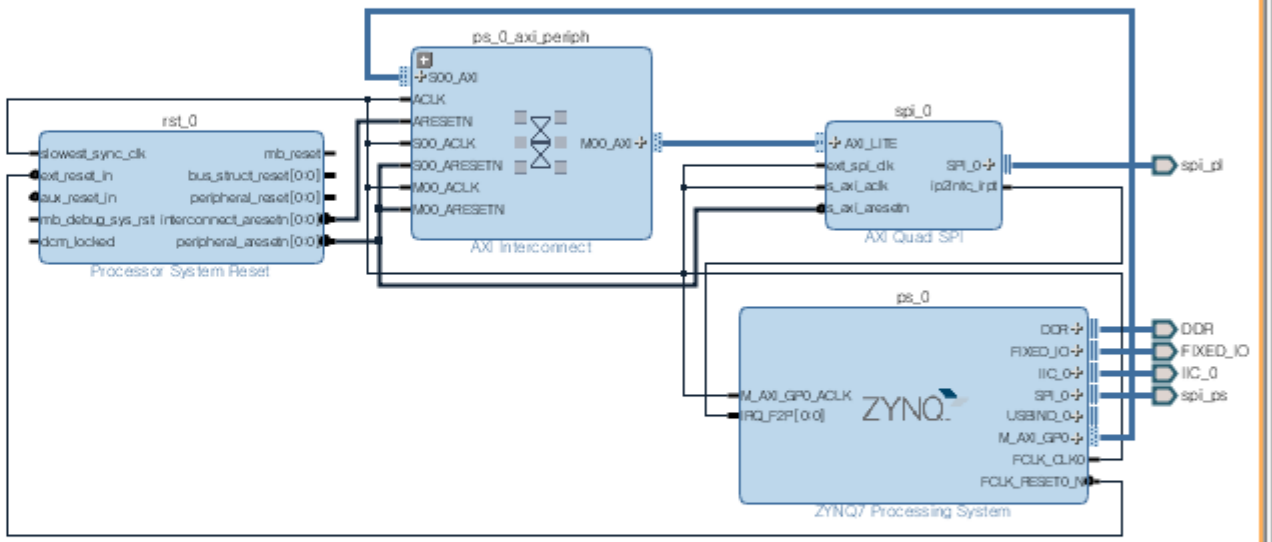
Name: ps_0

Parent name: system

General Properties IP

Diagram | Address Editor

system



Tcl Console

```
open_project: Time (s): cpu = 00:00:22 ; elapsed = 00:04:50 . Memory (MB): peak = 718.695 ; gain = 97.902
open_bd_design {C:/Users/hokim/work/embedded_linux/projects/spi/spi/spi.srcs/sources_1/bd/system/system.bd}
Adding cell -- xilinx.com:ip:processing_system7:5.5 - ps_0
```

Type a Tcl command here

Tcl Console | Messages | Log | Reports | Design Runs

Generate a programming file after implementation

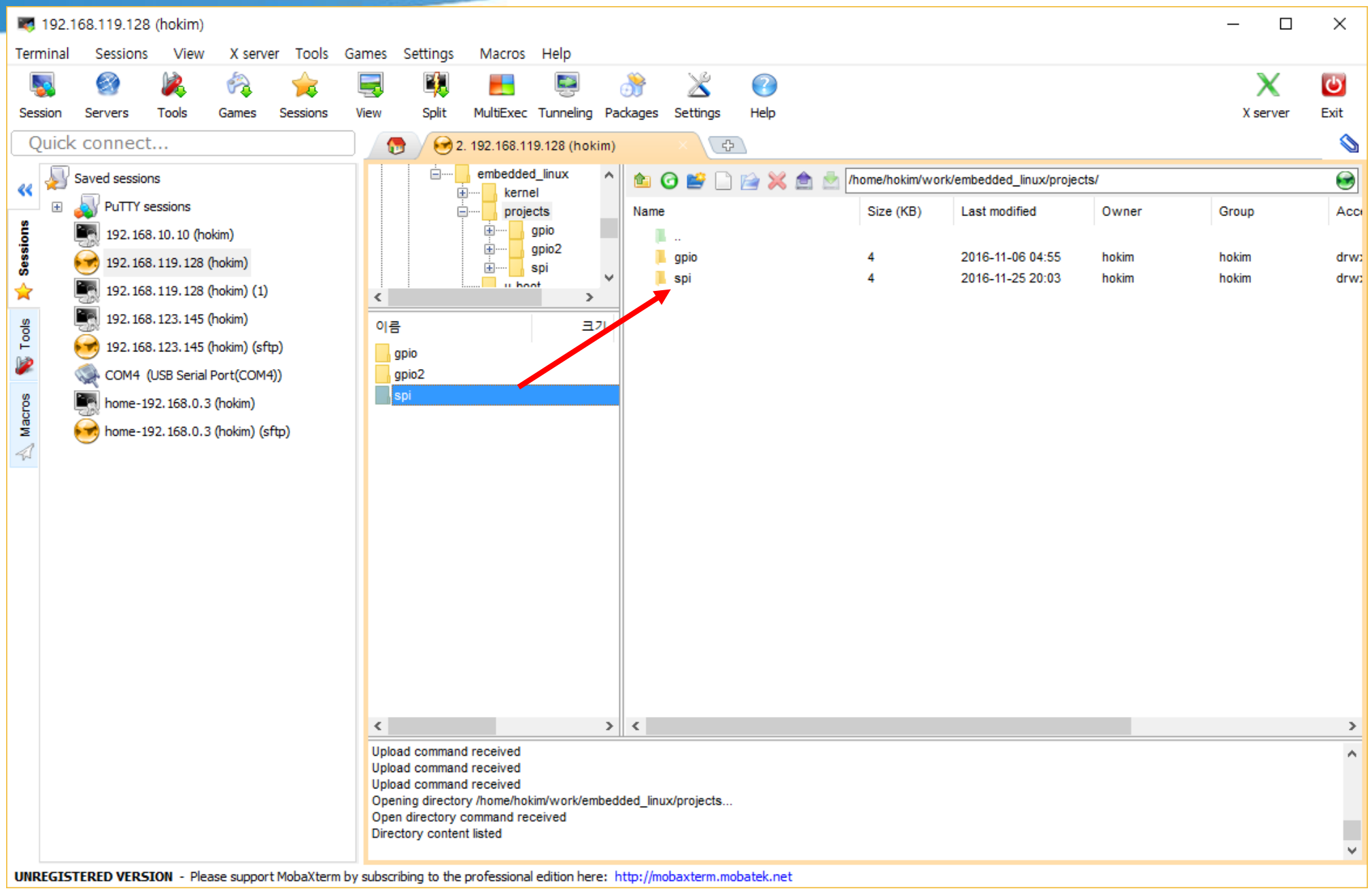
Boot.bin Build/DT Generation for SPI

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/spi
$ 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>;
};

+&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>;
+    };
+};
```

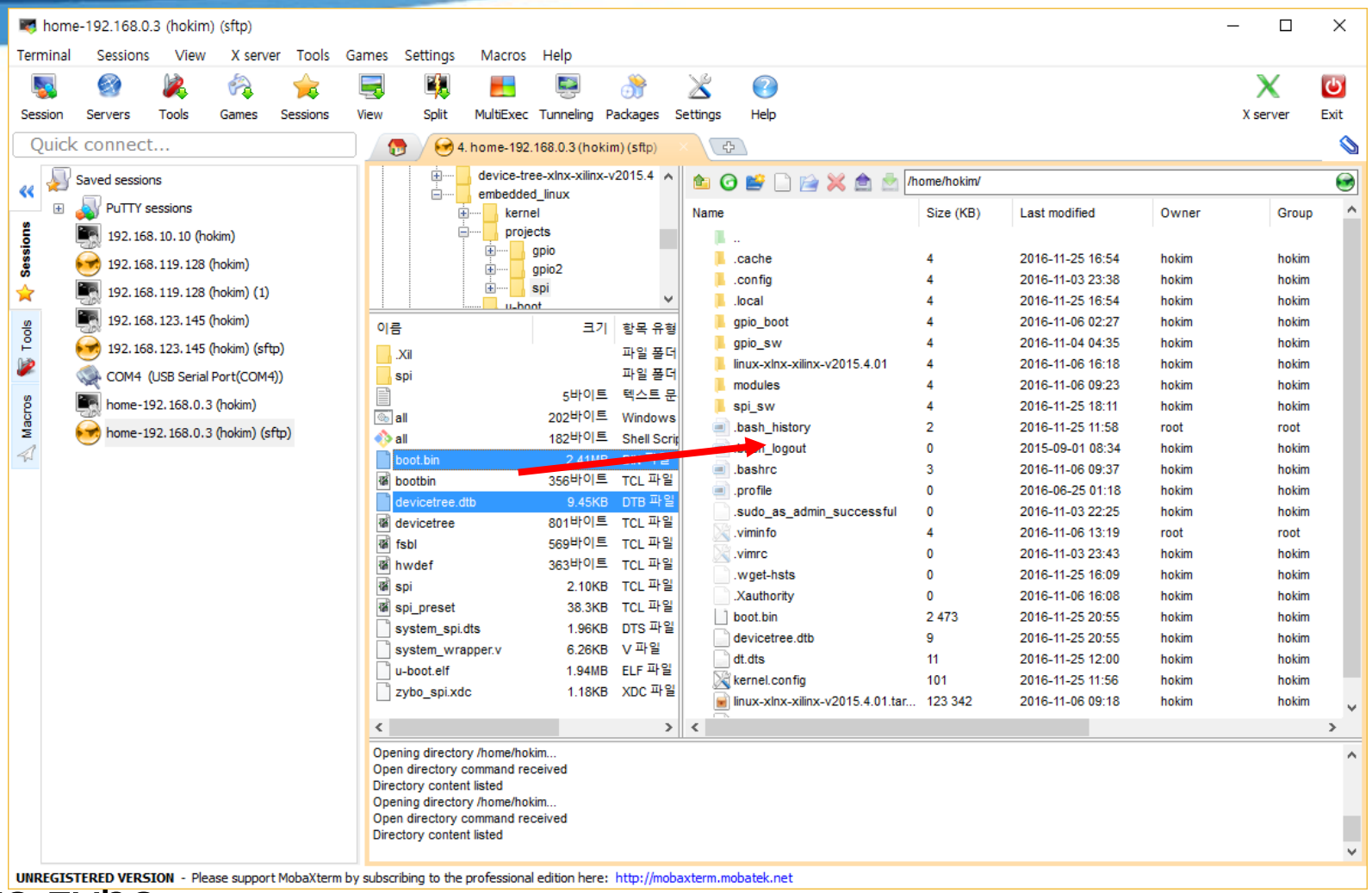
Device Tree Compile for SPI

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 -I 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
```

SPI sw & test

```
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
```

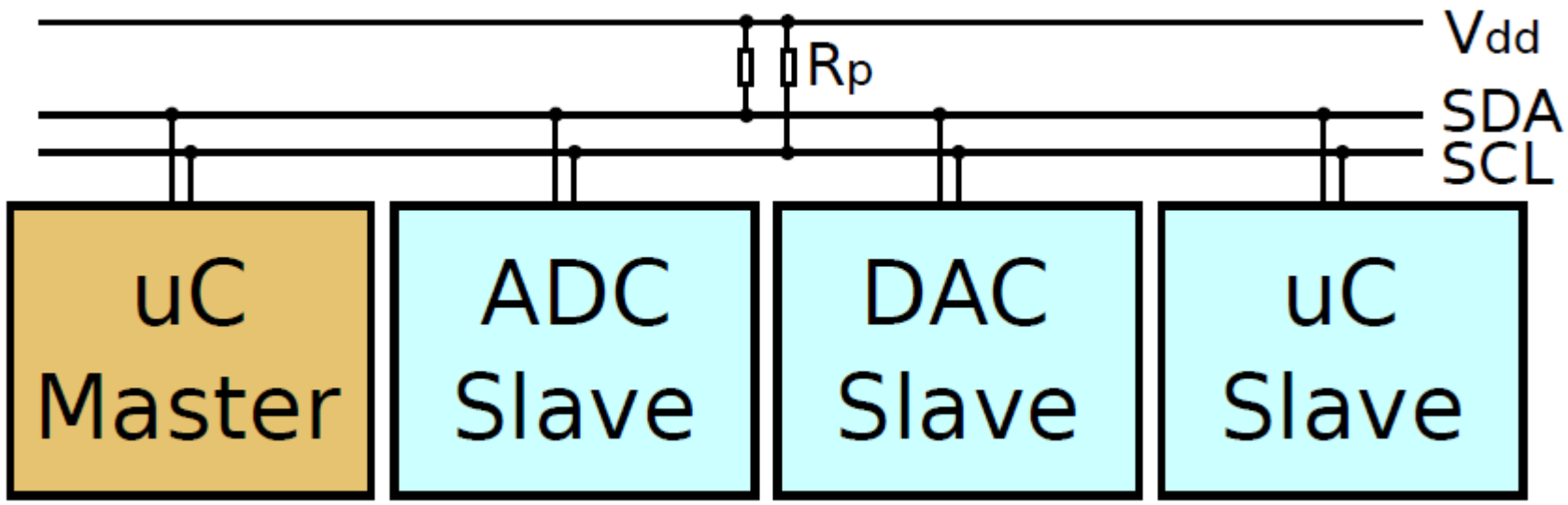
```
hokim@zybo:~$ cd spi_sw/python
hokim@zybo:~/spi_sw/python$ ls
pmodcls_spi.py  spi_test.py
hokim@zybo:~/spi_sw/python$ sudo apt install python-pip
hokim@zybo:~/spi_sw/python$ pip install spidev
hokim@zybo:~/spi_sw/python$ sudo python spi_test.py
```

I2C

- **Two wires are controlled by the master and slaves according to the protocol**

  SCL : Serial CLock

  SDA : Serial DAta
- **Each slave has a 7 bit address**
- **The 7 MSB of the first transmitted byte are slave address**
- **The LSB of this byte indicates read (1), or write (0)**
- **SMBus is a subset of I2C, with a stricter protocol definition**



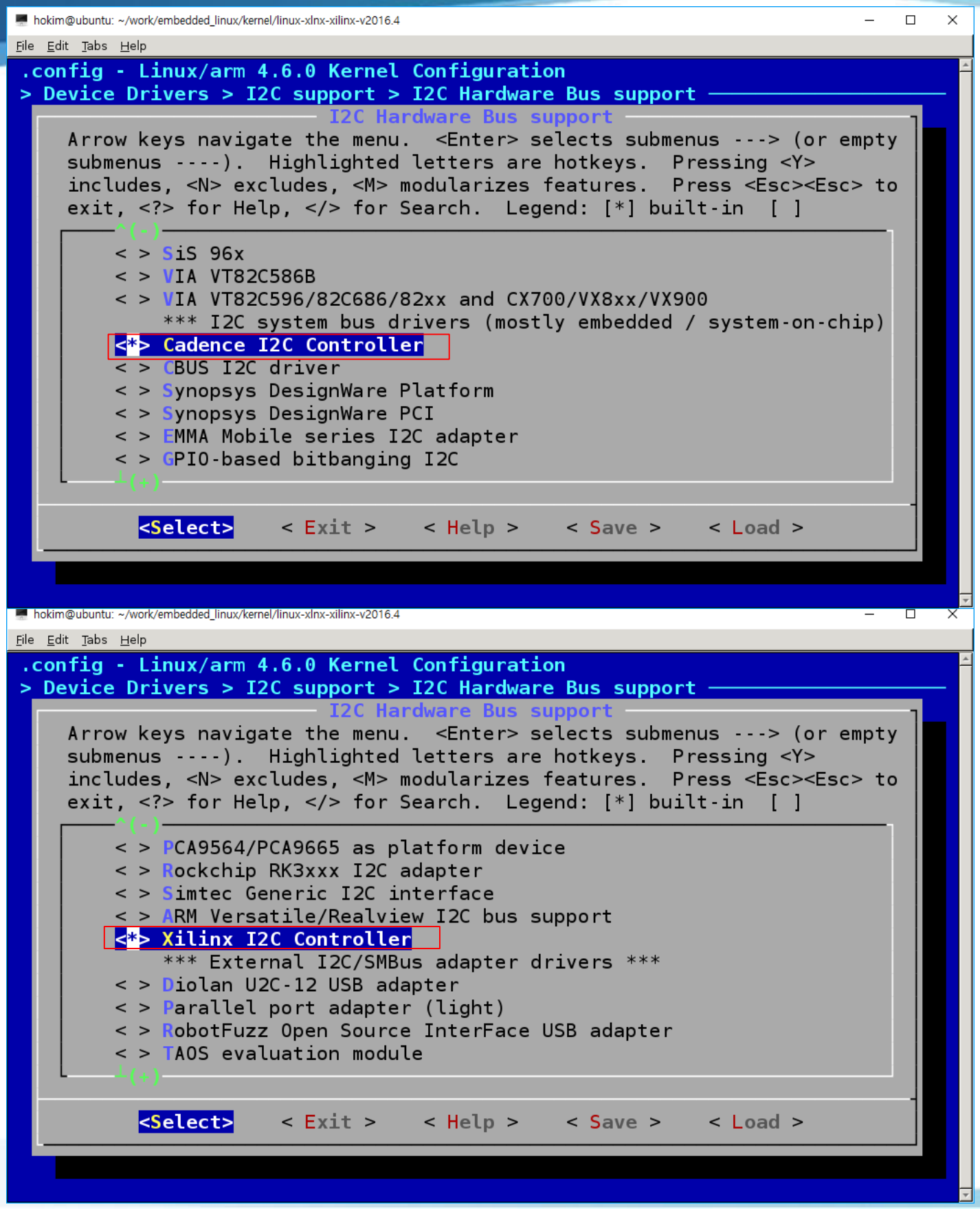
Linux I2C Userspace Interface

INIPRO

- Documented in Documentation/i2c/dev-interfaces
- I2C framework
- I2C master driver : drivers/i2c/busses/(i2c-cadence.c, i2c-xiic.c)
- Each master gets a character device node at /dev/i2c-N, where N is the ID master number

Linux I2C Userspace Interface

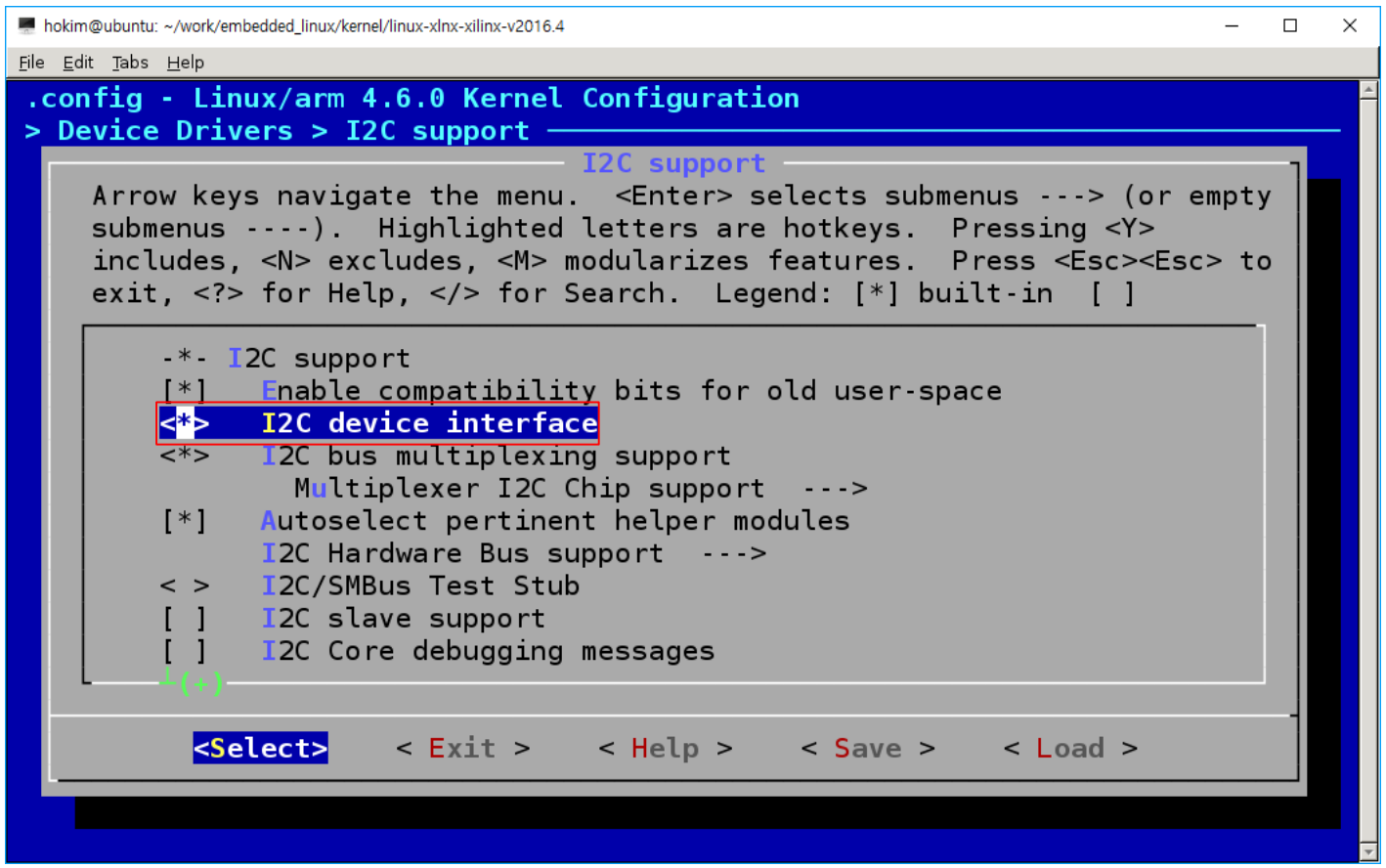
Kernel Configuration



Linux I2C Userspace Interface

INIPRO

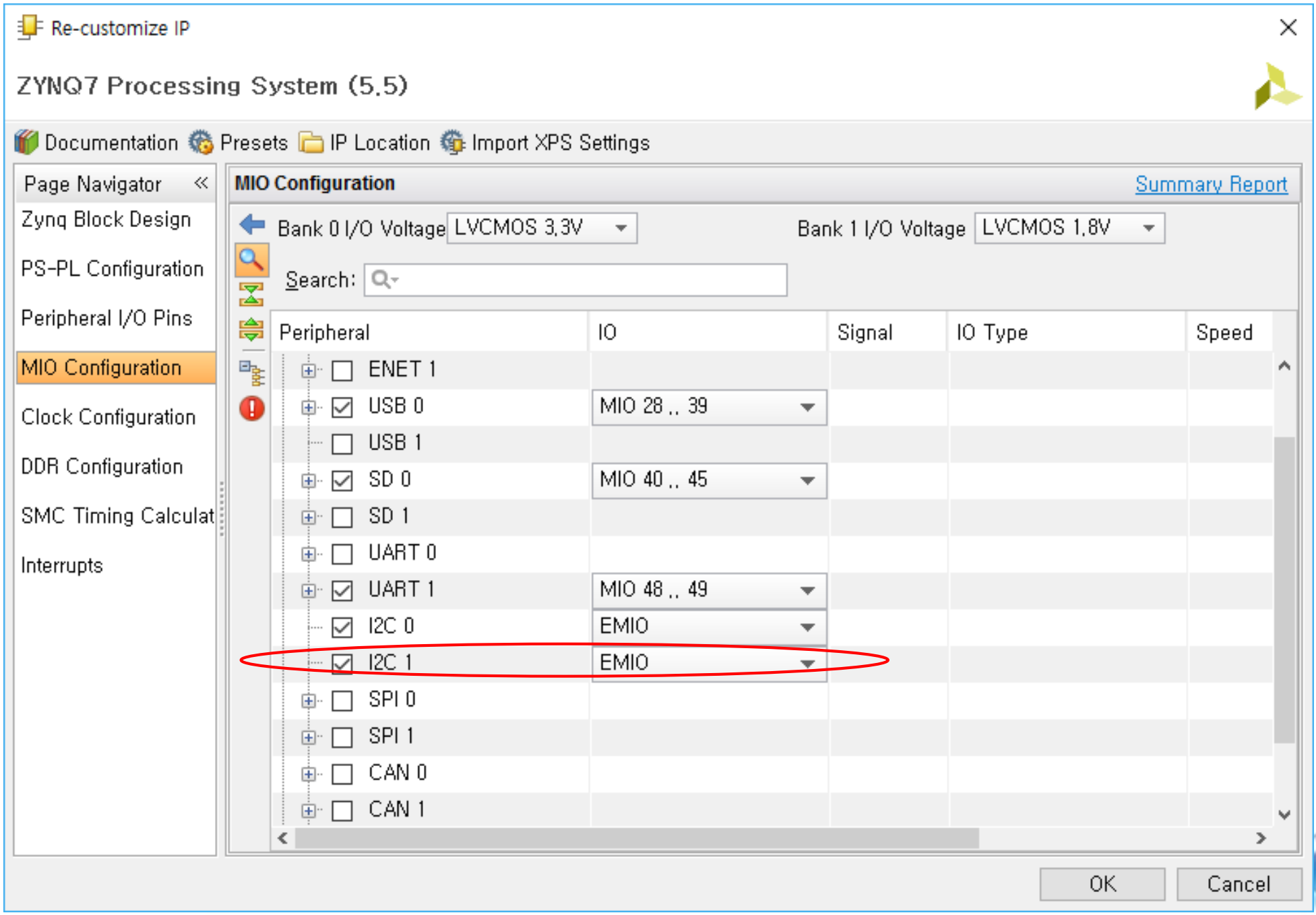
Kernel Configuration



Zynq7 PS Re-customize IP for I2C

INIPRO

Based on Zynq7 PS Re-customize IP of embedded_linux project



Zynq7 PS Re-customize IP for I2C

INIPRO

Based on Zynq7 PS Re-customize IP of embedded_linux project

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

PS-PL Configuration — Summary Report

Search:

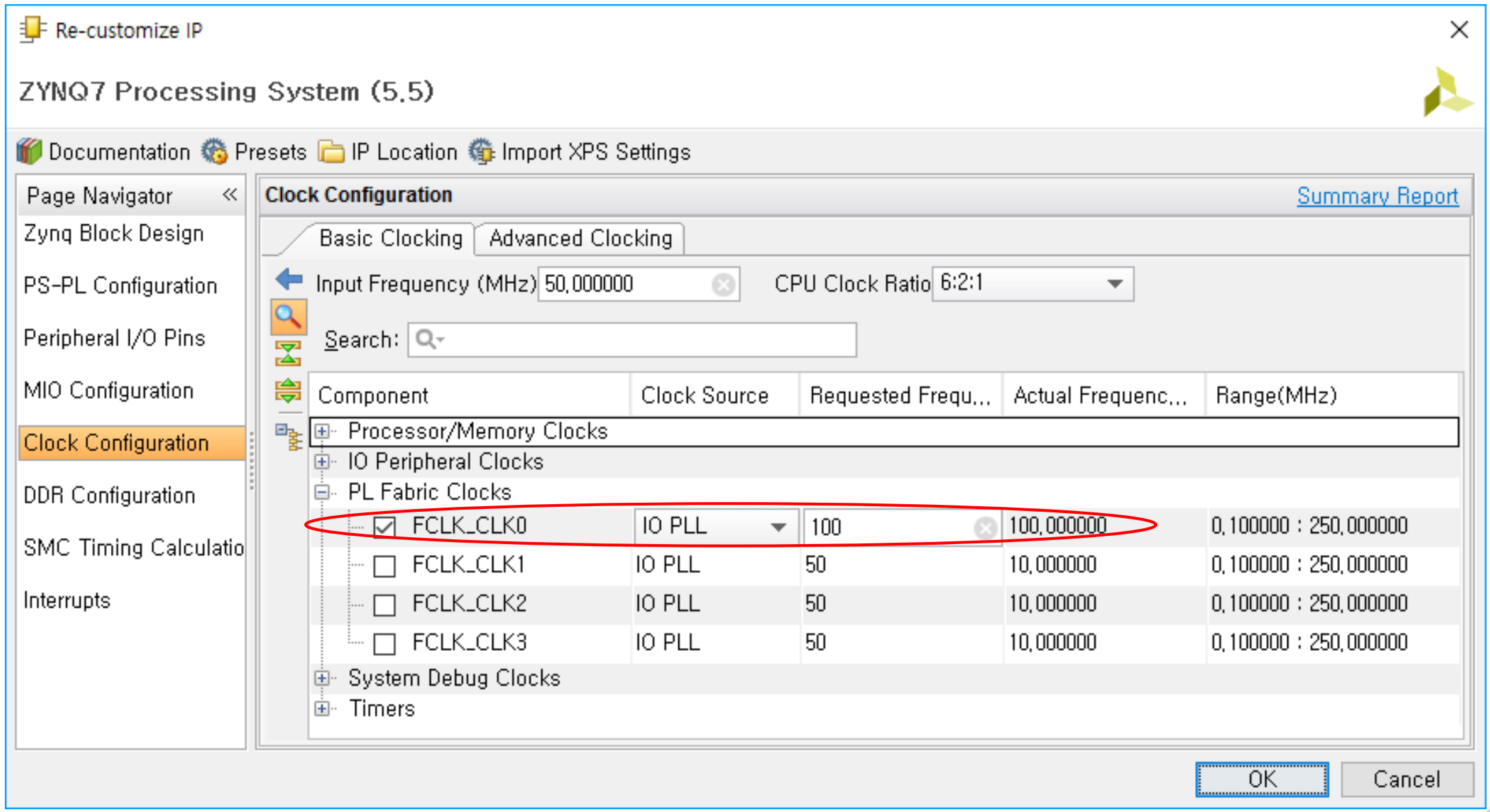
Name	Select	Description
General		
UART0 Baud Rate	115200	Baud rate is generated with internally fixed UART Ref Clock Freq=10...
UART1 Baud Rate	115200	Baud rate is generated with internally fixed UART Ref Clock Freq=10...
PL AXI idle Port	☐	Enables idle AXI signal to the PS used to indicate that there are no ...
DDR ARB bypass Port	☐	Enables DDR urgent/arb signal used to signal a critical memory sta...
PS-PL Debug interface	☐	Enables PL debug signals to PS and vice-versa
FTM Trace data interface	☐	Enables FTM Trace AXI stream interface used to capture data from PL to PS debug system
FTM Trace buffer	0	
FTM Data edge detector	0	Stores trace data in the FIFO when the data changes as marked by ...
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 be...
Include ACP transaction checker	☐	Enables ACP transaction checker.
Trace data/control signal pipeline width	8	Enables configurable number of pipeline stages on the TRACE DAT...
Power-on reset(POR) 4k timer	☐	Enables power-on reset(POR) 4k timer. By default, 64k timer is used.
Processor event interface	☐	Enables event bus which provides a low-latency and direct mecha...
Address Editor		
Enable Clock Triggers		
Enable Clock Resets		
FCLK_RESET0_N	☑	Enables general purpose reset signal 0 for PL logic
FCLK_RESET1_N	☐	Enables general purpose reset signal 1 for PL logic
FCLK_RESET2_N	☐	Enables general purpose reset signal 2 for PL logic
FCLK_RESET3_N	☐	Enables general purpose reset signal 3 for PL logic
AXI Non Secure Enablement	0	Enable AXI Non Secure Transaction
GP Master AXI Interface		
M AXI GP0 interface	☑	Enables General purpose AXI master interface 0
M AXI GP1 interface	☐	Enables General purpose AXI master interface 1
GP Slave AXI Interface		

OK | Cancel

Zynq7 PS Re-customize IP for I2C

INIPRO

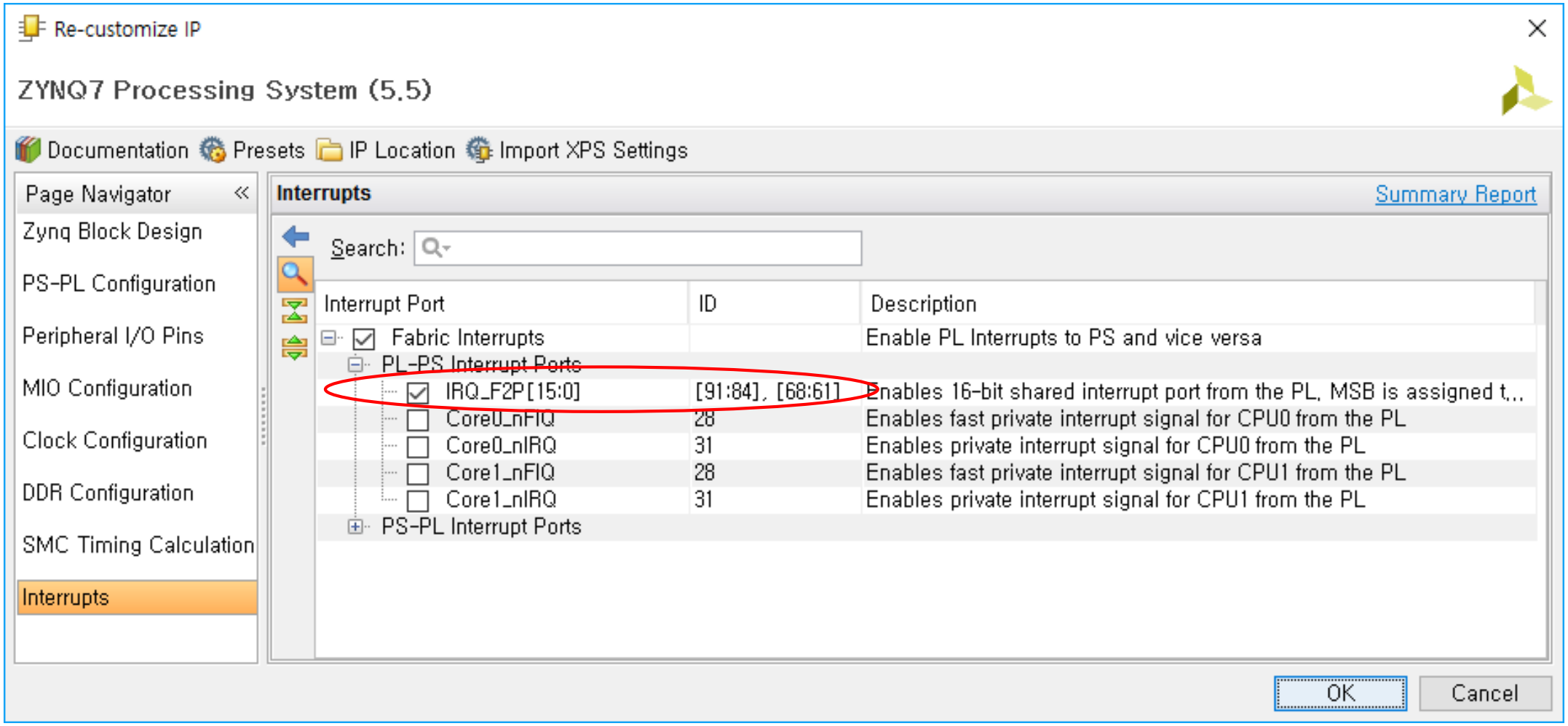
Based on Zynq7 PS Re-customize IP of embedded_linux project



Zynq7 PS Re-customize IP for I2C

INIPRO

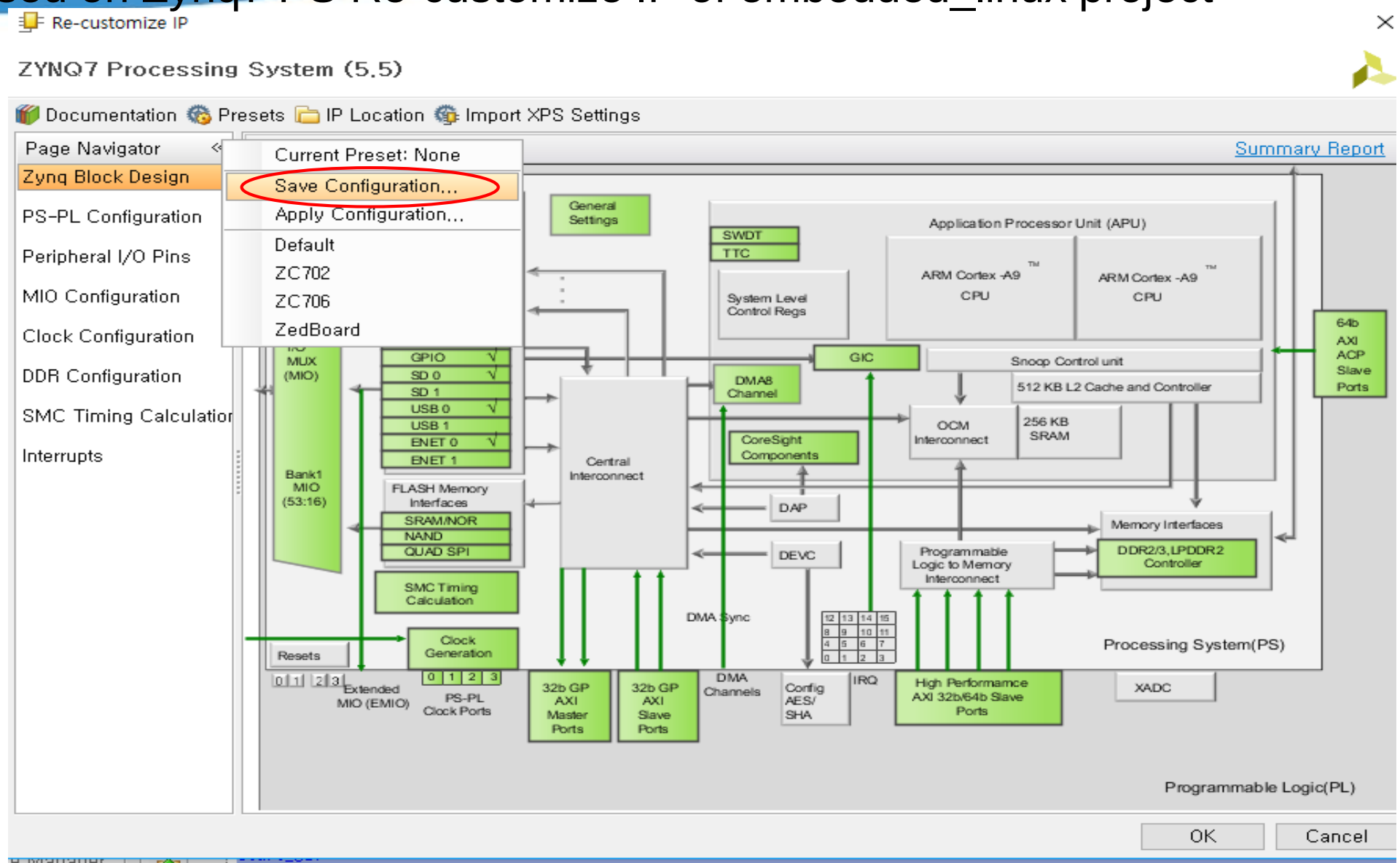
Based on Zynq7 PS Re-customize IP of embedded_linux project

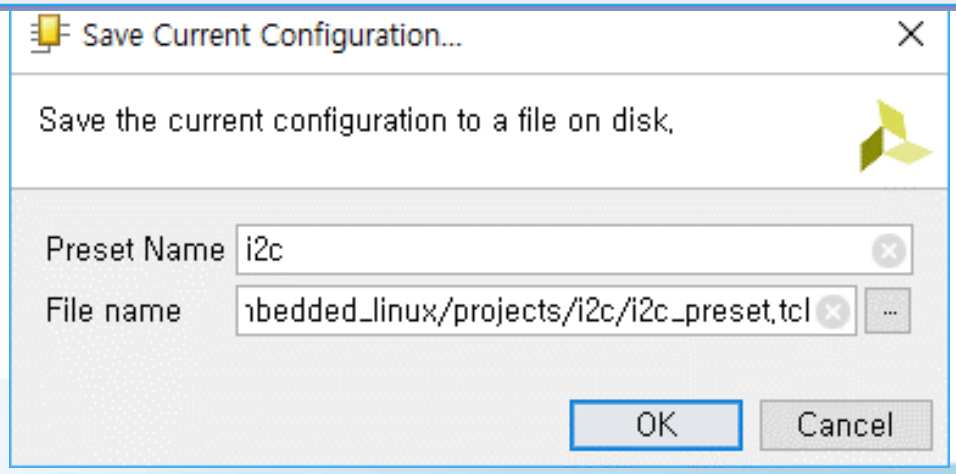


Zynq7 PS Re-customize IP for I2C

INIPRO

Based on Zynq7 PS Re-customize IP of embedded_linux project





Project for I2C

https://github.com/inipro/embedded_linux/projects/i2c

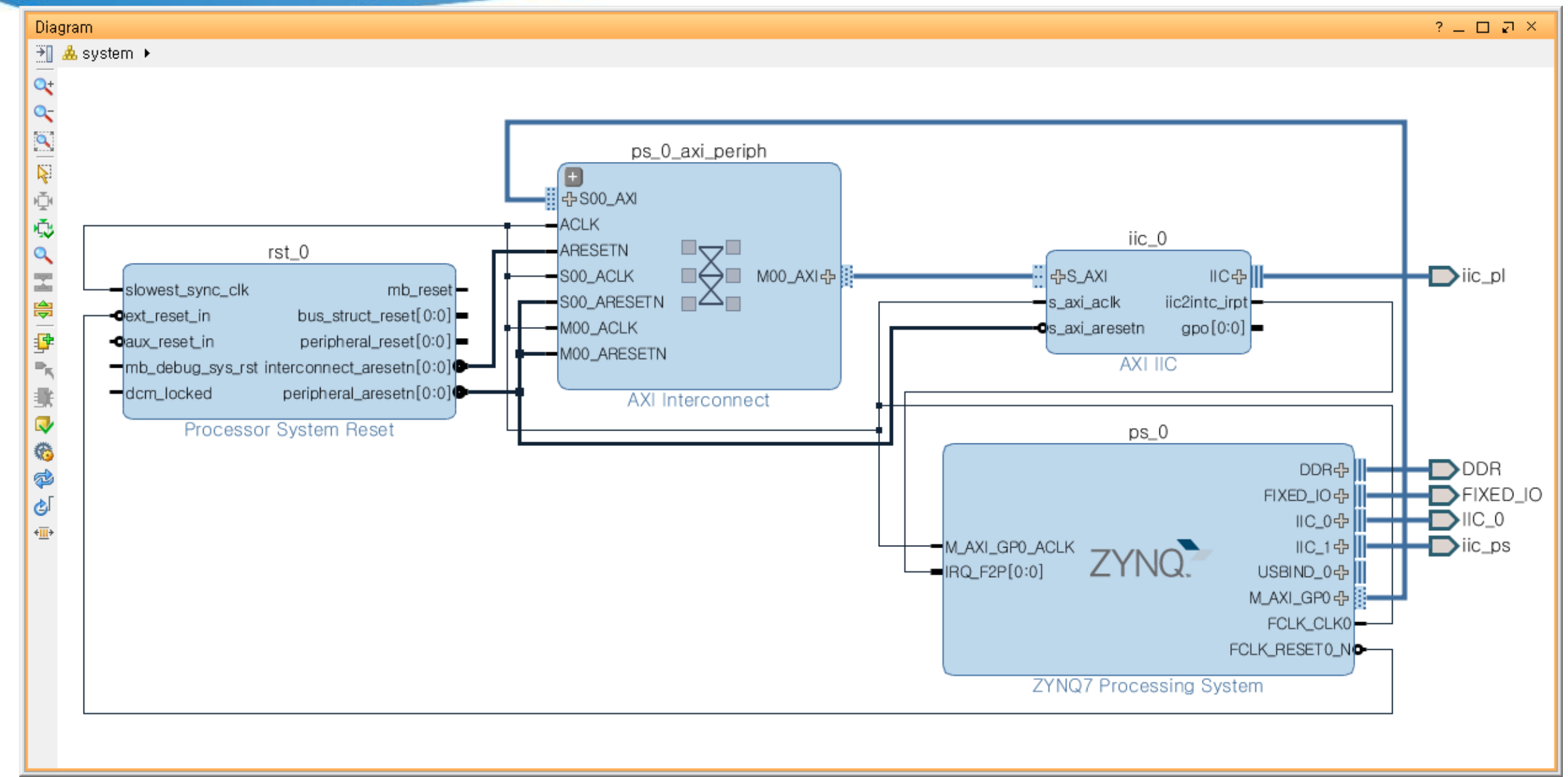
In cmd window for Vivado

```
C:\> cd C:\Users\hokim\work\embedded_linux\projects\i2c
C:\> vivado  -nolog -nojournal -mode batch -source i2c.tcl
```

output : i2c\i2c.xpr…

Project for I2C

INIPRO



Address Editor

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
ps_0					
Data (32 address bits : 0x40000000 [1G])					
iic_0	S_AXI	Reg	0x4000_0000	4K	0x4000_0FFF

Bit Generation for I2C

INIPRO

i2c - [C:/Users/hokim/work/embedded_linux/projects/i2c/i2c/i2c.xpr] - Vivado 2016.4

File Edit Flow Tools Window Layout View Help

Quick Access

Default Layout

Ready

Flow Navigator

- IP Integrator
  - Create Block Design
  - Open Block Design
  - Generate Block Design
- Simulation
  - Simulation Settings
  - Run Simulation
- RTL Analysis
  - Elaboration Settings
  - Open Elaborated Design
- Synthesis
  - Synthesis Settings
  - Run Synthesis
  - Open Synthesized Desig
- Implementation
  - Implementation Settings
  - Run Implementation
  - Open Implemented Desi
- Program and Debug
  - Bitstream Settings
  - Generate Bitstream
  - Open Hardware Manage

Block Design - system

Design

- system
  - External Interfaces
  - Interface Connections
  - Nets
  - iic_0 (AXI IIC:2.0)
  - ps_0 (ZYNQ7 Processing System
  - ps_0_axi_periph
  - rst_0 (Processor System Reset:5

Sour.. Design Signals

Block Properties

ps_0

Name: ps_0

Parent name: system

General Properties IP

Diagram Address Editor

system

ps_0_axi_periph

rst_0

Processor System Reset

AXI Interconnect

iic_0

AXI IIC

ps_0

ZYNQ7 Processing System

Tcl Console

```
open_project: Time (s): cpu = 00:00:14 ; elapsed = 00:00:19 . Memory (MB): peak = 728.027 ; gain = 147.734
open_bd_design {C:/Users/hokim/work/embedded_linux/projects/i2c/i2c/i2c.srcs/sources_1/bd/system/system.bd}
Adding cell -- xilinx.com:ip:processing_system7:5.5 - ps_0
Adding cell -- xilinx.com:ip:proc_sys_reset:5.0 - rst_0
```

Type a Tcl command here

Tcl Console Messages Log Reports Design Runs

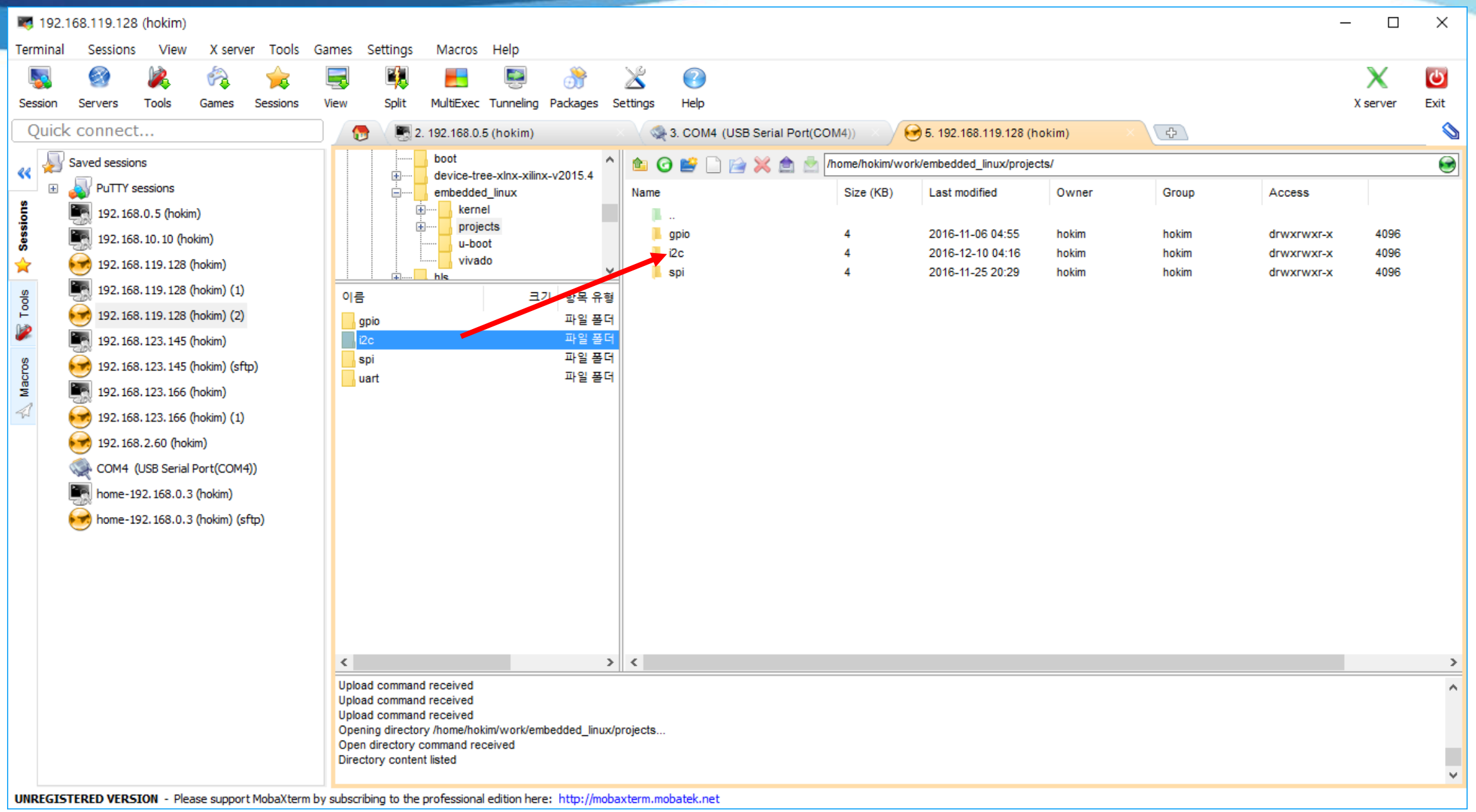
Boot.bin Build/DT Generation for I2C

C:\Users\hokim\work\embedded_linux\projects\i2c\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\i2c
C:\> all
```

Device Tree Compile for I2C




```
$ cd ~/work/embedded_linux/projects/i2c
$ cp i2c/i2c.tree/system.dts  system_i2c.dts
$ nano system_i2c.dts
```

Device Tree Compile for I2C

system_i2c.dts

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

+};
+&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 I2C

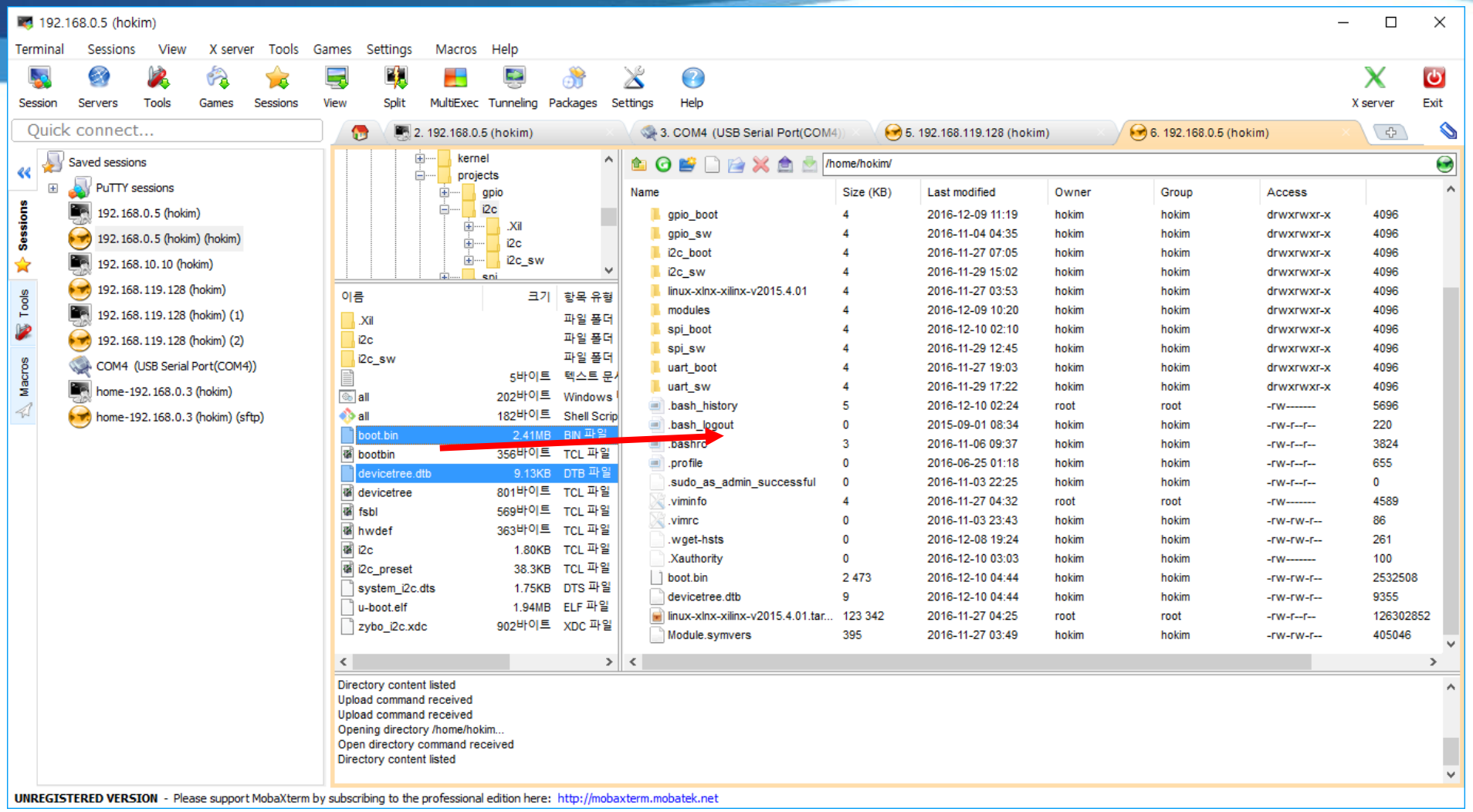
INIPRO

i2c/i2c.tree/*pl.dtsi*

```
/ {
    amba_pl: amba_pl {
        #address-cells = <1>;
        #size-cells = <1>;
        compatible = "simple-bus";
        ranges ;
        iic_0: i2c@40000000 {
            #address-cells = <1>;
            #size-cells = <0>;
            compatible = "xlnx,xps-iic-2.00.a";
            interrupt-parent = <&intc>;
            interrupts = <0 29 4>;
            reg = <0x40000000 0x1000>;
        };
    };
};
```

```
$ dtc -O dtb -I dts -i i2c/i2c.tree/ -o devicetree.dtb system_i2c.dts
```

Update boot.bin, devicetree.dtb for I2C



login into zybo

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

I2C sw & test

```
hokim@zybo:~$ ls /dev/i2c-*
/dev/i2c-0  /dev/i2c-1 /dev/i2c-2
hokim@zybo:~$ sudo i2cdetect -l
i2c-0   i2c             Cadence I2C at e0004000                 I2C adapter
i2c-1   i2c             Cadence I2C at e0005000                 I2C adapter
i2c-2   i2c             xiic-i2c                                I2C adapter
hokim@zybo:~$ sudo i2cdetect -y -r 0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- 1a -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: UU -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
hokim@zybo:~$ sudo i2cdetect -y -r 2
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- 48 -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

```
hokim@zybo:~$ cd i2c_sw/
hokim@zybo:~/i2c_sw$ ls
CMakeLists.txt  PmodCLS.c  PmodCLS.h  main.c  i2c.c  i2c.h
hokim@zybo:~/i2c_sw$ mkdir build
hokim@zybo:~/i2c_sw/$ cd build
hokim@zybo:~/i2c_sw/build$ cmake ..
hokim@zybo:~/i2c_sw/build$ make
hokim@zybo:~/i2c_sw/build$ sudo ./i2c_test
```

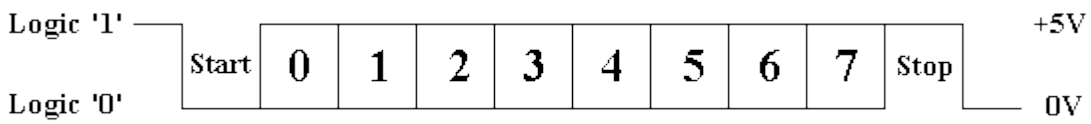
```
hokim@zybo:~$ cd i2c_sw/python
hokim@zybo:~/i2c_sw/python$ ls
i2c_test.py i2cdev.py pmodcls_i2c.py
hokim@zybo:~/i2c_sw/python$ sudo python i2c_test.py
```

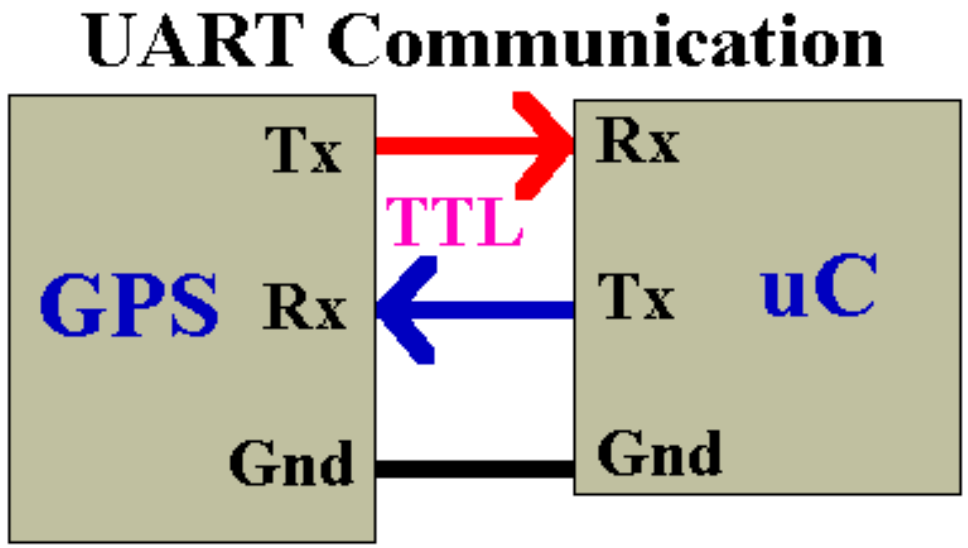
UART

- **Universal asynchronous receiver/transmitter**

  TX : Receiver

  RX : Transmitter
- **Data transmission from 5 bits to 9bits, with parity option and option for 1, 1.5, and 2 stop bits**
- **Various speeds starting from 300bps upto 4Mbps(Most commonly used baud rates : 96800, 115200)**





Linux TTY Interface for UART

INIPRO

- **General interface for each type terminal**

  Serial Port(UART) terminal : /dev/ttySn

  Pseudo terminal : /dev/pts/n

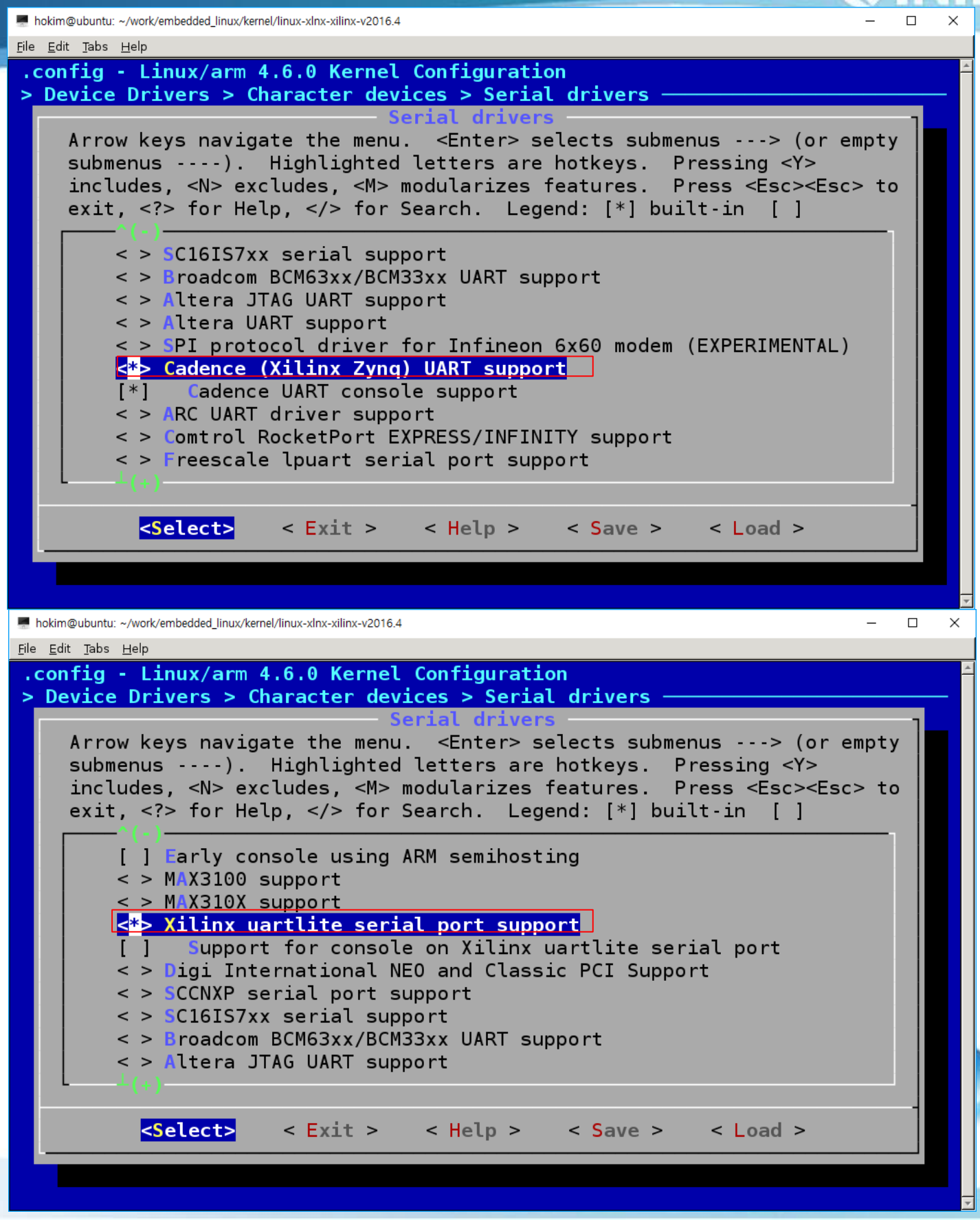
  Controlling terminal : /dev/tty

  Console terminal : /dev/ttyn, /dev/console

- **Documented in Document/serial**
- **Driver : drivers/tty/serial/(xilinx_uartps.c, uartlite.c)**
- **Xilinx UART character device : /dev/ttyPSn, /dev/ttyULn**

Linux UART Userspace Interface

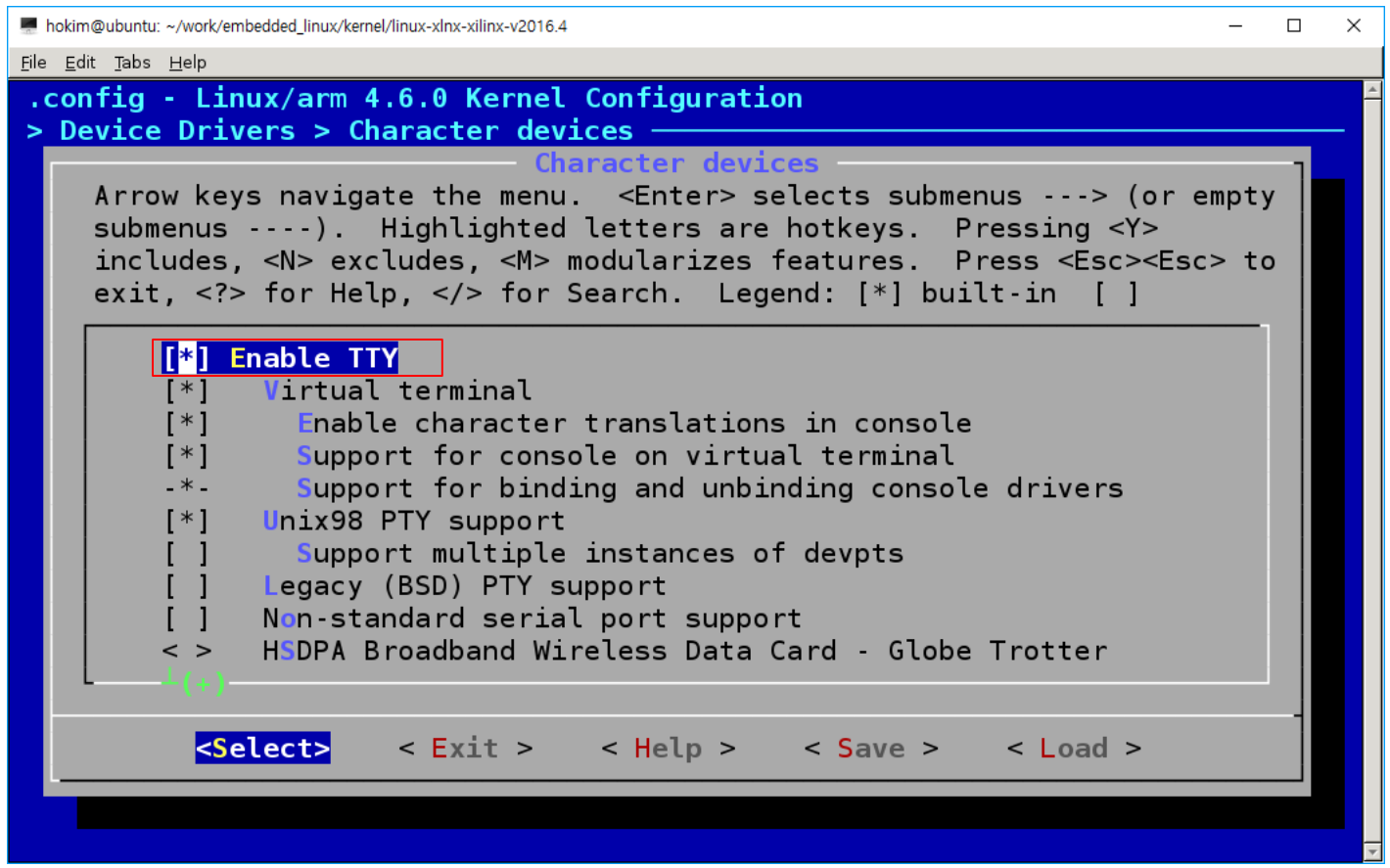
Kernel Configuration



Linux UART Userspace Interface

INIPRO

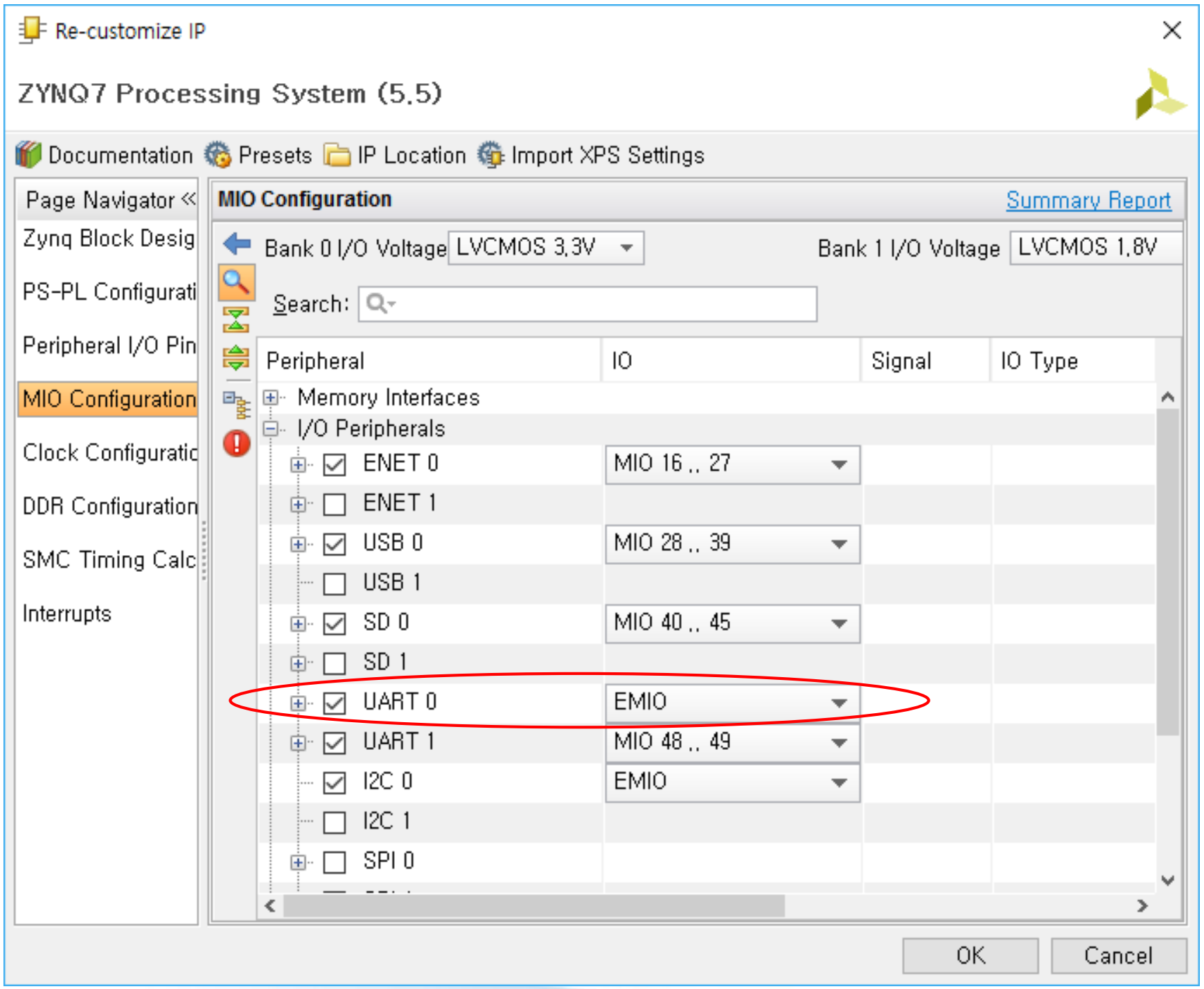
Kernel Configuration



Zynq7 PS Re-customize IP for UART

INIPRO

Based on Zynq7 PS Re-customize IP of embedded_linux project



Zynq7 PS Re-customize IP for UART

INIPRO

Based on Zynq7 PS Re-customize IP of embedded_linux project

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

PS-PL Configuration — Summary Report

Search:

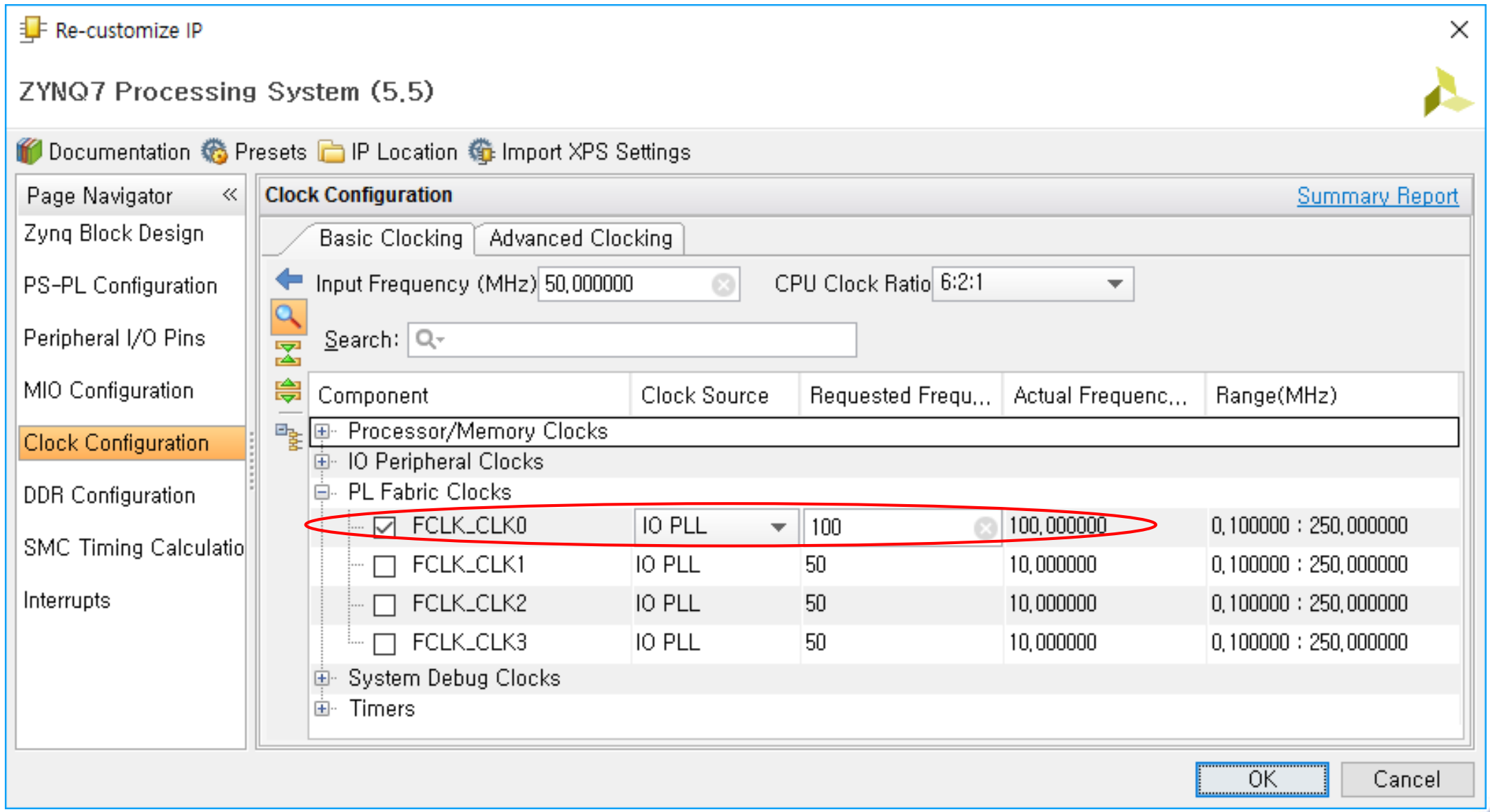
Name	Select	Description
General		
UART0 Baud Rate	115200	Baud rate is generated with internally fixed UART Ref Clock Freq=10...
UART1 Baud Rate	115200	Baud rate is generated with internally fixed UART Ref Clock Freq=10...
PL AXI idle Port	☐	Enables idle AXI signal to the PS used to indicate that there are no ...
DDR ARB bypass Port	☐	Enables DDR urgent/arb signal used to signal a critical memory sta...
PS-PL Debug interface	☐	Enables PL debug signals to PS and vice-versa
FTM Trace data interface	☐	Enables FTM Trace AXI stream interface used to capture data from PL to PS debug system
FTM Trace buffer	0	
FTM Data edge detector	0	Stores trace data in the FIFO when the data changes as marked by ...
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 be...
Include ACP transaction checker	☐	Enables ACP transaction checker.
Trace data/control signal pipeline width	8	Enables configurable number of pipeline stages on the TRACE DAT...
Power-on reset(POR) 4k timer	☐	Enables power-on reset(POR) 4k timer. By default, 64k timer is used.
Processor event interface	☐	Enables event bus which provides a low-latency and direct mecha...
Address Editor		
Enable Clock Triggers		
Enable Clock Resets		
FCLK_RESET0_N	☑	Enables general purpose reset signal 0 for PL logic
FCLK_RESET1_N	☐	Enables general purpose reset signal 1 for PL logic
FCLK_RESET2_N	☐	Enables general purpose reset signal 2 for PL logic
FCLK_RESET3_N	☐	Enables general purpose reset signal 3 for PL logic
AXI Non Secure Enablement	0	Enable AXI Non Secure Transaction
GP Master AXI Interface		
M AXI GP0 interface	☑	Enables General purpose AXI master interface 0
M AXI GP1 interface	☐	Enables General purpose AXI master interface 1
GP Slave AXI Interface		

OK | Cancel

Zynq7 PS Re-customize IP for UART

INIPRO

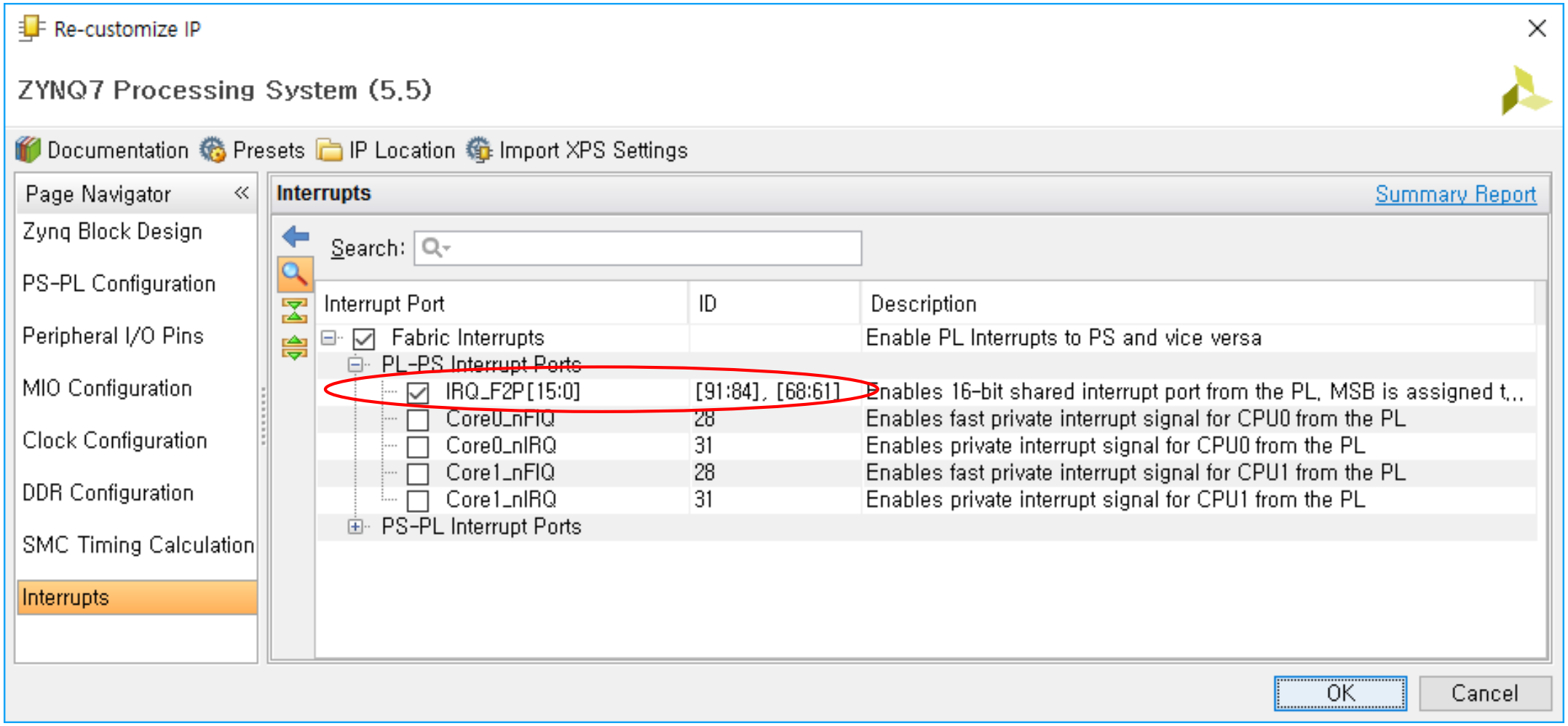
Based on Zynq7 PS Re-customize IP of embedded_linux project



Zynq7 PS Re-customize IP for UART

INIPRO

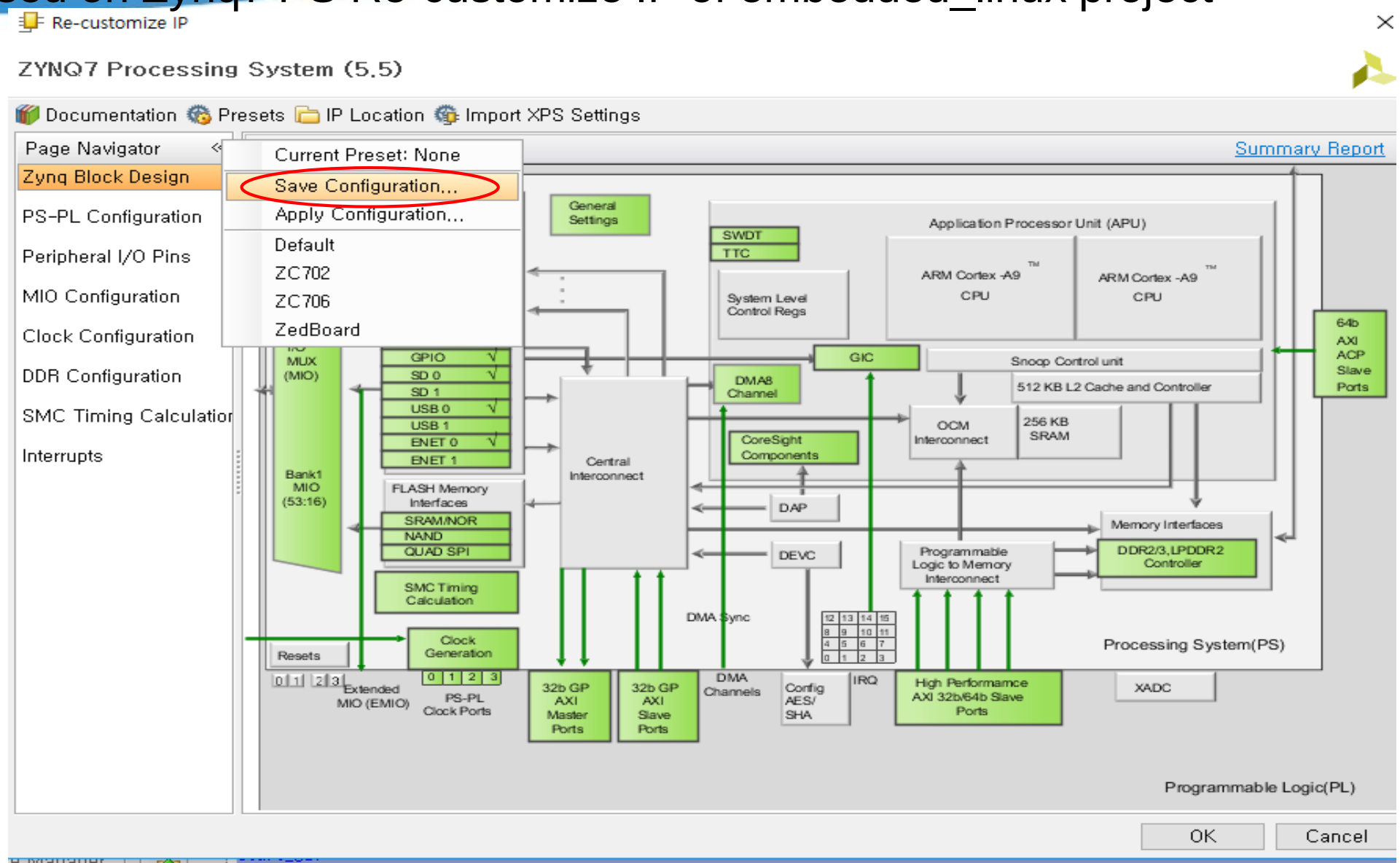
Based on Zynq7 PS Re-customize IP of embedded_linux project

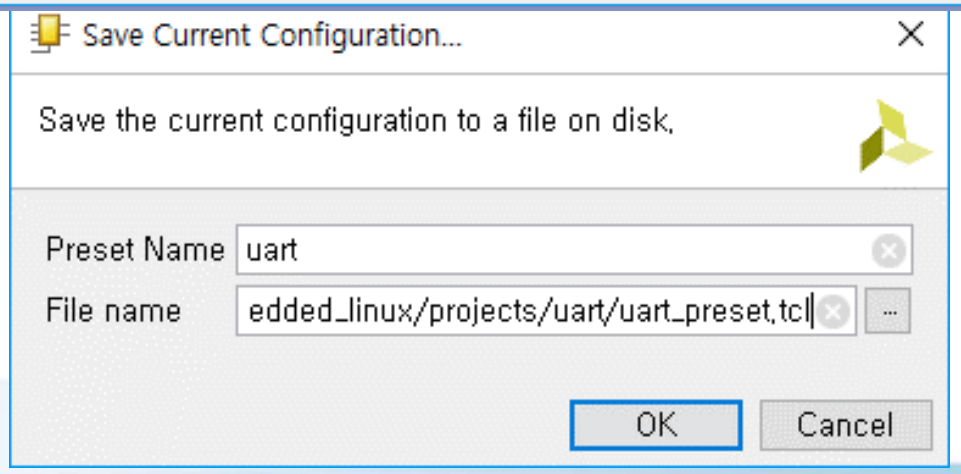


Zynq7 PS Re-customize IP for UART

INIPRO

Based on Zynq7 PS Re-customize IP of embedded_linux project





Project for UART

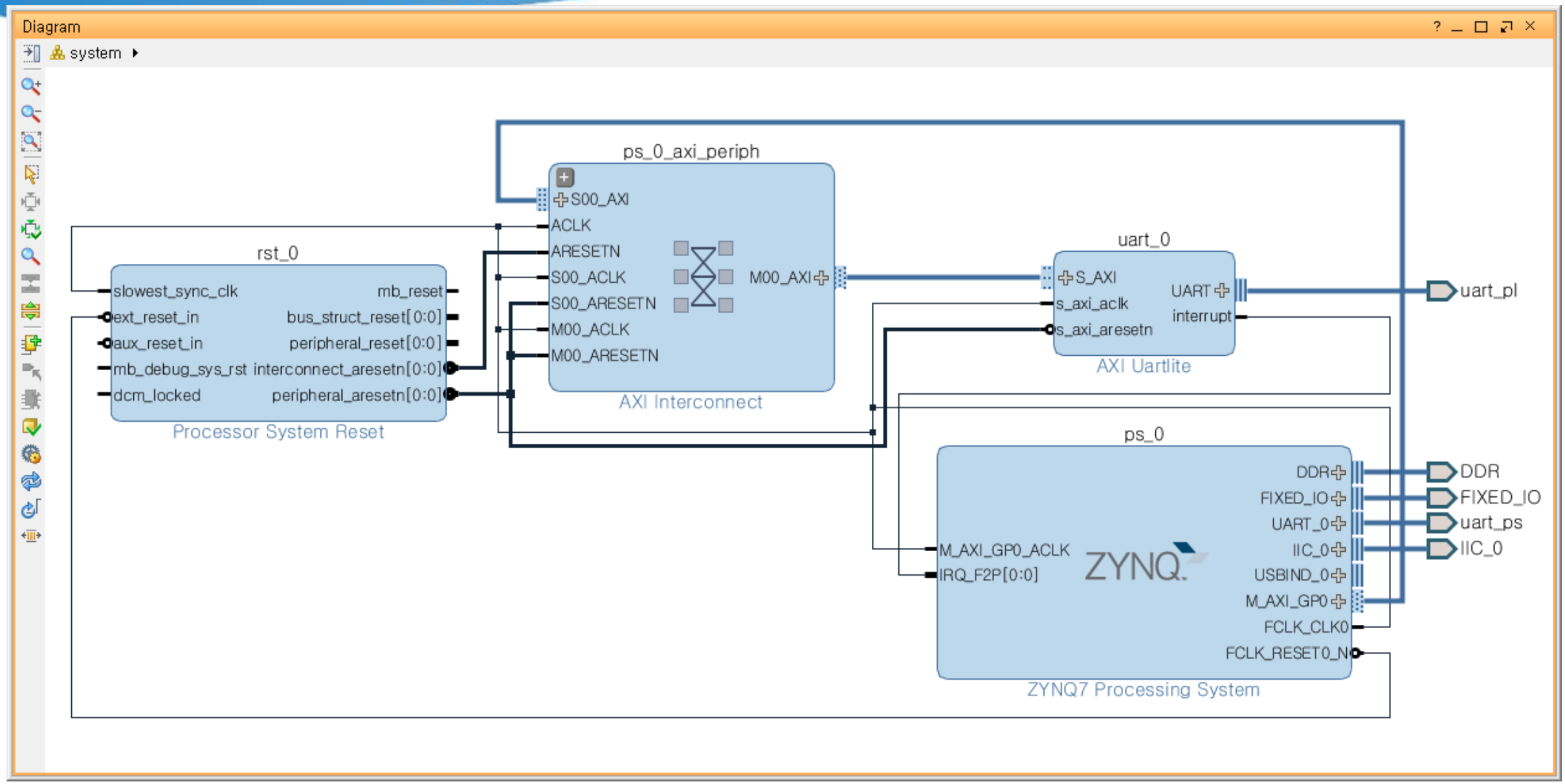
https://github.com/inipro/embedded_linux/projects/uart

In cmd window for Vivado

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

output : uart\uart.xpr…

Project for UART



Address Editor

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
ps_0					
Data (32 address bits : 0x40000000 [1G])					
uart_0	S_AXI	Reg	0x4000_0000	4K	0x4000_0FFF

Bit Generation for UART

INIPRO

uart - [C:/Users/hokim/work/embedded_linux/projects/uart/uart/uart.xpr] - Vivado 2016.4

File Edit Flow Tools Window Layout View Help

Quick Access

Default Layout

Ready

Flow Navigator

- **IP Integrator**
  - Create Block Design
  - Open Block Design
  - Generate Block Design
- Simulation
  - Simulation Settings
  - Run Simulation
- RTL Analysis
  - Elaboration Settings
  - Open Elaborated Design
- Synthesis
  - Synthesis Settings
  - Run Synthesis
  - Open Synthesized Desig
- Implementation
  - Implementation Settings
  - Run Implementation
  - Open Implemented Desig
- Program and Debug
  - Bitstream Settings
  - Generate Bitstream
  - Open Hardware Manage

Block Design - system

Design

- system
  - External Interfaces
  - Interface Connections
  - Nets
  - ps_0 (ZYNQ7 Processing System:5,5)
  - ps_0_axi_periph
  - rst_0 (Processor System Reset:5,0)
  - uart_0 (AXI Uartlite:2,0)

Sources | **Design** | Signals

Block Properties

ps_0

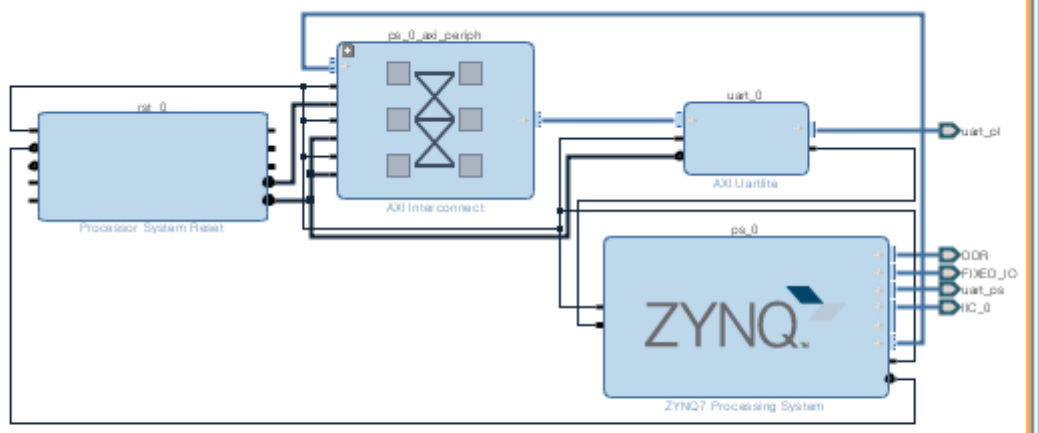
Name: ps_0

Parent name: system

General | Properties | IP

Diagram | Address Editor

system



Tcl Console

```
open_project: Time (s): cpu = 00:00:12 ; elapsed = 00:00:18 . Memory (MB): peak = 710.426 ; gain = 111.191
open_bd_design {C:/Users/hokim/work/embedded_linux/projects/uart/uart/uart.srcs/sources_1/bd/system/system.bd}
Adding cell -- xilinx.com:ip:processing_system7:5.5 - ps_0
Adding cell -- xilinx.com:ip:proc_sys_reset:5.0 - rst_0
```

Type a Tcl command here

Tcl Console | Messages | Log | Reports | Design Runs

Boot.bin Build/DT Generation for UART

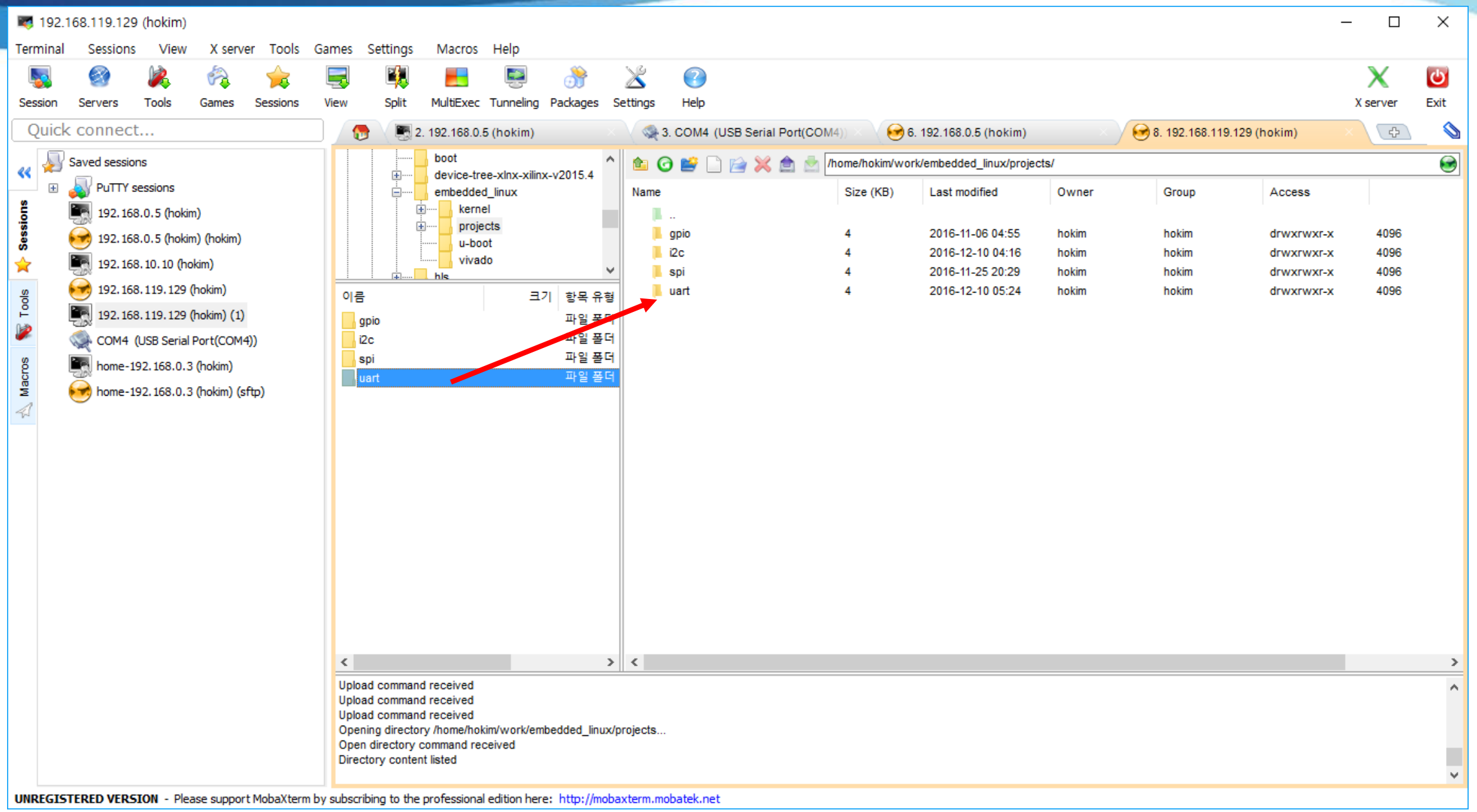
C:\Users\hokim\work\embedded_linux\projects\uart\all.bat

```
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\uart
C:\> all
```

Device Tree Compile for UART

INIPRO




```
$ cd ~/work/embedded_linux/projects/uart
$ cp uart/uart.tree/system.dts  system_uart.dts
$ nano system_uart.dts
```

Device Tree Compile for UART

system_uart.dts

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

```
+/ {
+    aliases {
+        serial0 = &uart1;
+        serial1 = &uart0;
+    };
+};
+&uart0 {
+    port-number = <1>;
+};
+&uart1 {
+    port-number = <0>;
+};
+&uart_0 {
+    port-number = <0>;
+};
```

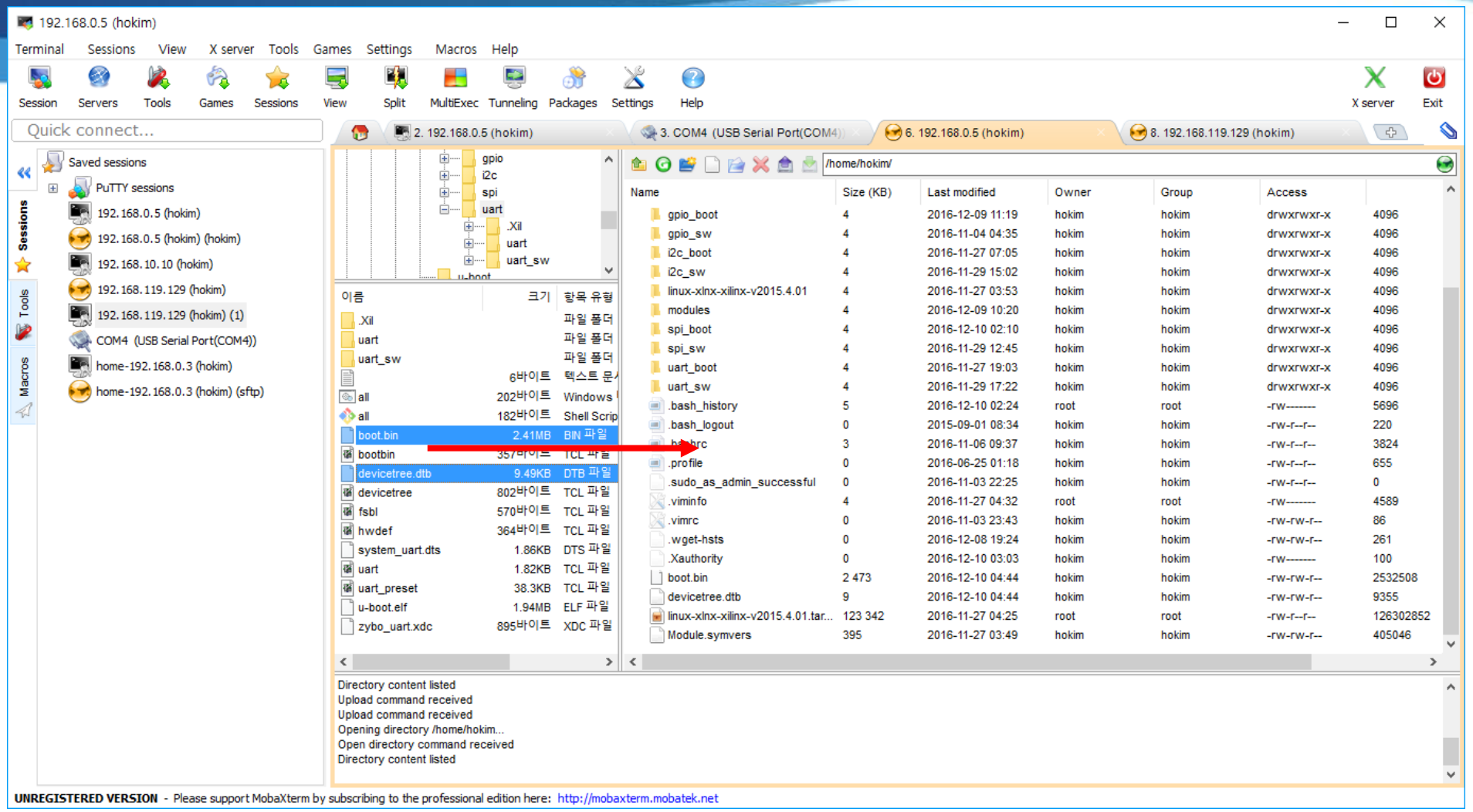
Device Tree Compile for UART

uart/uart.tree/*pl.dtsi*

```
/ {
    amba_pl: amba_pl {
        #address-cells = <1>;
        #size-cells = <1>;
        compatible = "simple-bus";
        ranges ;
        uart_0: serial@40000000 {
            clock-names = "ref_clk";
            clocks = <&clkc 0>;
            compatible = "xlnx,xps-uartlite-1.00.a";
            current-speed = <115200>;
            device_type = "serial";
            interrupt-parent = <&intc>;
            interrupts = <0 29 1>;
            port-number = <2>;
            reg = <0x40000000 0x1000>;
            xlnx,baudrate = <0x2580>;
            xlnx,data-bits = <0x8>;
            xlnx,odd-parity = <0x0>;
            xlnx,s-axi-aclk-freq-hz-d = "100.0";
            xlnx,use-parity = <0x0>;
        };
    };
};
```

```
$ dtc -O dtb -I dts -i uart/uart.tree/ -o devicetree.dtb system_uart.dts
```

Update boot.bin, devicetree.dtb for UART



login into zybo

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

UART sw & test

```
hokim@zybo:~$ ls /dev/ttyPS* /dev/ttyUL*
/dev/ttyPS0  /dev/ttyPS1  /dev/ttyUL0
hokim@zybo:~$ sudo cat /sys/class/tty/ttyPS0/iomem_base
0xE0001000
hokim@zybo:~$ sudo cat /sys/class/tty/ttyPS1/iomem_base
0xE0000000
hokim@zybo:~$ sudo cat /sys/class/tty/ttyUL0/iomem_base
0x40000000
```

```
hokim@zybo:~$ cd uart_sw/
hokim@zybo:~/uart_sw$ ls
CMakeLists.txt  PmodCLS.c  PmodCLS.h  main.c  uart.c  uart.h
hokim@zybo:~/uart_sw$ mkdir build
hokim@zybo:~/uart_sw/$ cd build
hokim@zybo:~/uart_sw/build$ cmake ..
hokim@zybo:~/uart_sw/build$ make
hokim@zybo:~/uart_sw/build$ sudo ./uart_test
```

```
hokim@zybo:~$ cd uart_sw/python
hokim@zybo:~/uart_sw/python$ ls
pmodcls_uart.py uart_test.py
hokim@zybo:~/uart_sw/python$ sudo apt install python-serial
hokim@zybo:~/uart_sw/python$ sudo python uart_test.py
```

Introduction to UIO

- **The Linux kernel provides a framework for doing user space drivers called UIO**
- **The framework is a character mode kernel driver (in drivers/uio) which runs as a layer under a user space driver**
- **UIO helps to offload some of the work to develop a driver**
- **UIO handles simple device drivers really well**

Simple driver: Device access and interrupt processing with no need to access kernel frameworks

Kernel Space Driver Characteristics

INIPRO

- **Advantages**

Runs in kernel space in the highest privilege mode to allow access to interrupts and hardware resources

There are a lot of kernel services such that kernel space drivers can be designed for complex devices

The kernel provides an API to user space which allows multiple applications to access a kernel space driver simultaneously

- Larger and more scalable software systems can be architected

Many drivers tend to be kernel space

- Asking questions in the open source community is going to be easier
- Pusing drivers to the open source community is likely easier

Kernel Space Driver Characteristics

INIPRO

- **Disadvantages**

  System call overhead to access drivers

  - A switch from user space to kernel space (and back) is required
  - Overhead can be non-deterministic having impact on real time applications

  Challenging learning curve for developers

  - The kernel API is different from the application level API such that it takes time to become productive

  Bugs can be fatal causing a kernel crash

  Challenging to debug

  - Kernel code is highly optimized and there are different debug tools

  Frequent kernel API changes

  - Kernel drivers built for one kernel version may not build for another

User Space Driver Characteristics

INIPRO

- **Advantages**

Less challenging to debug as debug tools are more readily available and common to normal application development

User space services such as floating point are available

Device access is very efficient as there is no system call required

The application API of Linux is very stable

The driver can be written in any language, not just “C”

User Space Driver Characteristics

INIPRO

- **Disadvantages**

No access to the kernel frameworks and services

- Contiguous memory allocation, direct cache control, and DMA are not available

- May have to duplicate kernel code or use a kernel driver to supplement

Interrupt handling cannot be done in user space

- It must be handled by a kernel driver which notifies user space causing some delay

There is no predefined API to allow applications to access the device driver

- Concurrency must also be considered if multiple applications access a driver

UIO Framework Features

- **There are two distinct UIO device drivers provided by Linux in drivers/uio**
- **UIO Driver (drivers/uio.c)**

For more advanced users as a minimal kernel space driver is required to setup the UIO framework

This is the most universal and likely to handle all situations since the kernel space driver can be very custom

The majority of work can be accomplished in the user space driver

- **UIO Platform Device Driver (drivers/uio_pdev_irqgen.c)**

This driver augments the UIO driver such that no kernel space driver is required

- It provides the required kernel space driver for uio.c

It works with device tree making it easy to use

- The device tree node for the device needs to use “generic-uio” in it’s compatible property

Best starting point since no kernel space code is needed

- It is the focus of this presentation

UIO Driver Kernel Configuration

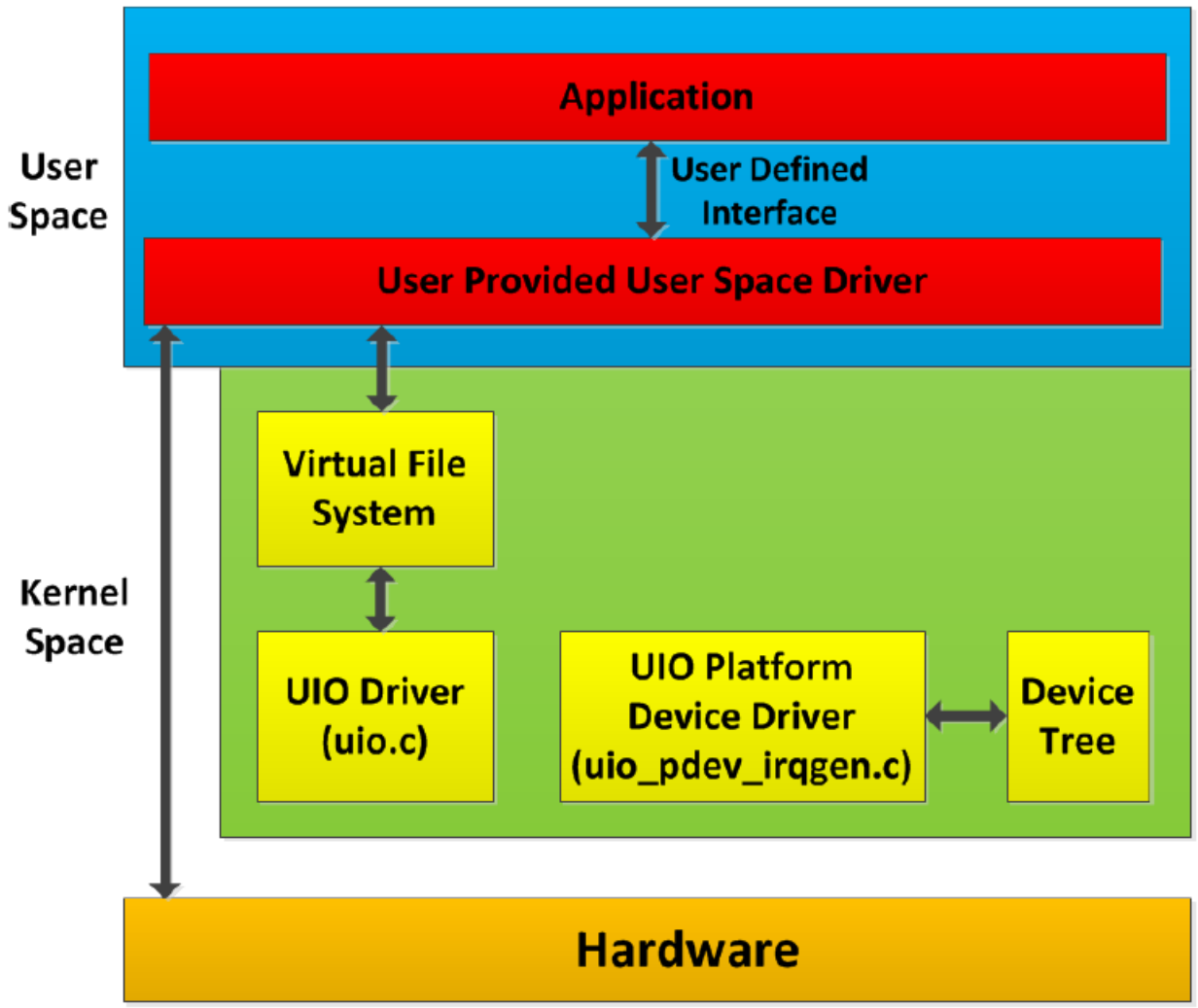
INIPRO

hokim@ubuntu: ~/work/embedded_linux/kernel/linux-xlnx-xilinx-v2016.4

File Edit Tabs Help

```
.config - Linux/arm 4.6.0 Kernel Configuration
> Device Drivers > Userspace I/O drivers
                        Userspace I/O 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  [ ]

      --- Userspace I/O drivers
      < >   generic Hilscher CIF Card driver
      <*>   Userspace I/O platform driver with generic IRQ handling
      < >   Userspace platform driver with generic irq and dynamic memo
      < >   AEC video timestamp device
      < >   Automata Sercos III PCI card driver
      < >   Generic driver for PCI 2.3 and PCI Express cards
      < >   Hilscher NetX Card driver
      < >   Texas Instruments PRUSS driver
      < >   Humusoft MF624 DAQ PCI card driver
      -(+)

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

UIO Platform Device Driver Details



- The user provides only a user space driver
- The UIO platform device driver configures from the device tree and registers a UIO device
- The user space driver has direct access to the hardware
- The user space driver gets notified of an interrupt by reading the UIO device file descriptor

Kernel UIO API – Sys Filesystem

- **The UIO driver in the kernel creates file attributes in the sys filesystem describing the UIO device**
- **/sys/class/uio is the root directory for all the file attributes**
- **A separate numbered directory structure is created under /sys/class/uio for each UIO device**

First UIO device: /sys/class/uio/uio0

/sys/class/uio/uio0/name contains the name of the devices which correlates to the name in the uio_info structure

/sys/class/uio/uio0/maps is a directory that has all the memory ranges for the device

Each numbered map directory has attributes to describe the device memory including the address, name, offset and size

- /sys/class/uio/uio0/maps/map0

User Space Driver Flow

- **The kernel space UIO device driver(s) must be loaded before the user space driver is started (if using modules)**
- **The user space application is started and the UIO device file is opened (/dev/uioX where X is 0, 1, 2…)**

From user space, the UIO device is a device node in the file system just like any other device

- **The device memory address information is found from the relevant sysfs directory, only the size is needed**
- **The device memory is mapped into the process address space by calling the mmap() function of the UIO driver**
- **The application accesses the device hardware to control the device**
- **The device memory is unmapped by calling munmap()**
- **The UIO device file is closed**

DeviceTree for UIO

1. Login into zybo
2. Based on devicetree for gpio project
(Assume devicetree.dtb and boot.bin are saved in ~hokim/gpio_boot directory)

```
$ cd ~hokim/gpio_boot
$ ls
 boot.bin  devicetree.dtb
```

3. Convert devicetree.dtb into devicetree_gpio-uio.dts for uio using dtc command

```
$ dtc -I dtb  -O dts -o devicetree_gpio-uio.dts devicetree.dtb
$ ls
 boot.bin  devicetree.dtb devicetree_gpio-uio.dts
```

4. Edit devicetree_gpio-uio.dts for UIO

```
$ nano devicetree_gpio-uio.dts
```

DeviceTree for UIO

INIPRO

devicetree_gpio-uio.dts

```
/dts-v1/;

/ {
    #address-cells = <0x1>;
    #size-cells = <0x1>;
    compatible = "xlnx,zynq-7000";

    chosen {
-       bootargs = "console=ttyPS0,115200 root=/dev/mmcblk0p2 ro
rootfstype=ext4 earlyprintk rootwait";
+       bootargs = "console=ttyPS0,115200 root=/dev/mmcblk0p2 ro
rootfstype=ext4 earlyprintk rootwait uio_pdrv_genirq.of_id=generic-
uio";
    };
.............
.............
+/*
      gpio@40000000 {
          #gpio-cells = <0x2>;
          #interrupt-cells = <0x2>;
          compatible = "xlnx,xps-gpio-1.00.a";
          gpio-controller;
          interrupt-controller;
          interrupt-parent = <0x3>;
          interrupts = <0x0 0x1f 0x4>;
          reg = <0x40000000 0x1000>;
.............
.............
        };
+*/
```

```
+       leds@40000000 {
+           compatible = "generic-uio";
+           interrupt-parent = <0x3>;
+           interrupts = <0x0 0x1f 0x4>;
+           reg = <0x40000000 0x1000>;
+       };
+/*
      gpio@40002000 {
          #gpio-cells = <0x2>;
          #interrupt-cells = <0x2>;
          compatible = "xlnx,xps-gpio-1.00.a";
          gpio-controller;
          interrupt-controller;
          interrupt-parent = <0x3>;
          interrupts = <0x0 0x1d 0x4>;
.............
.............
        };
+*/
+       sws@40002000 {
+           compatible = "generic-uio";
+           interrupt-parent = <0x3>;
+           interrupts = <0x0 0x1d 0x4>;
+           reg = <0x40002000 0x1000>;
+       };
```

DeviceTree for UIO

5. Convert devicetree_gpio-uio.dts into devicetree_gpio-uio.dtb

```
$ dtc -I dts  -O dtb -o devicetree_gpio-uio.dtb devicetree_gpio-uio.dts
$ ls
 boot.bin  devicetree.dtb devicetree_gpio-uio.dts devicetree_gpio-uio.dtb
```

6. Copy boot.bin, devicetree_gpio-uio.dtb to /boot directory and reboot

```
$ sudo -s
# cp boot.bin /boot
# cp devicetree_gpio-uio.dtb /boot/devicetree.dtb
# halt
```

UIO Test

INIPRO

uio_test.c

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/mman.h>
#include <fcntl.h>

#define GPIO_DATA_OFFSET    0x0     /* Data register for 1st
channel */
#define GPIO_TRI_OFFSET     0x4     /* I/O direction reg for 1st
channel */
#define GPIO_DATA2_OFFSET   0x8     /* Data register for 2nd
channel */
#define GPIO_TRI2_OFFSET    0xC     /* I/O direction reg for 2nd
channel */
#define GPIO_GIE_OFFSET     0x11C   /* Global interrupt enable
register */
#define GPIO_ISR_OFFSET     0x120   /* Interrupt status register */
#define GPIO_IER_OFFSET     0x128   /* Interrupt enable register
*/

void gpio_write(void *gpio_base, uint32_t offset, uint32_t value)
{
    *((volatile unsigned *)(gpio_base + offset)) = value;
}

uint32_t gpio_read(void *gpio_base, uint32_t offset)
{
    return *((volatile unsigned *)(gpio_base + offset));
}
int32_t get_memory_size(char *sysfs_path_file)
{
    FILE *size_fp;
    uint32_t size;
```

```
 // open the file that describes the memory range size that is based
on the
    // reg property of the node in the device tree
    size_fp = fopen(sysfs_path_file, "r");
int32_t get_memory_size(char *sysfs_path_file)
{
    FILE *size_fp;
    uint32_t size;

    // open the file that describes the memory range size that is
based on the
    // reg property of the node in the device tree
    size_fp = fopen(sysfs_path_file, "r");

    if (!size_fp) {
        printf("unable to open the uio size file\n");
        exit(-1);
    }

    // get the size which is an ASCII string such as 0xXXXXXXXX
and then be stop
    // using the file
    fscanf(size_fp, "0x%08X", &size);
    fclose(size_fp);

    return size;
}
void wait_for_interrupt(int fd, void *gpio_ptr)
{
    int pending = 0;
    int reenable = 1;
    uint32_t reg;

    // block on the file waiting for an interrupt
    read(fd, (void *)&pending, sizeof(int));
    //printf("Interrupt!\n");
```

UIO Test

INIPRO

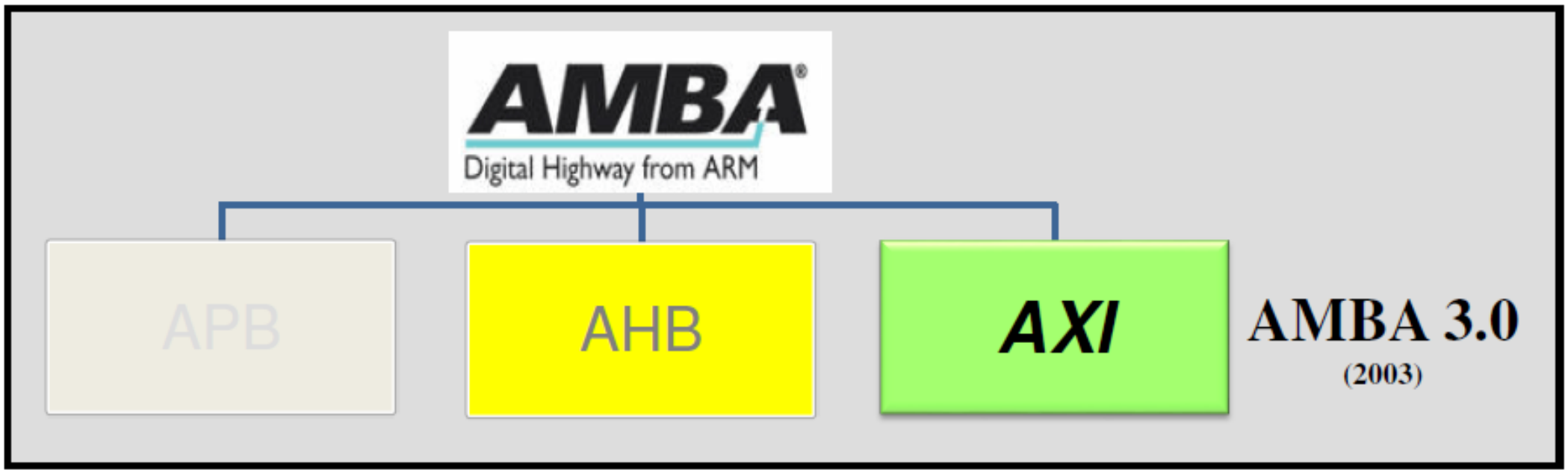
uio_test.c

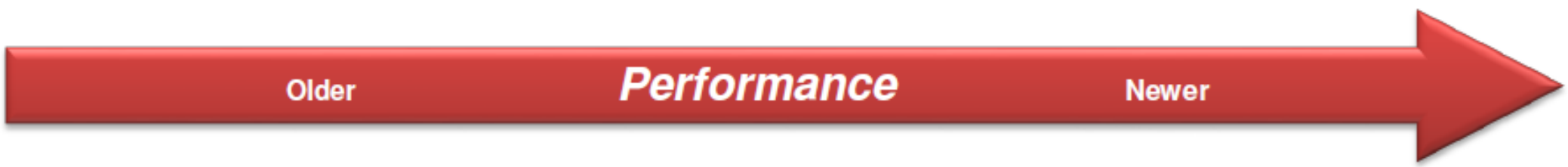
```
// the interrupt occurred for the 1st GPIO channel so clear it
    reg = gpio_read(gpio_ptr, GPIO_ISR_OFFSET);
    if (reg) gpio_write(gpio_ptr, GPIO_ISR_OFFSET, 1);

    // re-enable the interrupt in the interrupt controller thru
    // the UIO subsystem now that it's been handled

    write(fd, (void *)&reenable, sizeof(int));
}

int main()
{
    int uio0_fd, uio1_fd;
    void *ptr0;
    void *ptr1;
    uint32_t gpio_size0, gpio_size1;
    int reenable = 1;
    uint32_t value;

    if ((uio0_fd = open("/dev/uio0", O_RDWR)) < 0) {
        perror("open uio0");
    }
 if ((uio1_fd = open("/dev/uio1", O_RDWR)) < 0) {
        perror("open uio1");
    }

    gpio_size0 =
get_memory_size("/sys/class/uio/uio0/maps/map0/size");
    gpio_size1 =
get_memory_size("/sys/class/uio/uio1/maps/map0/size");

    ptr0 =  mmap(NULL, gpio_size0, PROT_READ|PROT_WRITE,
MAP_SHARED, uio0_fd, 0);
 if (ptr0 == MAP_FAILED) {
        perror("uio0 mmap call failure\n");
        return -1;
    }

    ptr1 =  mmap(NULL, gpio_size1, PROT_READ|PROT_WRITE,
MAP_SHARED, uio1_fd, 0);

    if (ptr1 == MAP_FAILED) {
        perror("uio1 mmap call failure\n");
        return -1;
    }

    gpio_write(ptr0, GPIO_TRI_OFFSET, 0x0); // GPIO Channel 1
input

    gpio_write(ptr1, GPIO_TRI_OFFSET, 0xF); // GPIO Channel 1
input
    gpio_write(ptr1, GPIO_GIE_OFFSET, 0x80000000); // GIER,
31st Bit
    gpio_write(ptr1, GPIO_IER_OFFSET, 1); // interrupt enable

    write(uio1_fd, (void*)&reenable,  sizeof(int));
    while (1)
    {
        wait_for_interrupt(uio1_fd, ptr1);
        value = gpio_read(ptr1, GPIO_DATA_OFFSET);
        //printf("sws = 0x%0X\n", value);
        gpio_write(ptr0, GPIO_DATA_OFFSET, value);
    }

    munmap(ptr0, gpio_size0);
    munmap(ptr1, gpio_size1);

    return 0;
}
```

UIO Test

```
$ cd ~/uio_sw
$ ls
$ CMakeLists.txt uio_test.c
$ mkdir build
$ cd build
$ cmake
$ make
$ sudo ./uio_test
```

AXI is Part's of ARM's AMBA

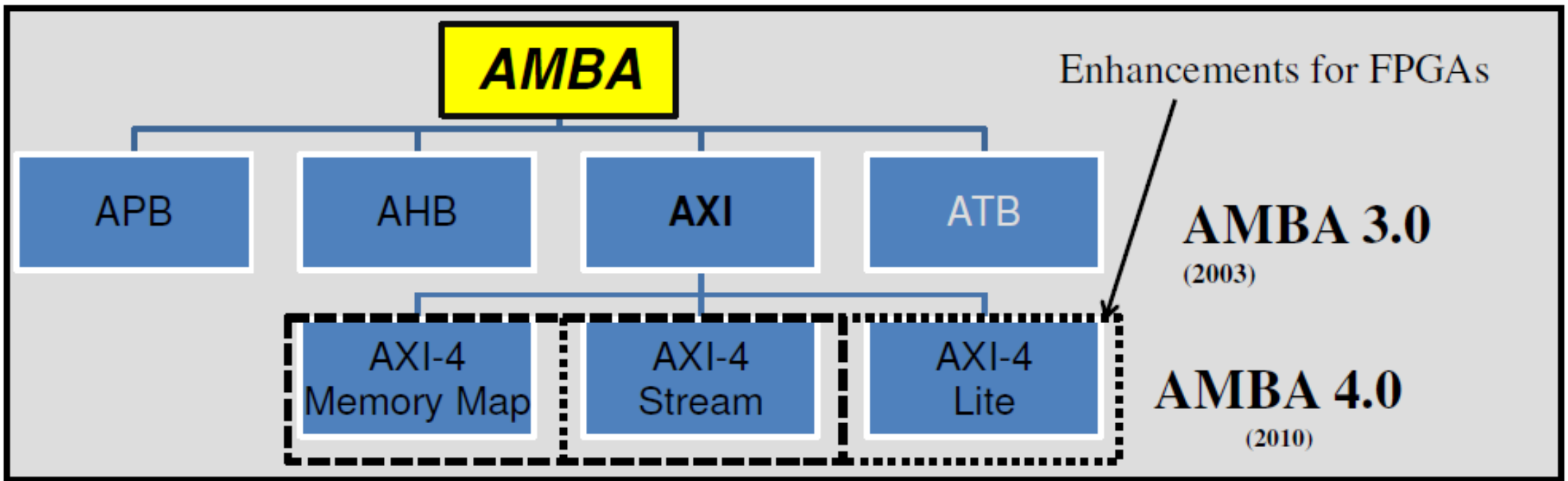




AMBA: Advanced Microcontroller Bus Architecture

AXI: Advanced Extensible Interface

AXI-4 (Full, Lite and Stream)



Interface	Features	Similar to
Memory Map / Full (AXI4)	Traditional Address/Data Burst (single address, multiple data)	PLBv46, PCI
Streaming (AXI4-Stream)	Data-Only, Burst	Local Link / DSP Interfaces / FIFO / FSL
Lite (AXI4-Lite)	Traditional Address/Data—No Burst (single address, single data)	PLBv46-single OPB

Types of AXI4 interfaces

- **AXI4 (a.k.a AXI4-full)**

  For high-performance memory-mapped requirements

- **AXI4-Lite**

  For simple, low-throughput memory-mapped communication (for example, to and from control and status registers)

- **AXI4-Stream**

  For high-speed streaming data

AXI4 Protocols

- **Memory Mapped Protocol : AXI3, AXI4, AXI4-Lite**

  All transactions involve the concept of a target address within a system memory space and data to be transferred

  Example : A typical peripheral or memory bank

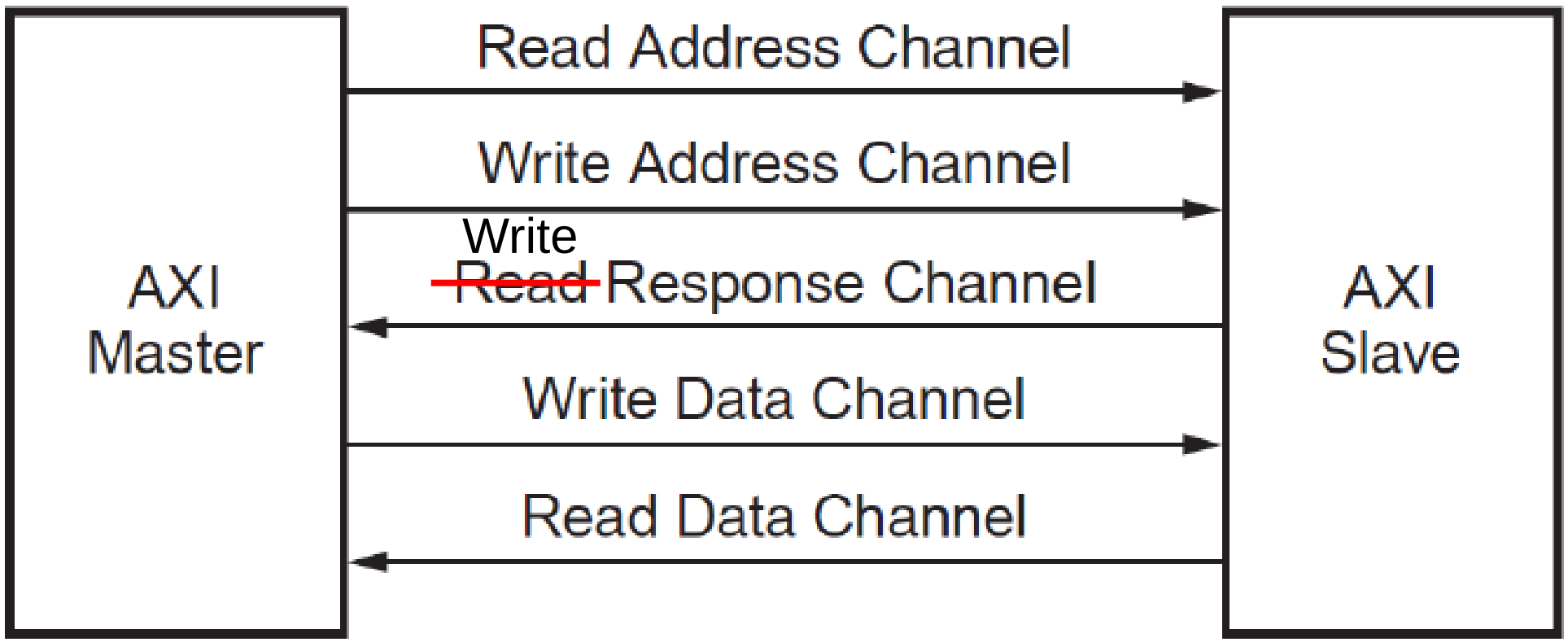
- **Stream Protocol : AXI4-Stream**

  It’s used for applications that typically focus on data-centric and data-flow paradigm where the concept of an address is not present or not required

  Example : data coming from an Ethernet controller

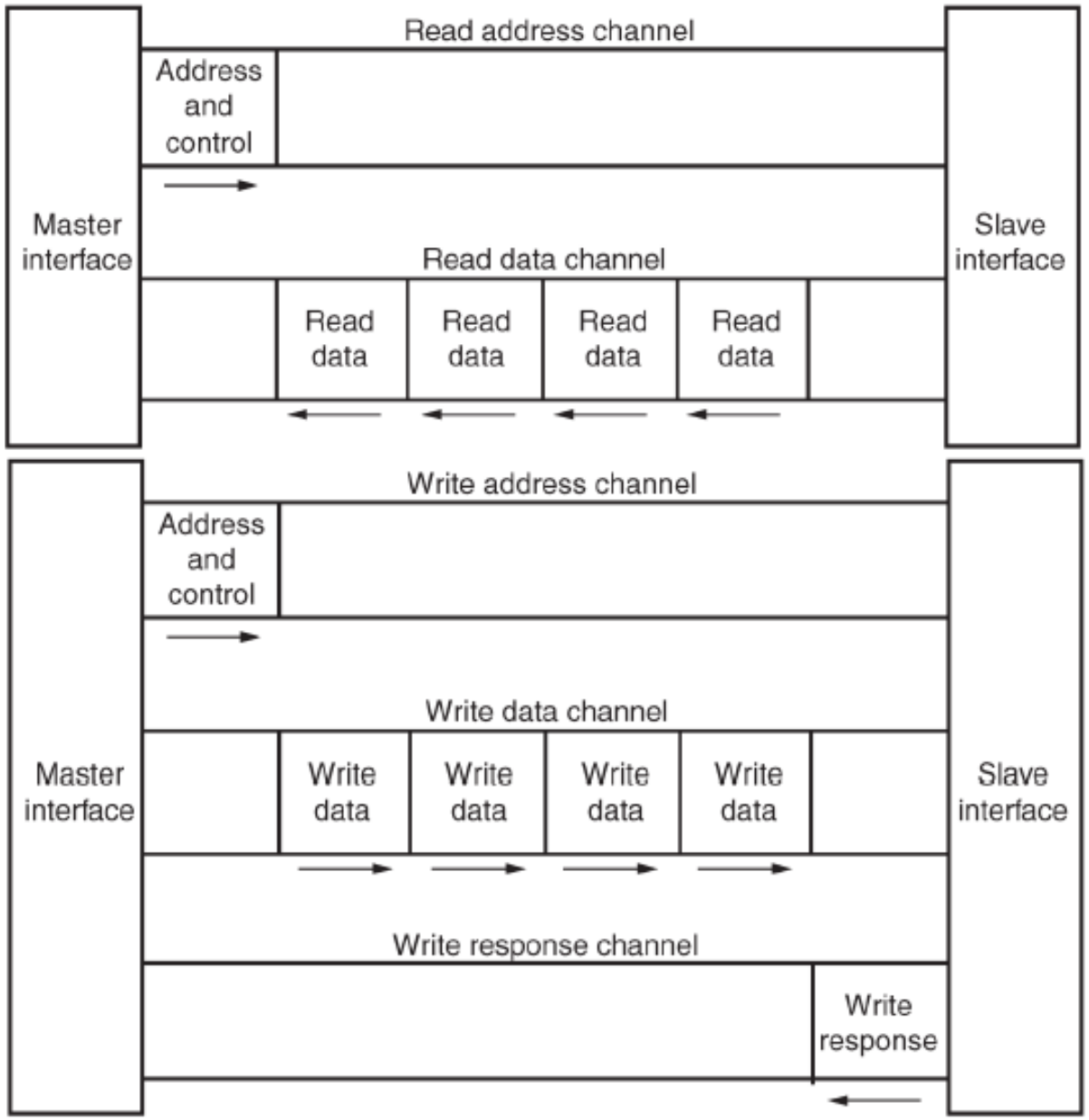
AXI4 MM Protocol

- **A typical MM master-slave connections**



AXI4 MM Protocol

- **Read address channel**
- **Read data channel**



- **Write address channel**
- **Write data channel**
- **Write response channel**

Non-posted write model

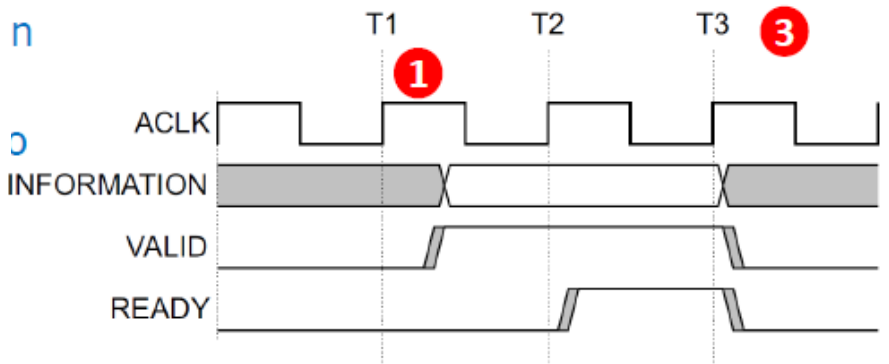
There will always will be a “write response”

AXI : Channel handshaking

- **All AXI channels use a basic VALID/READY handshaking:**

SOURCE asserts and holds VALID when DATA is available

DESTINATION asserts READY if able to accept DATA



DATA transferred when VALID and READY = 1

SOURCE sends next DATA(if an actual data channel) or deasserts VALID

DESTINATION deasserts READY if no longer able to accept DATA

AXI : Channel handshaking

- **AXI uses a valid/ready handsake acknowledge**
- **Each channel has its own valid/ready**

Address (read/write)

Data (read/write)

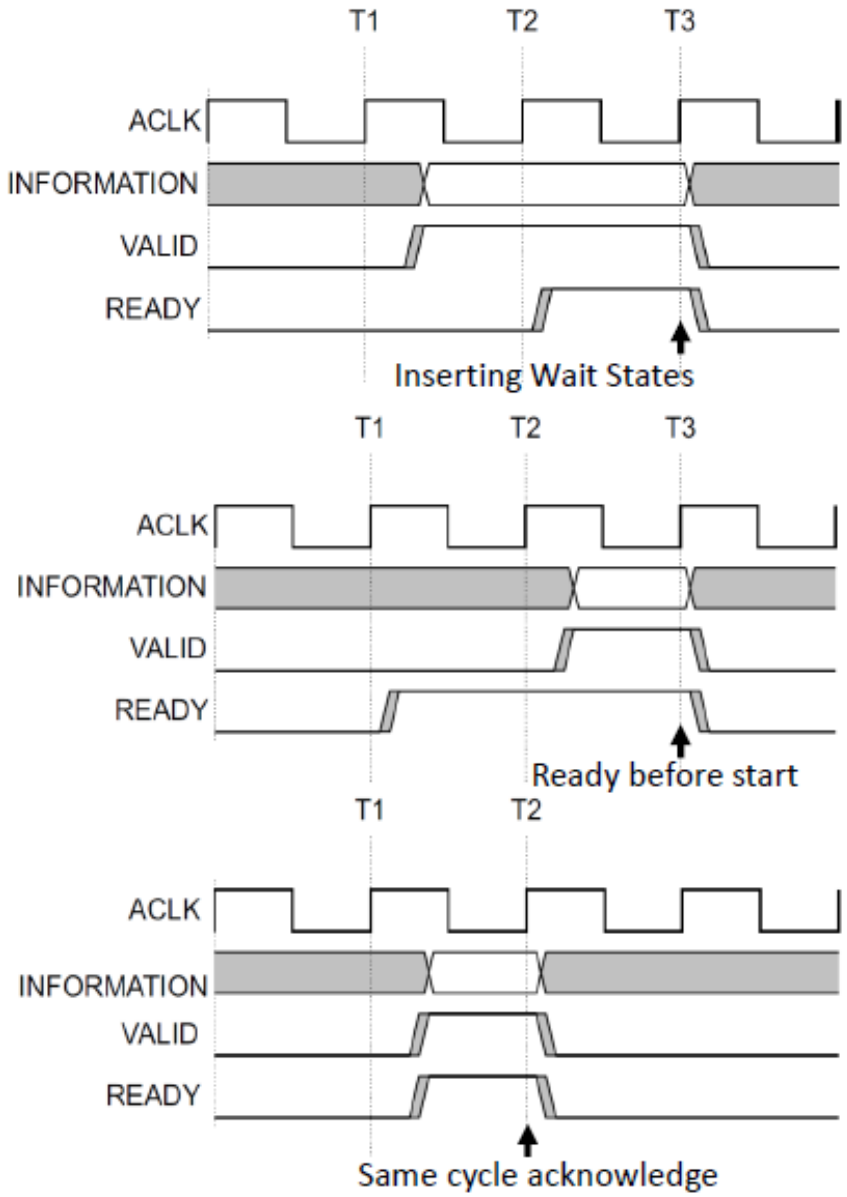
Response (read only)

- **Flexible signaling functionality**

Inserting wait states

Always ready

Same cycle acknowledge



AXI MM signals

	AXI4	AXI4-Lite
Glbl	ACLK	
	ARESETN	
Write Address	AWID	
	AWADDR	
	AWLEN	
	AWSIZE	
	AWBURST	
	AWLOCK	
	AWCACHE	
	AWPROT	
	AWQOS	
	AWSIZE	
	AWREGION	
	AWLOCK	
	AWUSER	
	AWVALID	
	AWREADY	

	AXI4	AXI4-Lite
Read Address	ARID	
	ARADDR	
	ARLEN	
	ARSIZE	
	ARBURST	
	ARLOCK	
	ARCACHE	ARCACHE
	ARPROT	ARPROT
	ARQOS	
	ARREGION	
	ARUSER	
	ARVALID	
	ARREADY	

	AXI4	AXI4-Lite
Write Data	WDATA	WDATA
	WSTRB	WSTRB
	WLAST	
	WUSER	
	WVALID	
	WREADY	
Write Resp.	BID	
	BRESP	BRESP
	BUSER	
	BVALID	
	BREADY	

	AXI4	AXI4-Lite
Read Data	RID	
	RDATA	RDATA
	RRESP	RRESP
	RLAST	
	RUSER	
	RVALID	
	WREADY	

AXI MM signals : Address channels

- **AWADDR (ARADDR): Write (read) address**

The write (read) address gives the address of the first transfer in a write (read) burst transaction

- **AWVALID (ARVALID): Write (read) address valid**

This signal indicates that the channel is signaling valid write (read) address and control information

- **AWREADY (ARREADY): Write (read) address ready**

This signal indicates that the slave is ready to accept an address and associated control signals

AXI MM signals : Address channels

- **AWID (ARID): Write (read) address ID**

  This signal is the identification tag for the write address group of signals

- **AWLEN (ARLEN): Burst length**

  This burst length gives the exact number of transfer in a burst (number of transfers) ; length = N + 1

- **AWSIZE (ARSIZE): Burst size**

  This signal indicates the size of each transfer in the burst(size=2^N)

- **AWBURST (ARBURST): Burst type**

  The burst type and the size information, determine how the address for each transfer within the burst is calculated. FIXED, INCR, WRAP

- **AWLOCK (ARLOCK): Lock type**

  Provides additional information about the atomic characteristics of the transfer

- **AWPROT (ARPROT): Protection type**

  This signal indicates the privilege and security level of the transaction, and whether the transaction is a data access or an instruction access

AXI MM signals : Address channels

- **AWCACHE (ARCACHE): Memory type**

The signal indicates how transactions are required to progress through a system

- **AWQOS (ARQOS): Quality of Service, QoS**

This QoS identifier sent for each write transaction

- **AWREGION (ARREGION): Region identifier**

Permits a single physical interface on a slave to be used for multiple logical interfaces

- **AWUSER (ARUSER): User signal**

Optional User-defined signal in the write address channel

AXI MM signals : Write data channel

- **WID: Write ID tag**

  This signal is the ID tag of the write data transfer

- **WDATA: Write data**
- **WSTRB: Write strobes**

  This signal indicates which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus

- **WLAST: Write last**

  The signal indicates the last transfer in a write burst

- **WUSER: User signal**

  Optional User-defined signal in the write data channel

- **WVALID: Write valid**

  This signal indicates that valid write data and strobes are available

- **WREADY: Write ready**

  This signal indicates that the slave can accept the write data

AXI MM signals : Write response channel

- **BID: Response ID tag**

  This signal is the ID tag of the write response
- **BRESP: Write response**

  This signal indicates the status of the write transaction

  OKAY, EXOKAY, SLVERR, DECERR
- **BUSER: User signal**

  Optional User-defined signal in the write response channel
- **BVALID: Write response valid**

  This signal indicates that the channel is signaling a valid write response
- **BREADY: Response ready**

  This signal indicates that the master can accept a write response

AXI MM signals : Read data channel

- **RID: Read ID tag**

This signal is the identification tag for the read data group of signals generated by the slave

- **RDATA: Read data**
- **RRESP: Read response**

This signal indicates the status of the read transaction

OKAY, EXOKAY, SLVERR, DECERR

- **RLAST: Read last**

This signal indicates the last transfer in a read burst

- **BUSER: User signal**

Optional User-defined signal in the read data channel

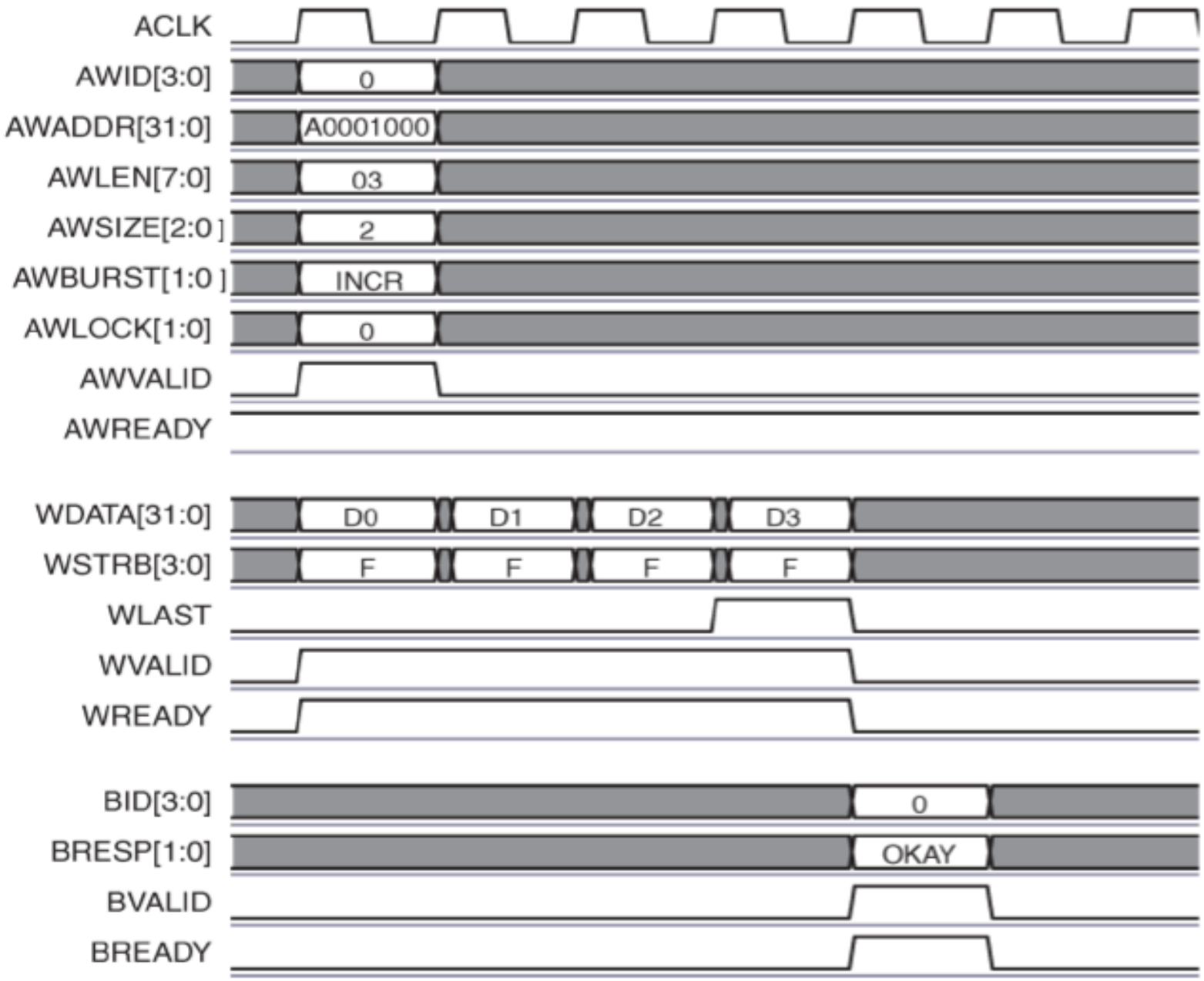
- **BVALID: Read valid**

This signal indicates that the channel is signaling the required read data

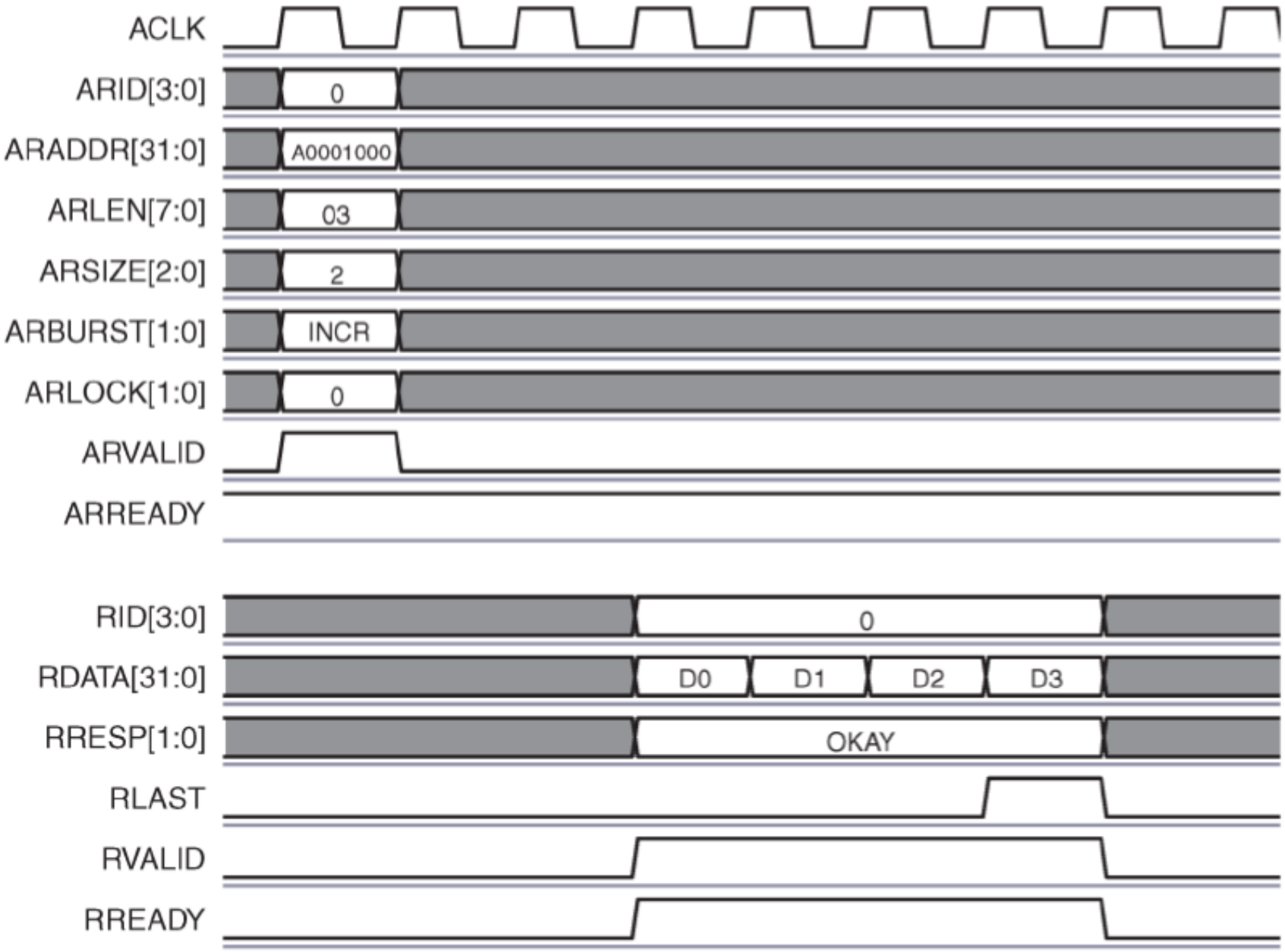
- **RREADY: Read ready**

This signal indicates that the master can accept the read data and response information

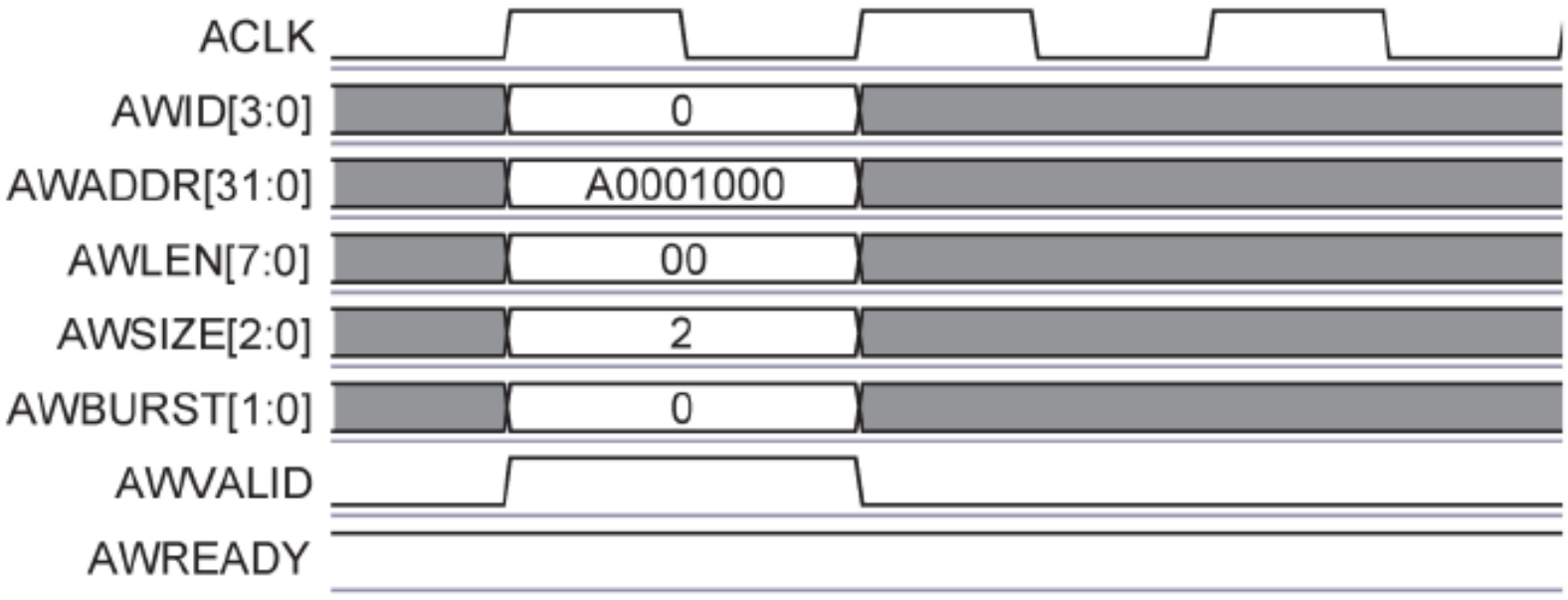
AXI4 Burst Write transaction

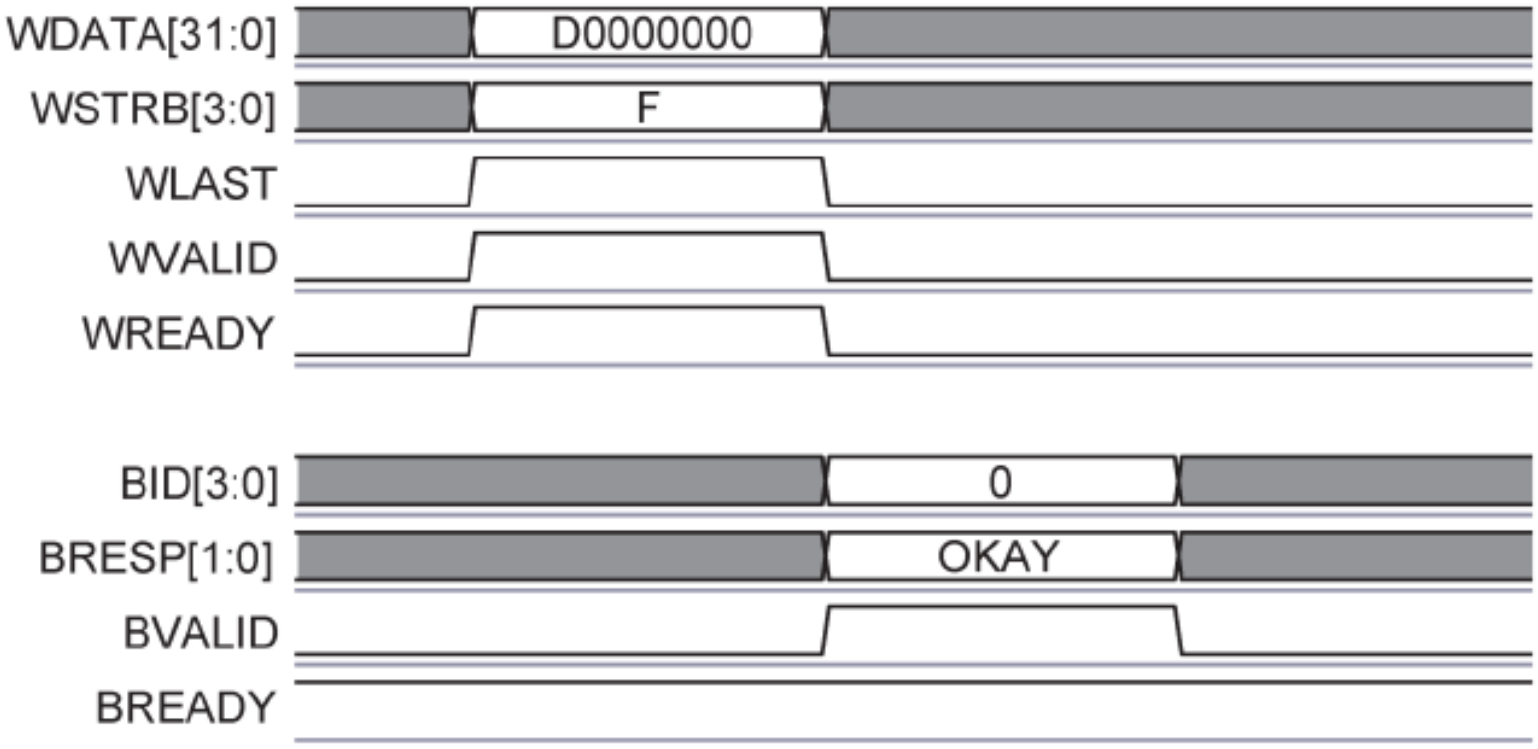


AXI4 Burst Read transaction

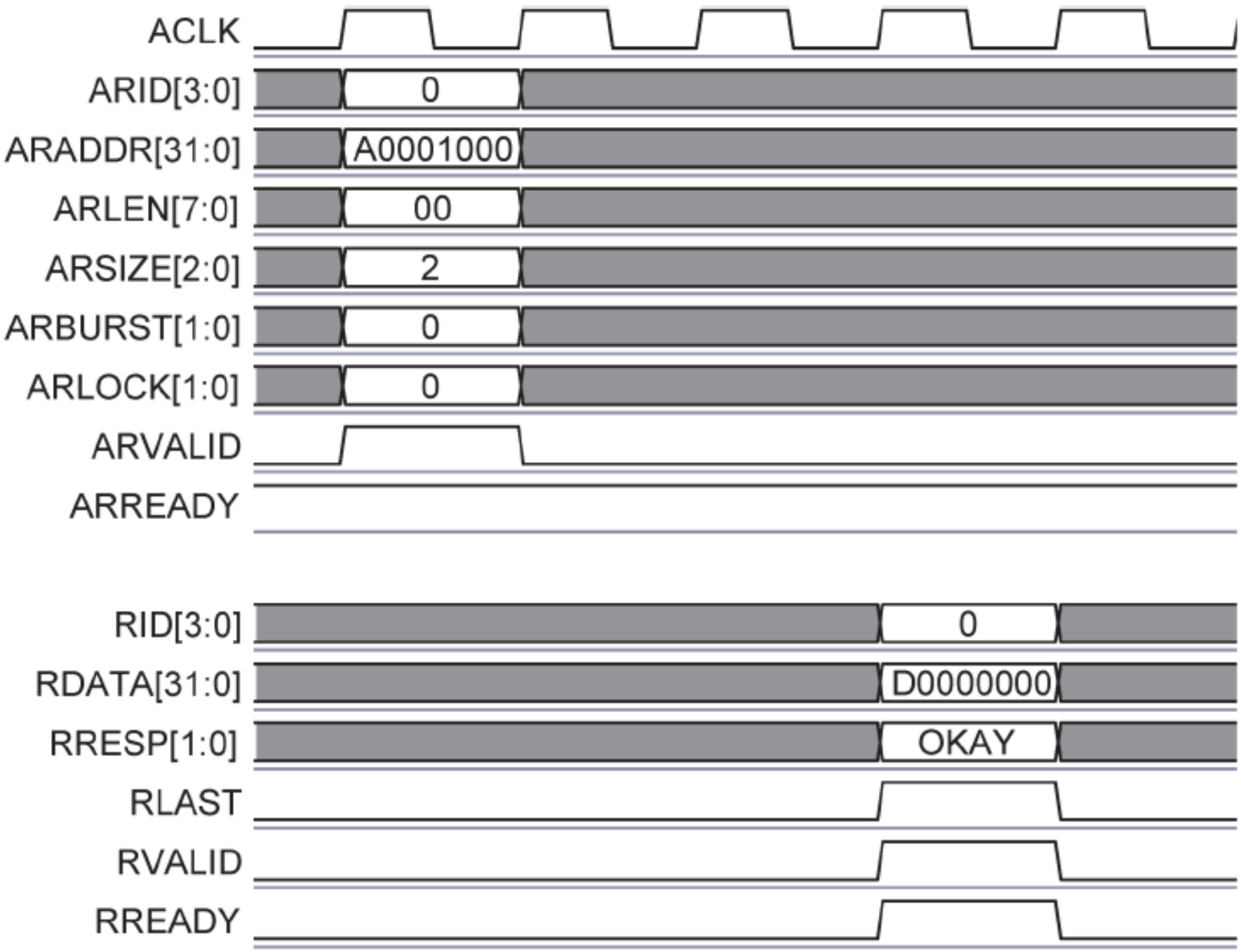


AXI4 Single Write transaction





AXI4 Single Read transaction



AXI LITE

- **AXI-lite is subset of AXI4 for communication with simpler control register style interfaces within components**

all transactions are of burst length 1

all data accesses use the full width of the data bus

AXI4-Lite supports a data bus width of 32-bit or 64-bit

all accesses are Non-modifiable, Non-bufferable

Exclusive accesses are not supported

AXI LITE : Signals

INIPRO

Global	Write address channel	Write data channel	Write response channel	Read address channel	Read data channel
ACLK	AWVALID	WVALID	BVALID	ARVALID	RVALID
ARESETn	AWREADY	WREADY	BREADY	ARREADY	RREADY
–	AWADDR	WDATA	BRESP	ARADDR	RDATA
–	AWPROT	WSTRB	–	ARPROT	RRESP

AXI LITE : Not supported signals

- **AWLEN, ARLEN**

  This burst length is defined to be 1, equivalent to an AxLEN value of zero

- **AWSIZE, ARSIZE**

  All accesses are defined to be the width of the data bus

  AXI4-Lite requires a fixed data bus width of ether 32-bits or 64-bits

- **AWBURST, ARBURST**

  This burst type has no meaning because the burst length is 1

- **AWLOCK, ARLOCK**

  All accesses are defined as Normal accesses, equivalent to an AxLOCK=0

- **AWCACHE, ARCACHE**

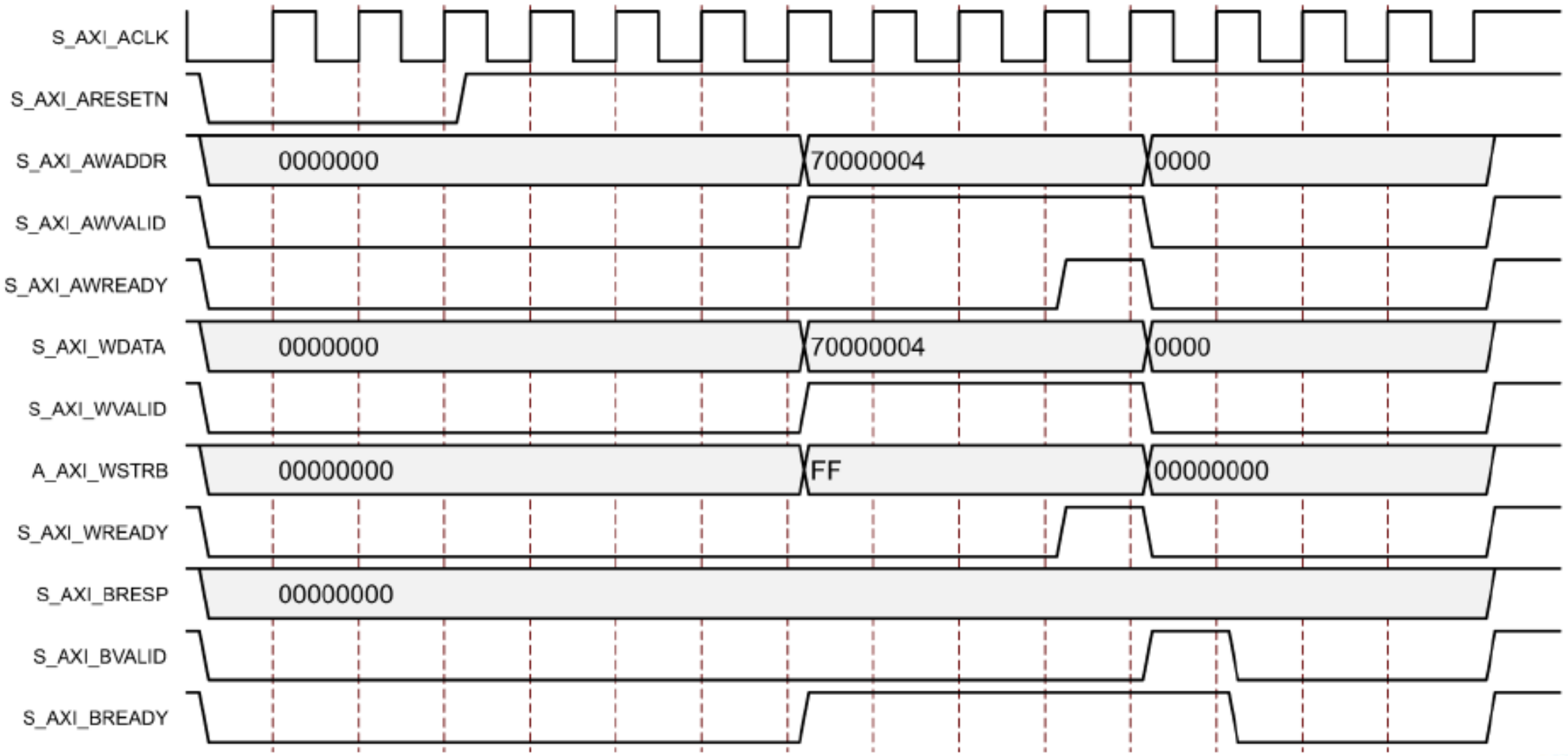
  All accesses are defined as Non-modifiable, Non-bufferable, equivalent to an AxCACHE=0b0000

- **WLAST, RLAST**

  All bursts are defined to be of length 1, equivalent to a WLAST or RLAST value of 1

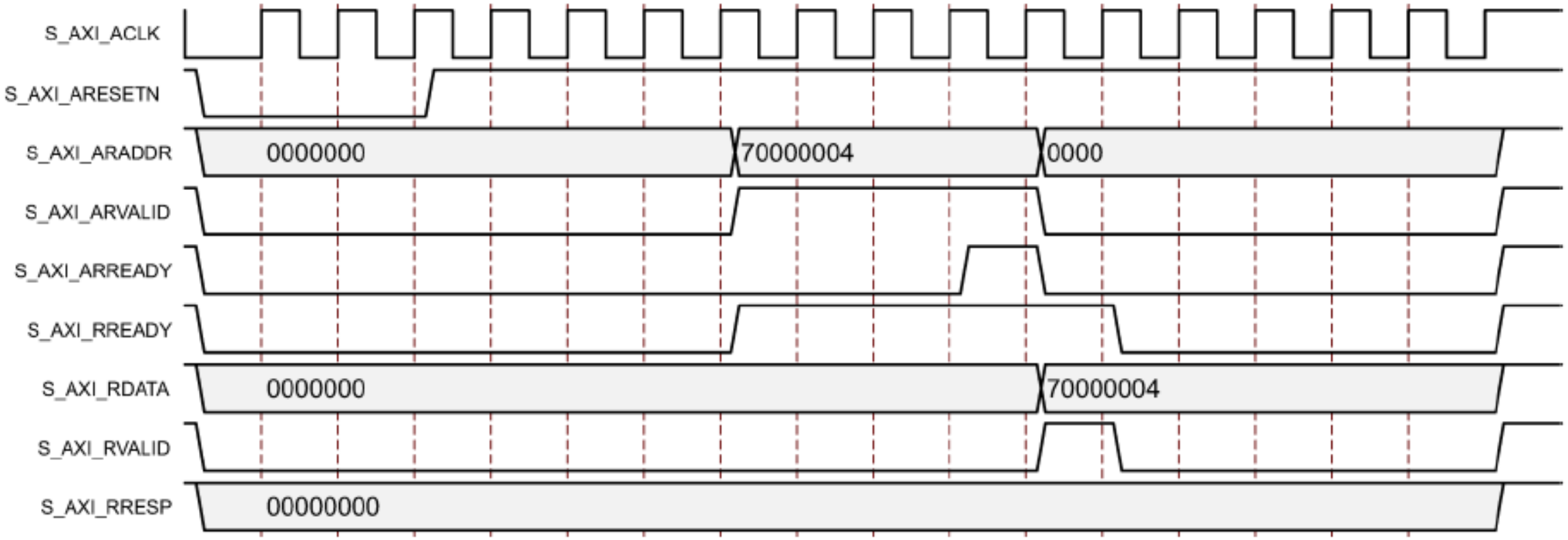
AXI LITE : Single Write transfer

INIPRO



AXI LITE : Single Read transfer

INIPRO



AXI STREAM

- **Data moves in a single direction from a master to a slave**

  Simplex communication

  Dual direction would be dual simplex (individual channels in each direction)

- **Data only**

  No address bus

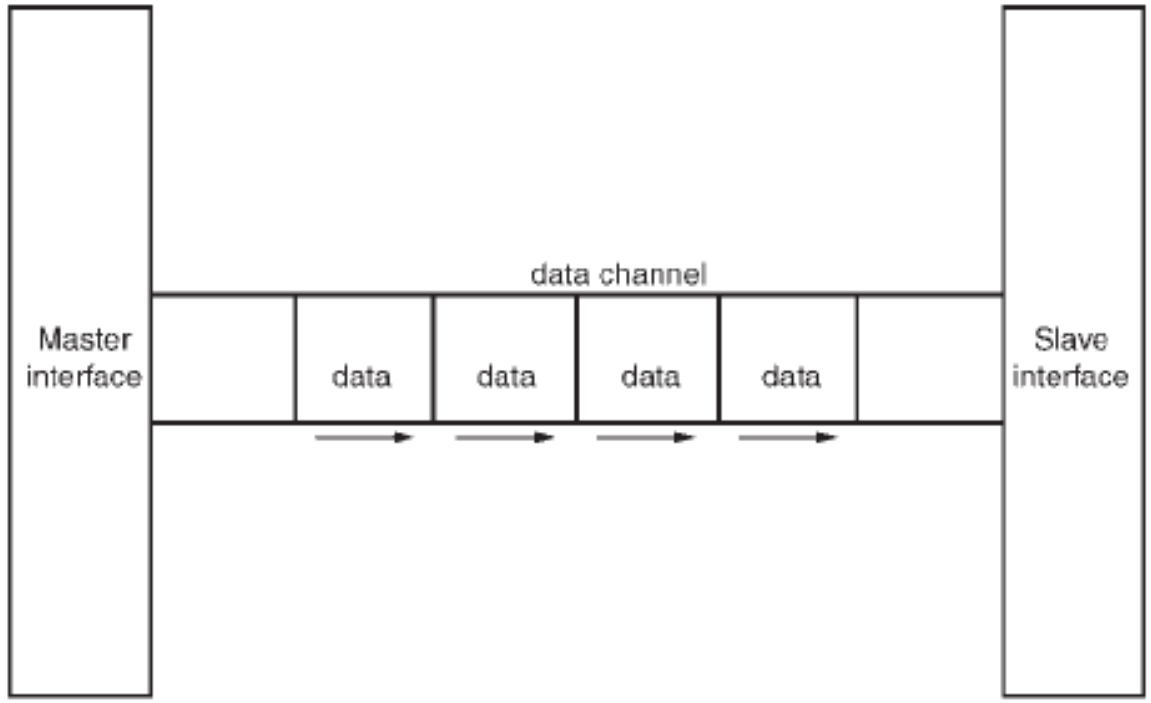
  Minimum control bus

- **Data interpretation**

  Packet or no packet

  Master and slave must have prior knowledge of data format

  Optional use of hardware signaling



AXI STREAM : signals

- **More signals defined than typically required**
- **Required signals**

  ACLK

  TDATA
- **Special, user-defined sideband signals consist of TUSER**
- **Other popular and useful signals are**

  TVALID

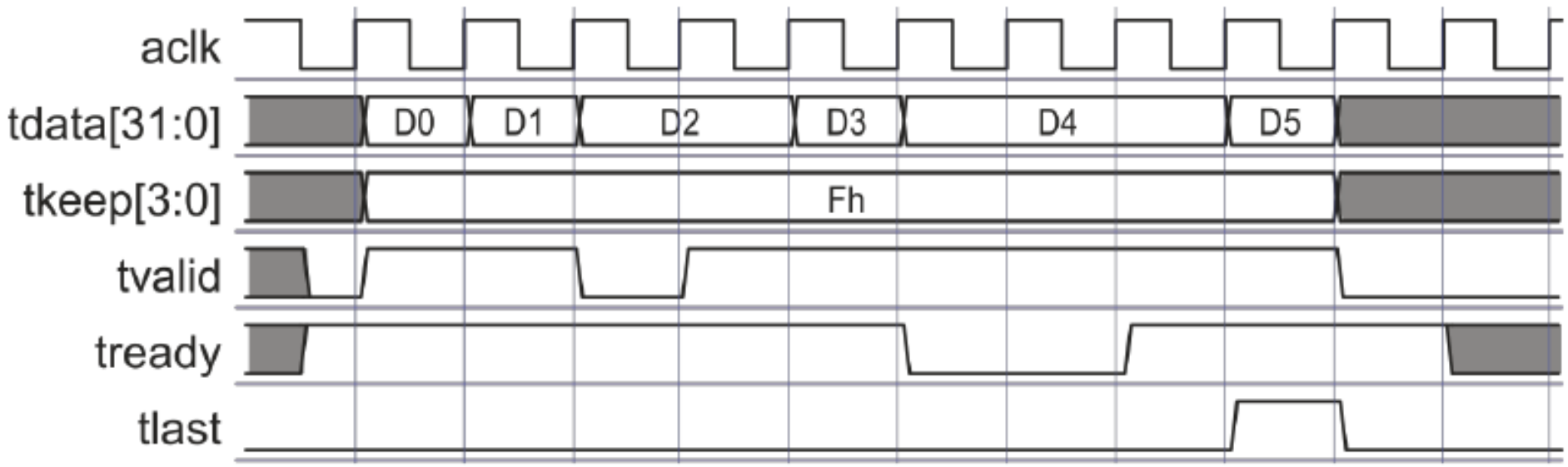
  TREADY

  TLAST

Signal	Source	Description
ACLK	Clock source	The global clock signal. All signals are sampled on the rising edge of **ACLK**.
ARESETn	Reset source	The global reset signal. **ARESETn** is active-LOW.
TVALID	Master	**TVALID** indicates that the master is driving a valid transfer. A transfer takes place when both **TVALID** and **TREADY** are asserted.
TREADY	Slave	**TREADY** indicates that the slave can accept a transfer in the current cycle.
TDATA[(8n-1):0]	Master	**TDATA** is the primary payload that is used to provide the data that is passing across the interface. The width of the data payload is an integer number of bytes.
TSTRB[(n-1):0]	Master	**TSTRB** is the byte qualifier that indicates whether the content of the associated byte of **TDATA** is processed as a data byte or a position byte.
TKEEP[(n-1):0]	Master	**TKEEP** is the byte qualifier that indicates whether the content of the associated byte of **TDATA** is processed as part of the data stream. Associated bytes that have the **TKEEP** byte qualifier deasserted are null bytes and can be removed from the data stream.
TLAST	Master	**TLAST** indicates the boundary of a packet.
TID[(i-1):0]	Master	**TID** is the data stream identifier that indicates different streams of data.
TDEST[(d-1):0]	Master	**TDEST** provides routing information for the data stream.
TUSER[(u-1):0]	Master	**TUSER** is user defined sideband information that can be transmitted alongside the data stream.

AXI STREAM : Basic protocol

INIPRO



Zynq7 PS Re-customize IP for AXI4 Lite

INIPRO

Based on Zynq7 PS Re-customize IP of embedded_linux project

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

PS-PL Configuration — Summary Report

Search:

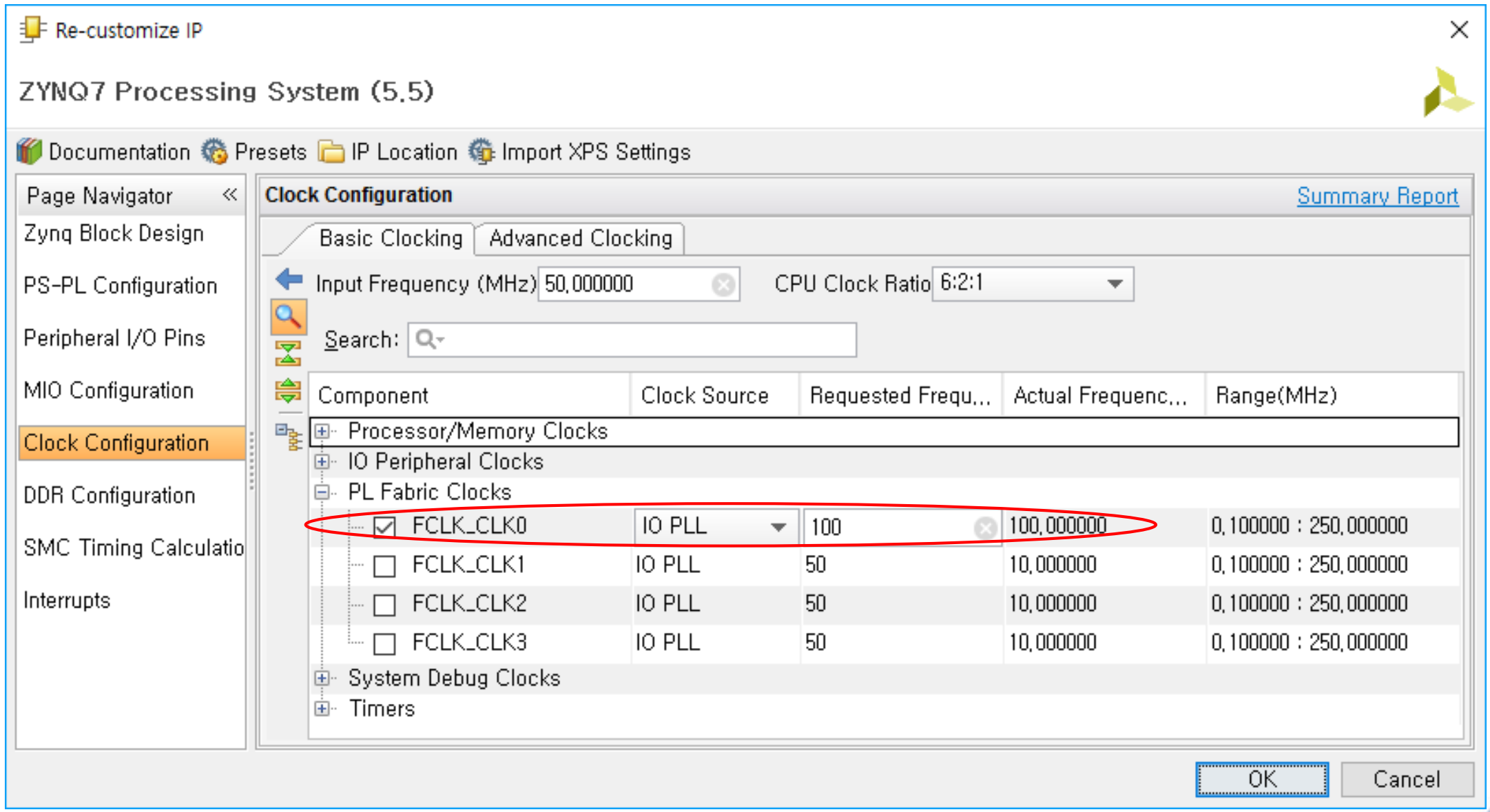
Name	Select	Description
General		
UART0 Baud Rate	115200	Baud rate is generated with internally fixed UART Ref Clock Freq=10...
UART1 Baud Rate	115200	Baud rate is generated with internally fixed UART Ref Clock Freq=10...
PL AXI idle Port	☐	Enables idle AXI signal to the PS used to indicate that there are no ...
DDR ARB bypass Port	☐	Enables DDR urgent/arb signal used to signal a critical memory sta...
PS-PL Debug interface	☐	Enables PL debug signals to PS and vice-versa
FTM Trace data interface	☐	Enables FTM Trace AXI stream interface used to capture data from PL to PS debug system
FTM Trace buffer	0	
FTM Data edge detector	0	Stores trace data in the FIFO when the data changes as marked by ...
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 be...
Include ACP transaction checker	☐	Enables ACP transaction checker.
Trace data/control signal pipeline width	8	Enables configurable number of pipeline stages on the TRACE DAT...
Power-on reset(POR) 4k timer	☐	Enables power-on reset(POR) 4k timer. By default, 64k timer is used.
Processor event interface	☐	Enables event bus which provides a low-latency and direct mecha...
Address Editor		
Enable Clock Triggers		
Enable Clock Resets		
FCLK_RESET0_N	☑	Enables general purpose reset signal 0 for PL logic
FCLK_RESET1_N	☐	Enables general purpose reset signal 1 for PL logic
FCLK_RESET2_N	☐	Enables general purpose reset signal 2 for PL logic
FCLK_RESET3_N	☐	Enables general purpose reset signal 3 for PL logic
AXI Non Secure Enablement	0	Enable AXI Non Secure Transaction
GP Master AXI Interface		
M AXI GP0 interface	☑	Enables General purpose AXI master interface 0
M AXI GP1 interface	☐	Enables General purpose AXI master interface 1
GP Slave AXI Interface		

OK | Cancel

Zynq7 PS Re-customize IP for AXI4 Lite

INIPRO

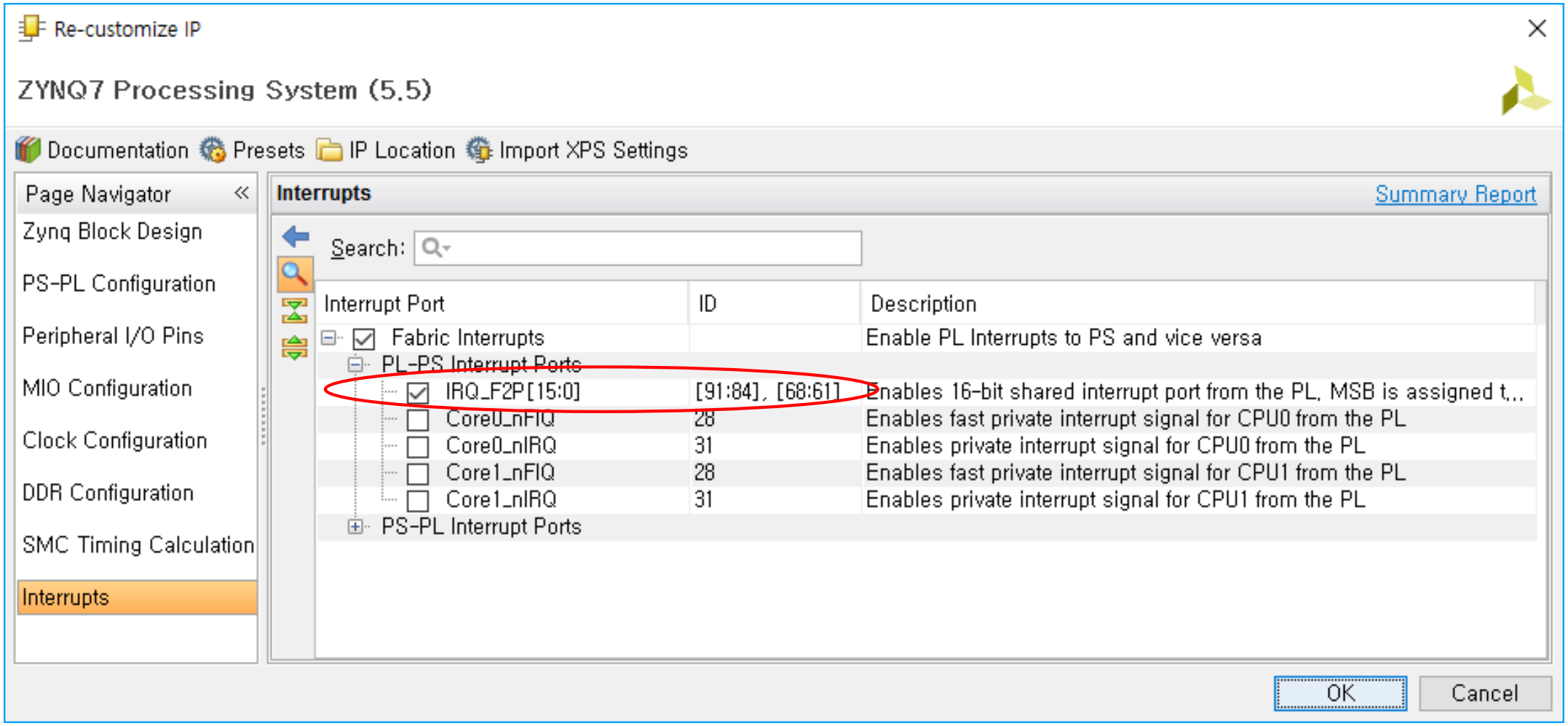
Based on Zynq7 PS Re-customize IP of embedded_linux project



Zynq7 PS Re-customize IP for AXI4 Lite

INIPRO

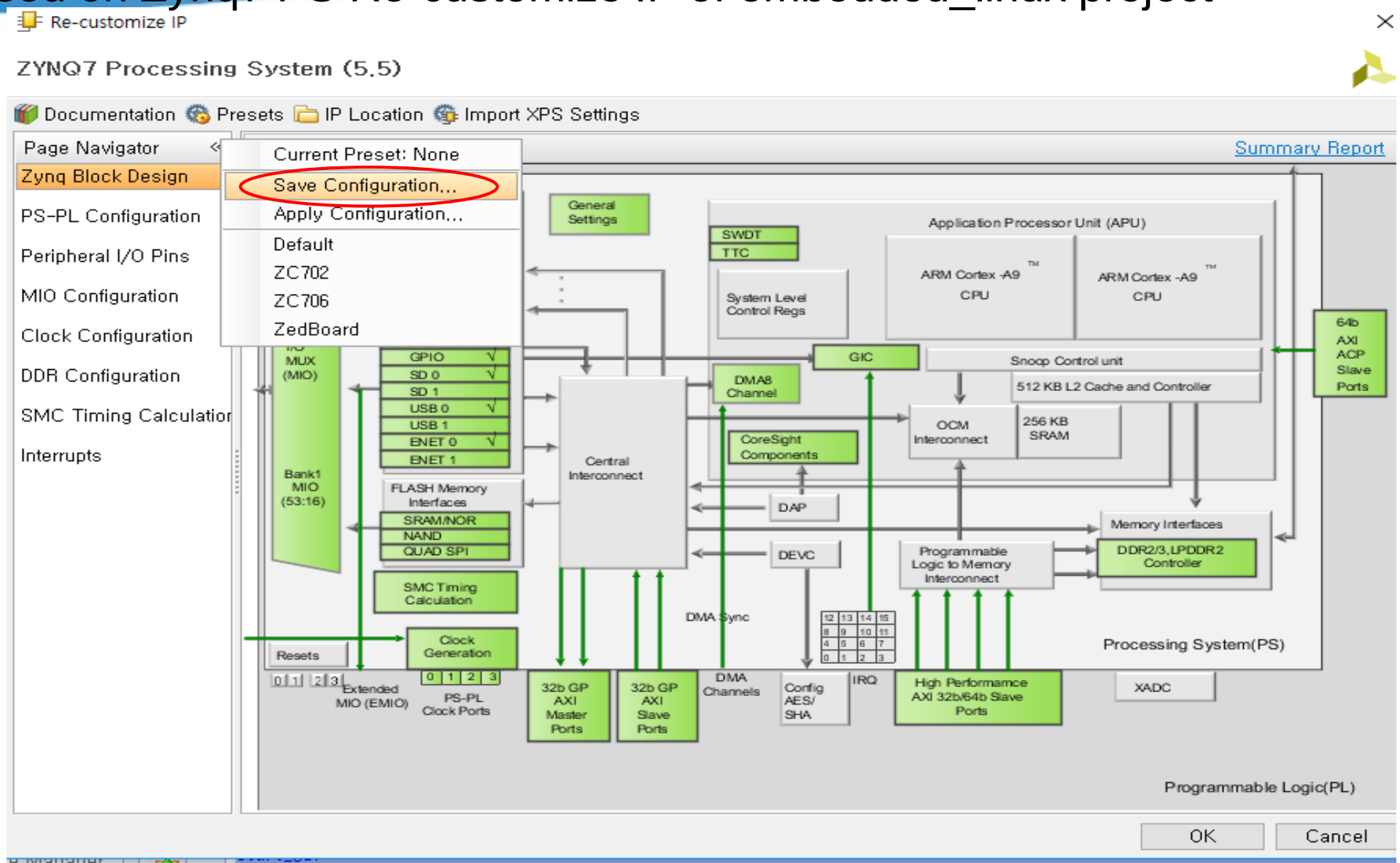
Based on Zynq7 PS Re-customize IP of embedded_linux project

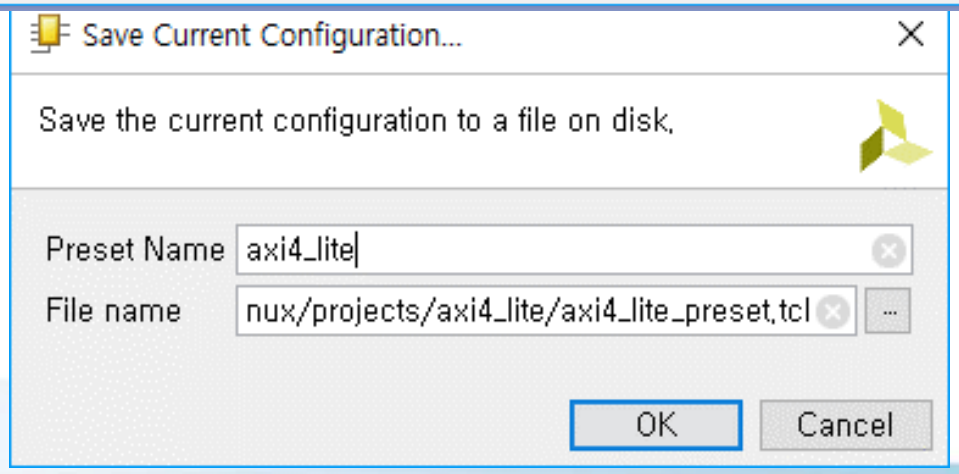


Zynq7 PS Re-customize IP for AXI4 Lite

INIPRO

Based on Zynq7 PS Re-customize IP of embedded_linux project





Project for AXI4-Lite

https://github.com/inipro/embedded_linux/projects/axi4_lite

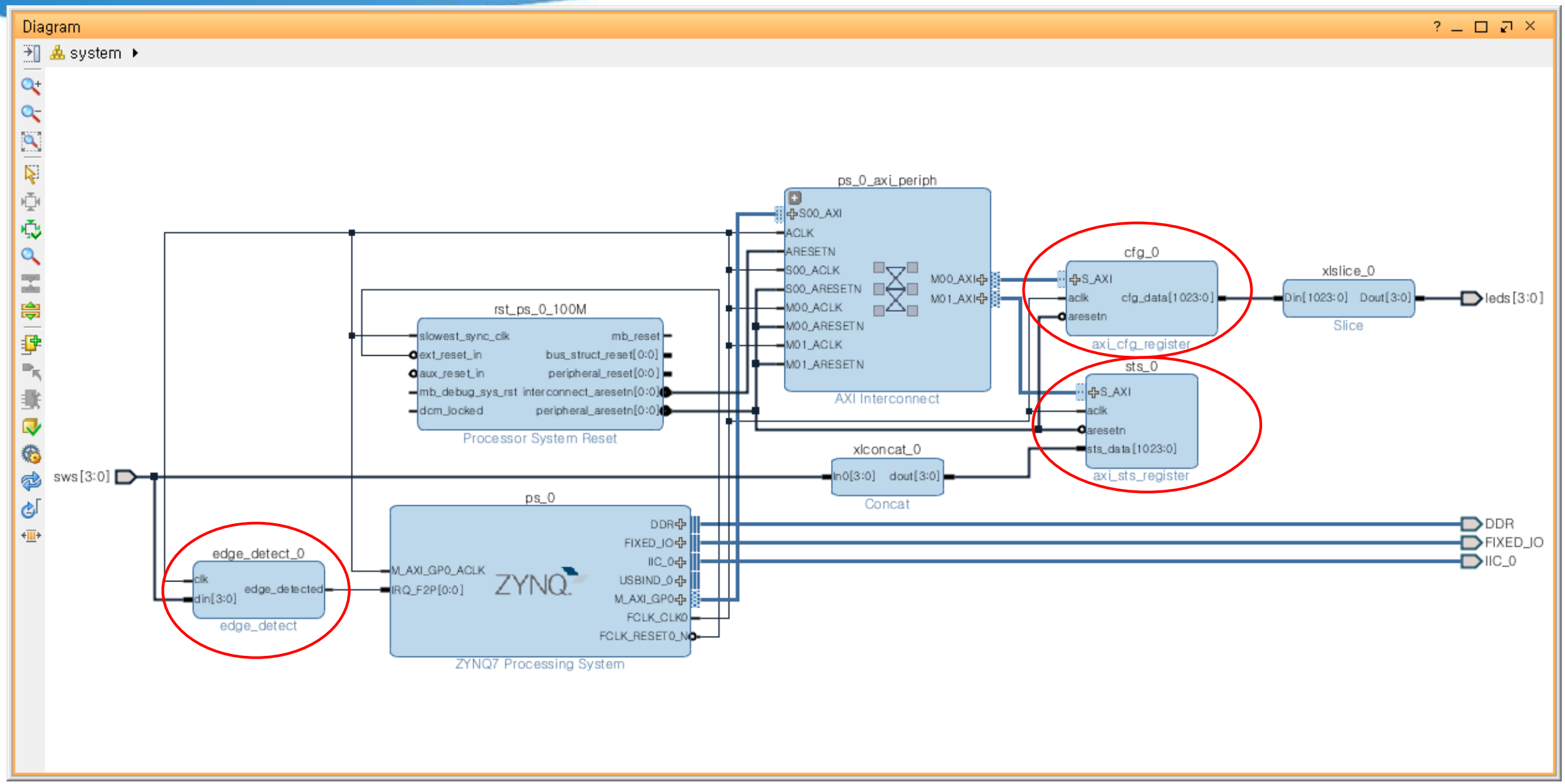
In cmd window for Vivado

```
C:\> cd C:\Users\hokim\work\embedded_linux\projects\axi4_lite
C:\> vivado  -nolog -nojournal -mode batch -source axi4_lite.tcl
```

output : axi4_lite\axi4_lite.xpr…

Project for AXI4_Lite

INIPRO



Address Editor

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
ps_0					
Data (32 address bits : 0x40000000 [1G])					
cfg_0	S_AXI	reg0	0x4000_0000	64K	0x4000_FFFF
sts_0	S_AXI	reg0	0x4001_0000	64K	0x4001_FFFF

IP for AXI4_Lite Project

C:\Users\hokim\work\embedded_linux\projects\axi4_lite\ip\edge_dectector_v1_0

→ Non AXI IP for interrupt detection from 4-bit switches

hdl\edge_detector.v

```
`timescale 1ns / 1ps

module edge_detect
#(
    parameter N = 8
)
(
    input                   clk,
    input [N-1:0]           din,
    output                  edge_detected
);

    reg [N-1:0]             tmp= 0;
    reg                     edge_detected_i = 0;

    assign edge_detected = edge_detected_i;

    always @ (posedge clk)
    begin
        tmp <= din;

        if (tmp != din)
            edge_detected_i <= 1'b1;
        else
            edge_detected_i <= 1'b0;
    end
endmodule
```

edge_detector.tcl

```
set ip_name "edge_detect"
file delete -force component.xml xgui

# Collect the names of the HDL source files that are used by this
IP here.
set file_list [list "hdl/edge_detect.v"]

# Create a new Vivado project for this IP and add the source
files.
create_project $ip_name . -force
set proj_dir [get_property directory [current_project]]
set proj_name [get_projects $ip_name]
set proj_fileset [get_filesets sources_1]
add_files -norecurse -scan_for_includes -fileset $proj_fileset
$file_list
set_property "top" "$ip_name" $proj_fileset
ipx::package_project -root_dir .

# Set the IP core information properties.
set core [ipx::current_core]
set_property vendor {hokim} $core
set_property library {ip} $core
set_property version {1.0} $core
set_property vendor_display_name {Hyunok Kim} $core
set_property company_url {https://github.com/hokim72} $core
set_property supported_families {{zynq} {Production}} $core
set_property name $ip_name $core
set_property display_name $ip_name $core
set_property description $ip_name $core

# Generate the XGUI files to accompany this IP core.
ipx::create_xgui_files $core
```

IP for AXI4_Lite Project

```
C:\> cd C:\Users\hokim\work\embedded_linux\projects\axi4_lite\ip\edge_detector_v1_0
C:\> vivado -nolog -nojournal -mode batch -source edge_detector.tcl
```

C:\Users\hokim\work\embedded_linux\projects\axi4_lite\ip\axi_cfg_register_v1_0
→ AXI4-Lite IP for control register which are used for 4-bit leds here

hdl\axi_cfg_register.v

```
`timescale 1 ns / 1 ps

module axi_cfg_register #
(
  parameter integer CFG_DATA_WIDTH = 1024,
  parameter integer AXI_DATA_WIDTH = 32,
  parameter integer AXI_ADDR_WIDTH = 32
)
(
  // System signals
  input  wire                      aclk,
  input  wire                      aresetn,

  // Configuration bits
  output wire [CFG_DATA_WIDTH-1:0]   cfg_data,

  // Slave side
  input  wire [AXI_ADDR_WIDTH-1:0]   s_axi_awaddr,  // AXI4-Lite slave: Write address
  input  wire                        s_axi_awvalid, // AXI4-Lite slave: Write address valid
  output wire                        s_axi_awready, // AXI4-Lite slave: Write address ready
  input  wire [AXI_DATA_WIDTH-1:0]   s_axi_wdata,   // AXI4-Lite slave: Write data
  input  wire [AXI_DATA_WIDTH/8-1:0] s_axi_wstrb,   // AXI4-Lite slave: Write strobe
  input  wire                        s_axi_wvalid,  // AXI4-Lite slave: Write data valid
```

IP for AXI4_Lite Project

hdl\axi_cfg_register.v

```
  output wire                    s_axi_wready,  // AXI4-Lite slave: Write data ready
  output wire [1:0]              s_axi_bresp,   // AXI4-Lite slave: Write response
  output wire                    s_axi_bvalid,  // AXI4-Lite slave: Write response valid
  input  wire                    s_axi_bready,  // AXI4-Lite slave: Write response ready
  input  wire [AXI_ADDR_WIDTH-1:0]   s_axi_araddr,  // AXI4-Lite slave: Read address
  input  wire                    s_axi_arvalid, // AXI4-Lite slave: Read address valid
  output wire                    s_axi_arready, // AXI4-Lite slave: Read address ready
  output wire [AXI_DATA_WIDTH-1:0]   s_axi_rdata,   // AXI4-Lite slave: Read data
  output wire [1:0]              s_axi_rresp,   // AXI4-Lite slave: Read data response
  output wire                    s_axi_rvalid,  // AXI4-Lite slave: Read data valid
  input  wire                    s_axi_rready   // AXI4-Lite slave: Read data ready
);

 function integer clogb2 (input integer value);
   for(clogb2 = 0; value > 0; clogb2 = clogb2 + 1) value = value >> 1;
 endfunction

 localparam integer ADDR_LSB = clogb2(AXI_DATA_WIDTH/8 - 1);
 localparam integer CFG_SIZE = CFG_DATA_WIDTH/AXI_DATA_WIDTH;
 localparam integer CFG_WIDTH = CFG_SIZE > 1 ? clogb2(CFG_SIZE-1) : 1;

 reg int_bvalid_reg, int_bvalid_next;

 reg int_rvalid_reg, int_rvalid_next;
 reg [AXI_DATA_WIDTH-1:0] int_rdata_reg, int_rdata_next;

 wire [AXI_DATA_WIDTH-1:0] int_data_mux [CFG_SIZE-1:0];
 wire [CFG_DATA_WIDTH-1:0] int_data_wire;
 wire [CFG_SIZE-1:0] int_ce_wire;
 wire int_wvalid_wire;

 genvar j, k;

 assign int_wvalid_wire = s_axi_awvalid & s_axi_wvalid;
```

IP for AXI4_Lite Project

INIPRO

hdl\axi_cfg_register.v

```
generate
  for(j = 0; j < CFG_SIZE; j = j + 1)
  begin : WORDS
    assign int_data_mux[j] = int_data_wire[j*AXI_DATA_WIDTH+AXI_DATA_WIDTH-
1:j*AXI_DATA_WIDTH];
    assign int_ce_wire[j] = int_wvalid_wire & (s_axi_awaddr[ADDR_LSB+CFG_WIDTH-
1:ADDR_LSB] == j);
    for(k = 0; k < AXI_DATA_WIDTH; k = k + 1)
    begin : BITS
      FDRE #(
        .INIT(1'b0)
      ) FDRE_inst (
        .CE(int_ce_wire[j] & s_axi_wstrb[k/8]),
        .C(aclk),
        .R(~aresetn),
        .D(s_axi_wdata[k]),
        .Q(int_data_wire[j*AXI_DATA_WIDTH + k])
      );
    end
  end
endgenerate

always @(posedge aclk)
begin
  if(~aresetn)
  begin
    int_bvalid_reg <= 1'b0;
    int_rvalid_reg <= 1'b0;
    int_rdata_reg <= {(AXI_DATA_WIDTH){1'b0}};
  end
  else
  begin
    int_bvalid_reg <= int_bvalid_next;
    int_rvalid_reg <= int_rvalid_next;
    int_rdata_reg <= int_rdata_next;
  end
end
```

IP for AXI4_Lite Project

hdl\axi_cfg_register.v

```
always @*
 begin
  int_bvalid_next = int_bvalid_reg;

  if(int_wvalid_wire)
  begin
    int_bvalid_next = 1'b1;
  end

  if(s_axi_bready & int_bvalid_reg)
  begin
    int_bvalid_next = 1'b0;
  end
end

always @*
begin
  int_rvalid_next = int_rvalid_reg;
  int_rdata_next = int_rdata_reg;

  if(s_axi_arvalid)
  begin
    int_rvalid_next = 1'b1;
    int_rdata_next = int_data_mux[s_axi_araddr[ADDR_LSB+CFG_WIDTH-1:ADDR_LSB]];
  end

  if(s_axi_rready & int_rvalid_reg)
  begin
    int_rvalid_next = 1'b0;
  end
end
```

IP for AXI4_Lite Project

hdl\axi_cfg_register.v

```
 assign cfg_data = int_data_wire;

  assign s_axi_bresp = 2'd0;
  assign s_axi_rresp = 2'd0;

  assign s_axi_awready = int_wvalid_wire;
  assign s_axi_wready = int_wvalid_wire;
  assign s_axi_bvalid = int_bvalid_reg;
  assign s_axi_arready = 1'b1;
  assign s_axi_rdata = int_rdata_reg;
  assign s_axi_rvalid = int_rvalid_reg;

endmodule
```

hdl\axi_cfg_register.tcl

```
set ip_name "axi_cfg_register"
file delete -force component.xml xgui

# Collect the names of the HDL source files that are used by this IP here.
set file_list [list "hdl/axi_cfg_register.v"]

# Create a new Vivado project for this IP and add the source files.
create_project $ip_name . -force
set proj_dir [get_property directory [current_project]]
set proj_name [get_projects $ip_name]
set proj_fileset [get_filesets sources_1]
add_files -norecurse -scan_for_includes -fileset $proj_fileset $file_list
set_property "top" "$ip_name" $proj_fileset
ipx::package_project -root_dir .
```

IP for AXI4_Lite Project

hdl\axi_cfg_register.tcl

```
# Set the IP core information properties.
set core [ipx::current_core]
set_property vendor {hokim} $core
set_property library {ip} $core
set_property version {1.0} $core
set_property vendor_display_name {Hyunok Kim} $core
set_property company_url {https://github.com/hokim72} $core
set_property supported_families {{zynq} {Production}} $core
set_property name $ip_name $core
set_property display_name $ip_name $core
set_property description $ip_name $core


set parameter [ipx::get_user_parameters AXI_DATA_WIDTH -of_objects $core]
set_property DISPLAY_NAME {AXI DATA WIDTH} $parameter
set_property DESCRIPTION {Width of the AXI data bus.} $parameter

set parameter [ipgui::get_guiparamspec -name AXI_DATA_WIDTH -component $core]
set_property DISPLAY_NAME {AXI DATA WIDTH} $parameter
set_property TOOLTIP {Width of the AXI data bus.} $parameter

set parameter [ipx::get_user_parameters AXI_ADDR_WIDTH -of_objects $core]
set_property DISPLAY_NAME {AXI ADDR WIDTH} $parameter
set_property DESCRIPTION {Width of the AXI address bus.} $parameter

set parameter [ipgui::get_guiparamspec -name AXI_ADDR_WIDTH -component $core]
set_property DISPLAY_NAME {AXI ADDR WIDTH} $parameter
set_property TOOLTIP {Width of the AXI address bus.} $parameter

set parameter [ipx::get_user_parameters CFG_DATA_WIDTH -of_objects $core]
set_property DISPLAY_NAME {CFG DATA WIDTH} $parameter
set_property DESCRIPTION {Width of the configuration data.} $parameter

set parameter [ipgui::get_guiparamspec -name CFG_DATA_WIDTH -component $core]
set_property DISPLAY_NAME {CFG DATA WIDTH} $parameter
set_property TOOLTIP {Width of the configuration data.} $parameter
```

IP for AXI4_Lite Project

INIPRO

hdl\axi_cfg_register.tcl

```
set bus [ipx::get_bus_interfaces -of_objects $core s_axi]
set_property NAME S_AXI $bus
set_property INTERFACE_MODE slave $bus

set bus [ipx::get_bus_interfaces aclk]
set parameter [ipx::get_bus_parameters -of_objects $bus ASSOCIATED_BUSIF]
set_property VALUE S_AXI $parameter

# Generate the XGUI files to accompany this IP core.
ipx::create_xgui_files $core

# Save the IP core.
ipx::save_core $core

# Close the current project.
close_project

file delete -force $ip_name.cache $ip_name.hw $ip_name.ip_user_files $ip_name.xpr
```

```
C:\> cd C:\Users\hokim\work\embedded_linux\projects\axi4_lite\ip\axi_cfg_register_v1_0
C:\> vivado -nolog -nojournal -mode batch -source ax_cfg_register.tcl
```

IP for AXI4_Lite Project

C:\Users\hokim\work\embedded_linux\projects\axi4_lite\ip\axi_sts_register_v1_0
→ AXI4-Lite IP for status register which are used for 4-bit switches here

hdl\axi_sts_register.v

```
`timescale 1 ns / 1 ps

module axi_sts_register #
(
  parameter integer STS_DATA_WIDTH = 1024,
  parameter integer AXI_DATA_WIDTH = 32,
  parameter integer AXI_ADDR_WIDTH = 32
)
(
  // System signals
  input  wire                    aclk,
  input  wire                    aresetn,

  // Status bits
  input  wire [STS_DATA_WIDTH-1:0] sts_data,

  // Slave side
  input  wire [AXI_ADDR_WIDTH-1:0] s_axi_araddr,  // AXI4-Lite slave: Read address
  input  wire                    s_axi_arvalid, // AXI4-Lite slave: Read address valid
  output wire                     s_axi_arready, // AXI4-Lite slave: Read address ready
  output wire [AXI_DATA_WIDTH-1:0] s_axi_rdata,   // AXI4-Lite slave: Read data
  output wire [1:0]                s_axi_rresp,   // AXI4-Lite slave: Read data response
  output wire                     s_axi_rvalid,  // AXI4-Lite slave: Read data valid
  input  wire                    s_axi_rready   // AXI4-Lite slave: Read data ready
);

  function integer clogb2 (input integer value);
    for(clogb2 = 0; value > 0; clogb2 = clogb2 + 1) value = value >> 1;
  endfunction
```

IP for AXI4_Lite Project

hdl\axi_sts_register.v

```
  localparam integer ADDR_LSB = clogb2(AXI_DATA_WIDTH/8 - 1);
  localparam integer STS_SIZE = STS_DATA_WIDTH/AXI_DATA_WIDTH;
  localparam integer STS_WIDTH = STS_SIZE > 1 ? clogb2(STS_SIZE-1) : 1;

  reg int_rvalid_reg, int_rvalid_next;
  reg [AXI_DATA_WIDTH-1:0] int_rdata_reg, int_rdata_next;

  wire [AXI_DATA_WIDTH-1:0] int_data_mux [STS_SIZE-1:0];

  genvar j, k;

  generate
    for(j = 0; j < STS_SIZE; j = j + 1)
    begin : WORDS
      assign int_data_mux[j] = sts_data[j*AXI_DATA_WIDTH+AXI_DATA_WIDTH-
1:j*AXI_DATA_WIDTH];
    end
  endgenerate

  always @(posedge aclk)
  begin
    if(~aresetn)
    begin
      int_rvalid_reg <= 1'b0;
      int_rdata_reg <= {(AXI_DATA_WIDTH){1'b0}};
    end
    else
    begin
      int_rvalid_reg <= int_rvalid_next;
      int_rdata_reg <= int_rdata_next;
    end
  end
```

IP for AXI4_Lite Project

hdl\axi_sts_register.v

```
 always @*
  begin
    int_rvalid_next = int_rvalid_reg;
    int_rdata_next = int_rdata_reg;

    if(s_axi_arvalid)
    begin
      int_rvalid_next = 1'b1;
      int_rdata_next = int_data_mux[s_axi_araddr[ADDR_LSB+STS_WIDTH-
1:ADDR_LSB]];
    end

    if(s_axi_rready & int_rvalid_reg)
    begin
      int_rvalid_next = 1'b0;
    end
  end

  assign s_axi_rresp = 2'd0;

  assign s_axi_arready = 1'b1;
  assign s_axi_rdata = int_rdata_reg;
  assign s_axi_rvalid = int_rvalid_reg;

endmodule
```

IP for AXI4_Lite Project

INIPRO

axi_sts_register.tcl

```
set ip_name "axi_sts_register"
file delete -force component.xml xgui

# Collect the names of the HDL source files that are used by this IP here.
set file_list [list "hdl/axi_sts_register.v"]

# Create a new Vivado project for this IP and add the source files.
create_project $ip_name . -force
set proj_dir [get_property directory [current_project]]
set proj_name [get_projects $ip_name]
set proj_fileset [get_filesets sources_1]
add_files -norecurse -scan_for_includes -fileset $proj_fileset $file_list
set_property "top" "$ip_name" $proj_fileset
ipx::package_project -root_dir .

# Set the IP core information properties.
set core [ipx::current_core]
set_property vendor {hokim} $core
set_property library {ip} $core
set_property version {1.0} $core
set_property vendor_display_name {Hyunok Kim} $core
set_property company_url {https://github.com/hokim72} $core
set_property supported_families {{zynq} {Production}} $core
set_property name $ip_name $core
set_property display_name $ip_name $core
set_property description $ip_name $core
```

IP for AXI4_Lite Project

axi_sts_register.tcl

```
set parameter [ipx::get_user_parameters AXI_DATA_WIDTH -of_objects $core]
set_property DISPLAY_NAME {AXI DATA WIDTH} $parameter
set_property DESCRIPTION {Width of the AXI data bus.} $parameter

set parameter [ipgui::get_guiparamspec -name AXI_DATA_WIDTH -component $core]
set_property DISPLAY_NAME {AXI DATA WIDTH} $parameter
set_property TOOLTIP {Width of the AXI data bus.} $parameter

set parameter [ipx::get_user_parameters AXI_ADDR_WIDTH -of_objects $core]
set_property DISPLAY_NAME {AXI ADDR WIDTH} $parameter
set_property DESCRIPTION {Width of the AXI address bus.} $parameter

set parameter [ipgui::get_guiparamspec -name AXI_ADDR_WIDTH -component $core]
set_property DISPLAY_NAME {AXI ADDR WIDTH} $parameter
set_property TOOLTIP {Width of the AXI address bus.} $parameter

set parameter [ipx::get_user_parameters STS_DATA_WIDTH -of_objects $core]
set_property DISPLAY_NAME {STS DATA WIDTH} $parameter
set_property DESCRIPTION {Width of the status data.} $parameter

set parameter [ipgui::get_guiparamspec -name STS_DATA_WIDTH -component $core]
set_property DISPLAY_NAME {STS DATA WIDTH} $parameter
set_property TOOLTIP {Width of the status data.} $parameter
```

IP for AXI4_Lite Project

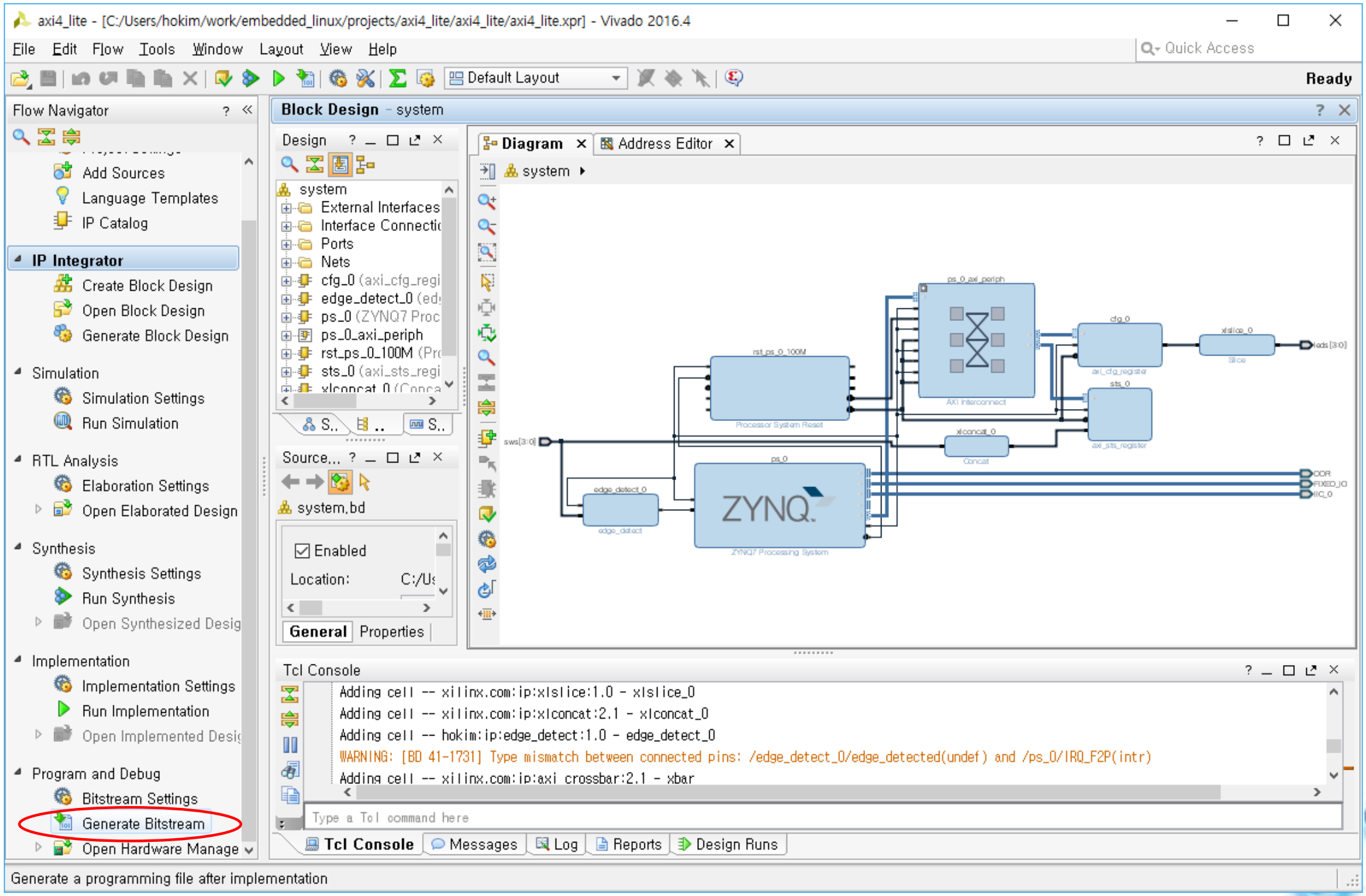
axi_sts_register.tcl

```
set bus [ipx::get_bus_interfaces -of_objects $core s_axi]
set_property NAME S_AXI $bus
set_property INTERFACE_MODE slave $bus

set bus [ipx::get_bus_interfaces aclk]
set parameter [ipx::get_bus_parameters -of_objects $bus ASSOCIATED_BUSIF]
set_property VALUE S_AXI $parameter

# Generate the XGUI files to accompany this IP core.
ipx::create_xgui_files $core

# Save the IP core.
ipx::save_core $core

# Close the current project.
close_project
```

```
C:\> cd C:\Users\hokim\work\embedded_linux\projects\axi4_lite\ip\axi_sts_register_v1_0
C:\> vivado -nolog -nojournal -mode batch -source ax_sts_register.tcl
```

Bit Generation for AXI4_Lite

INIPRO



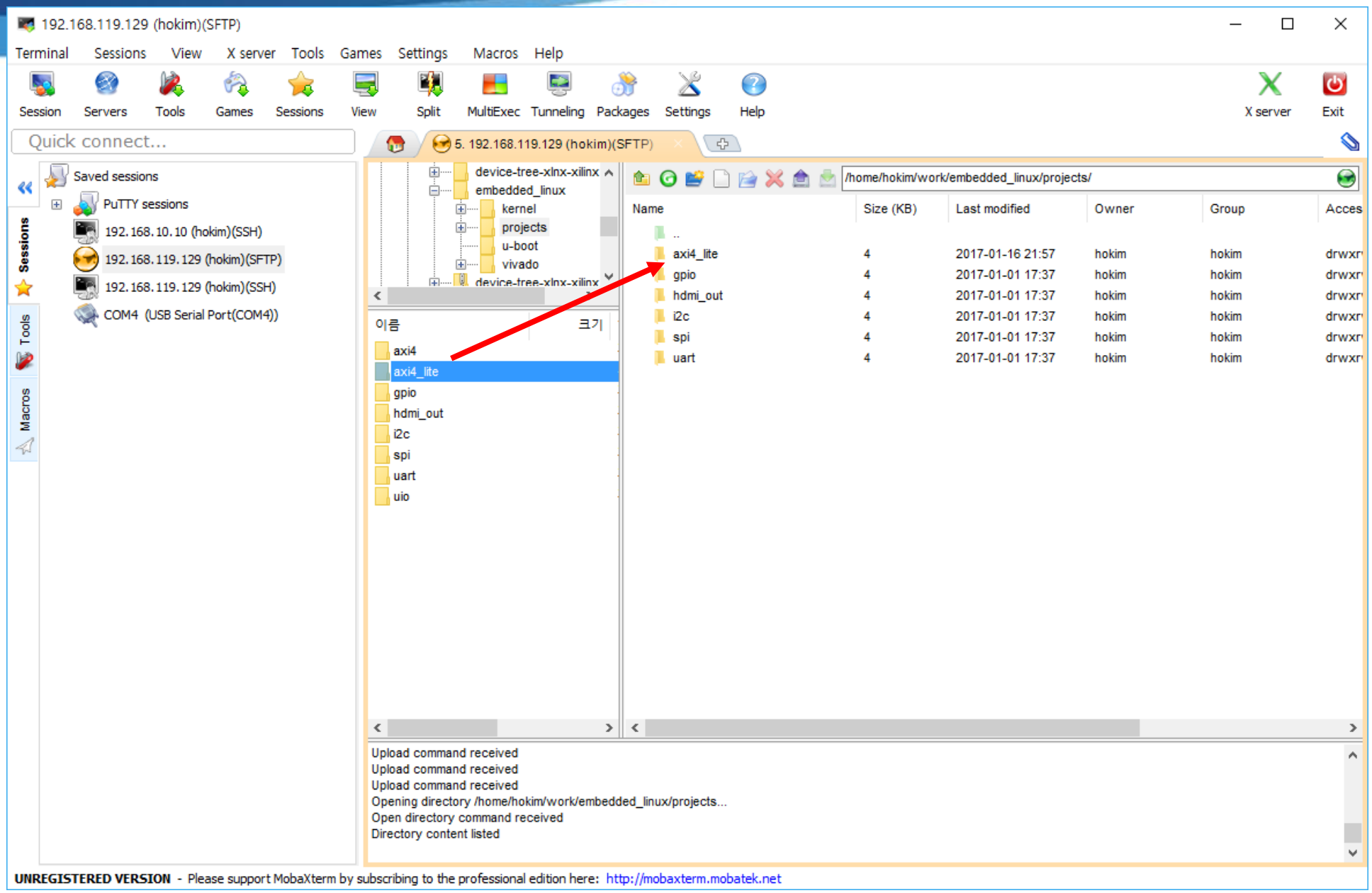
Boot.bin Build/DT Generation for AXI4_Lite

C:\Users\hokim\work\embedded_linux\projects\axi4_lite\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\axi4_lite
C:\> all
```

Device Tree Compile for AXI4_Lite




```
$ cd ~/work/embedded_linux/projects/axi4_lite
$ cp axi4_lite/axi4_lite.tree/system.dts  system_axi4_lite.dts
$ nano system_axi4_lite.dts
```

Device Tree Compile for AXI4_Lite

system_axi4_lite.dts

```
/dts-v1/;
/include/ "zynq-7000.dtsi"
/include/ "pl.dtsi“
…….
    chosen {
-       bootargs = "console=ttyPS0,115200
root=/dev/mmcblk0p2 ro rootfstype=ext4 earlyprintk rootwait";
+       bootargs = "console=ttyPS0,115200
root=/dev/mmcblk0p2 ro rootfstype=ext4 earlyprintk rootwait
uio_pdrv_genirq.of_id=generic-uio";
    };

…….
&clkc {
    fclk-enable = <0x1>;
    ps-clk-frequency = <50000000>;
};
+&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>;
+};

+&cfg_0 {
+               compatible = "generic-uio";
+};
+&sts_0 {
+               compatible = "generic-uio";
+               interrupt-parent =<&intc>;
+               interrupts = <0x0 0x1d 0x1>;
+};
```

Device Tree Compile for AXI4_Lite

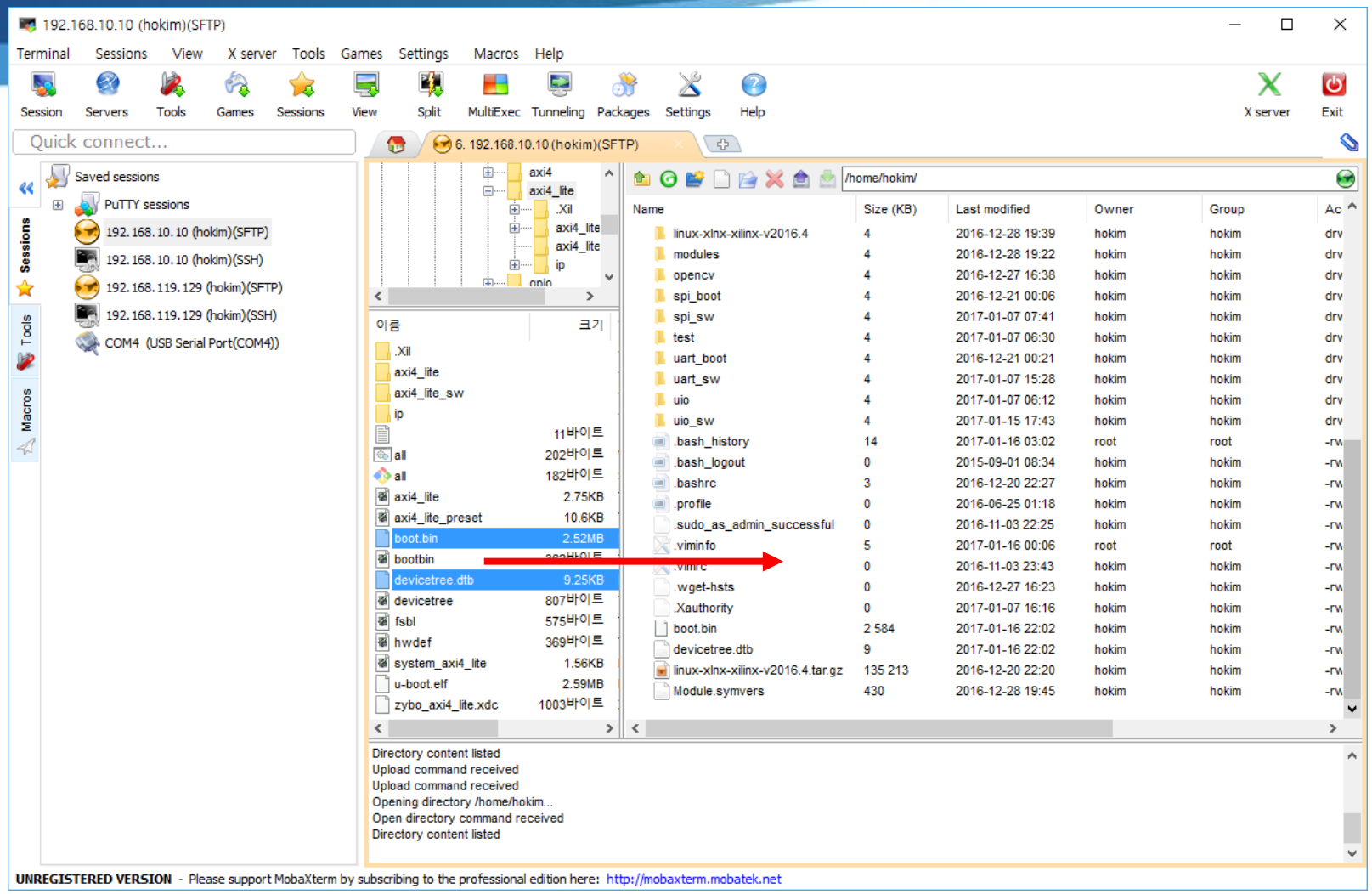
INIPRO

axi4_lite/axi4_lite.tree/*pl.dtsi*

```
/ {
  amba_pl: amba_pl {
    #address-cells = <1>;
    #size-cells = <1>;
    compatible = "simple-bus";
    ranges ;
    cfg_0: axi_cfg_register@40000000 {
      compatible = "xlnx,axi-cfg-register-1.0";
      reg = <0x40000000 0x10000>;
    };
    sts_0: axi_sts_register@40010000 {
      compatible = "xlnx,axi-sts-register-1.0";
      reg = <0x40010000 0x10000>;
    };
  };
};
```

```
$ dtc -O dtb -I dts -i axi4_lite/axi4_lite.tree/ -o devicetree.dtb system_axi4_lite.dts
```

Update boot.bin, devicetree.dtb for AXI4



login into zybo

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

AXI4_Lite sw & test

```
hokim@zybo:~$ ls /sys/class/uio
uio0  uio1
hokim@zybo:~$ cat /sys/class/uio/uio0/name
axi_cfg_register
hokim@zybo:~$ cat /sys/class/uio/uio1/name
axi_sts_register
```

axi4_lite_test.c

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/mman.h>
#include <fcntl.h>

#define DATA_OFFSET		0x0			/* Data register for
cfg and sts */

void uio_write(void *uio_base, uint32_t offset, uint32_t value)
{
		*((volatile uint32_t *)(uio_base + offset)) = value;
}

uint32_t uio_read(void *uio_base, uint32_t offset)
{
		return *((volatile uint32_t *)(uio_base + offset));
}

uint32_t get_memory_size(char *sysfs_path_file)
{
		FILE *size_fp;
		uint32_t size;

		// open the file that describes the memory range size that is based
on the
		// reg property of the node in the device tree
		size_fp = fopen(sysfs_path_file, "r");

		if (!size_fp) {
				printf("unable to open the uio size file\n");
				exit(-1);
		}
```

AXI4_Lite sw & test

INIPRO

axi4_lite_test.c

```
		// get the size which is an ASCII string such as 0xXXXXXXXX and
then be stop
		// using the file
		fscanf(size_fp, "0x%08X", &size);
		fclose(size_fp);

		return size;
}

void wait_for_interrupt(int fd)
{
		int pending = 0;
		int reenable = 1;
		uint32_t reg;

		// block on the file waiting for an interrupt
		read(fd, (void *)&pending, sizeof(int));
		//printf("Interrupt!\n");

		// re-enable the interrupt in the interrupt controller thru
		// the UIO subsystem now that it's been handled

		write(fd, (void *)&reenable, sizeof(int));
}

int main()
{
		int cfg0_fd, sts0_fd;
		void *cfg0;
		void *sts0;
		uint32_t cfg0_size, sts0_size;
		int reenable = 1;
		uint32_t value;
```

AXI4_Lite sw & test

INIPRO

axi4_lite_test.c

```
		if ((cfg0_fd = open("/dev/uio0", O_RDWR)) < 0) {
			perror("open cfg0");
			return -1;
		}

		if ((sts0_fd = open("/dev/uio1", O_RDWR)) < 0) {
			perror("open sts0");
			return -1;
		}

		cfg0_size =
get_memory_size("/sys/class/uio/uio0/maps/map0/size");
		sts0_size =
get_memory_size("/sys/class/uio/uio1/maps/map0/size");

		cfg0 =  mmap(NULL, cfg0_size, PROT_READ|PROT_WRITE,
MAP_SHARED, cfg0_fd, 0);

		if (cfg0 == MAP_FAILED) {
			perror("cfg0 mmap call failure");
			return -1;
		}

		sts0 =  mmap(NULL, sts0_size, PROT_READ|PROT_WRITE,
MAP_SHARED, sts0_fd, 0);

		if (sts0 == MAP_FAILED) {
			perror("sts0 mmap call failure");
			return -1;
		}

		write(sts0_fd, (void *)&reenable, sizeof(int));
```

AXI4_Lite sw & test

axi4_lite_test.c

```
                while (1)
                {
                        wait_for_interrupt(sts0_fd);
                        value = uio_read(sts0, DATA_OFFSET);
                        printf("sws = 0x%0X\n", value);
                        uio_write(cfg0, DATA_OFFSET, value);
                }

                munmap(cfg0, cfg0_size);
                munmap(sts0, sts0_size);

                return 0;
}
```

```
$ cd ~/axi4_lite_sw
$ ls
$ CMakeLists.txt axi4_lite_test.c
$ mkdir build
$ cd build
$ cmake
$ make
$ sudo ./axi4_lite_test
```

Zynq7 PS Re-customize IP for AXI4

INIPRO

Based on Zynq7 PS Re-customize IP of embedded_linux project

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

PS-PL Configuration — Summary Report

Search:

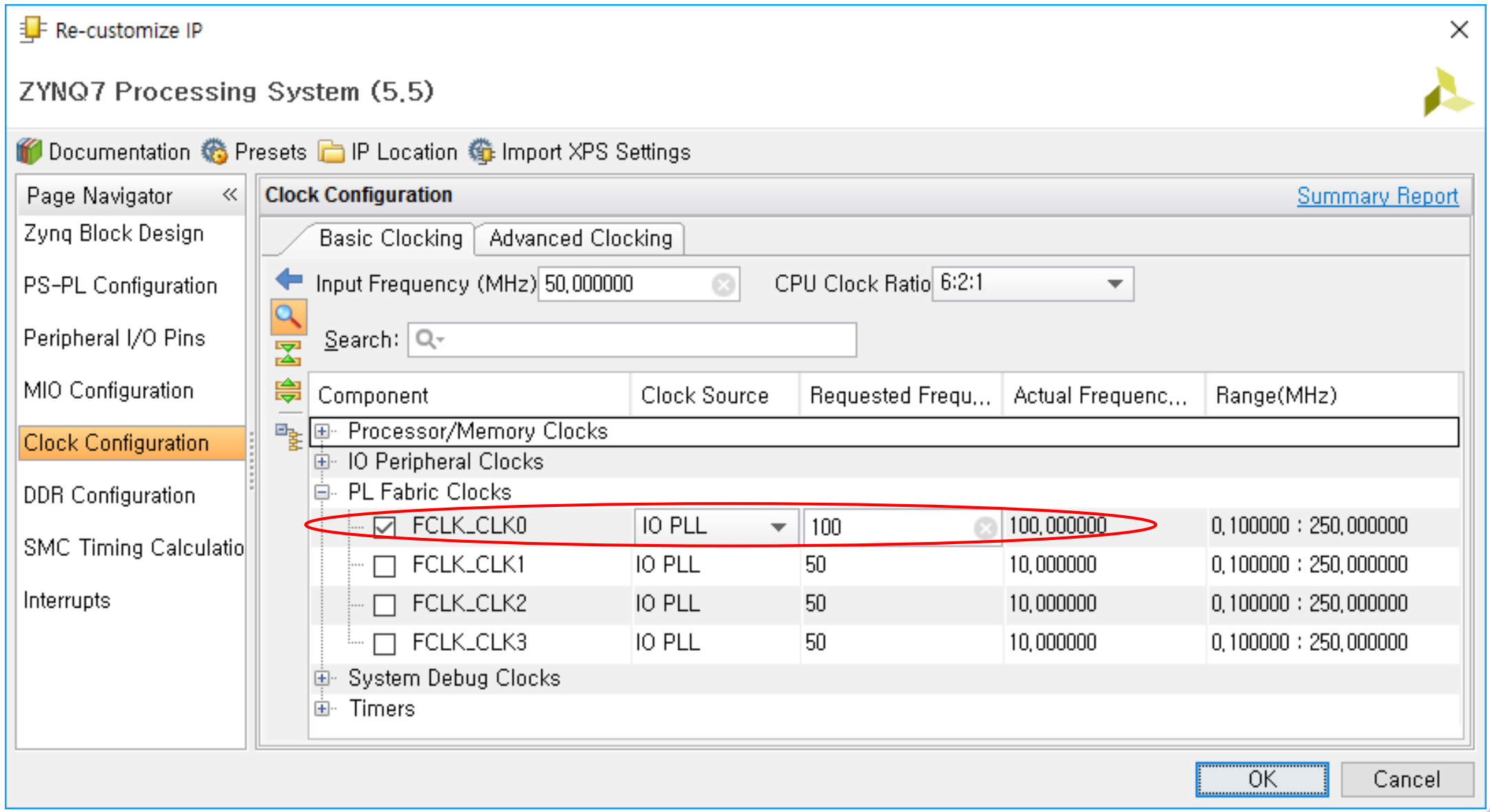
Name	Select	Description
FTM Trace buffer clock delay	12	Number of clock cycles interval for a trace data output from FIFI
Include ACP transaction checker	☐	Enables ACP transaction checker.
Trace data/control signal pipeline width	8	Enables configurable number of pipeline stages on the TRACE
Power-on reset(POR) 4k timer	☐	Enables power-on reset(POR) 4k timer. By default, 64k timer is
Processor event interface	☐	Enables event bus which provides a low-latency and direct me
Address Editor		
Enable Clock Triggers		
Enable Clock Resets		
FCLK_RESET0_N	☑	Enables general purpose reset signal 0 for PL logic
FCLK_RESET1_N	☐	Enables general purpose reset signal 1 for PL logic
FCLK_RESET2_N	☐	Enables general purpose reset signal 2 for PL logic
FCLK_RESET3_N	☐	Enables general purpose reset signal 3 for PL logic
AXI Non Secure Enablement	0	Enable AXI Non Secure Transaction
GP Master AXI Interface		
M AXI GP0 interface	☑	Enables General purpose AXI master interface 0
M AXI GP1 interface	☐	Enables General purpose AXI master interface 1
GP Slave AXI Interface		
HP Slave AXI Interface		
S AXI HP0 interface	☑	Enables AXI high performance slave interface 0
S AXI HP1 interface	☐	Enables AXI high performance slave interface 1
S AXI HP2 interface	☐	Enables AXI high performance slave interface 2

OK | Cancel

Zynq7 PS Re-customize IP for AXI4

INIPRO

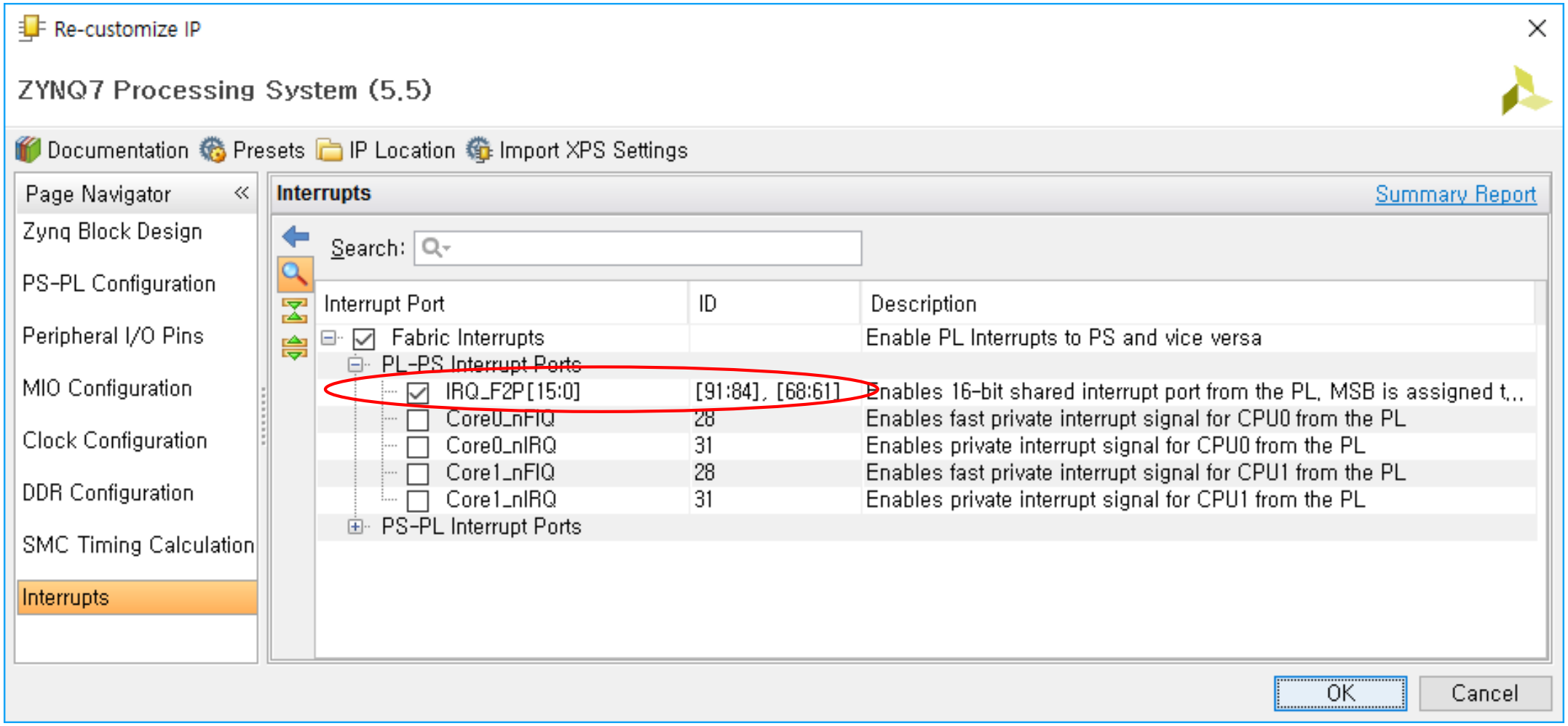
Based on Zynq7 PS Re-customize IP of embedded_linux project



Zynq7 PS Re-customize IP for AXI4

INIPRO

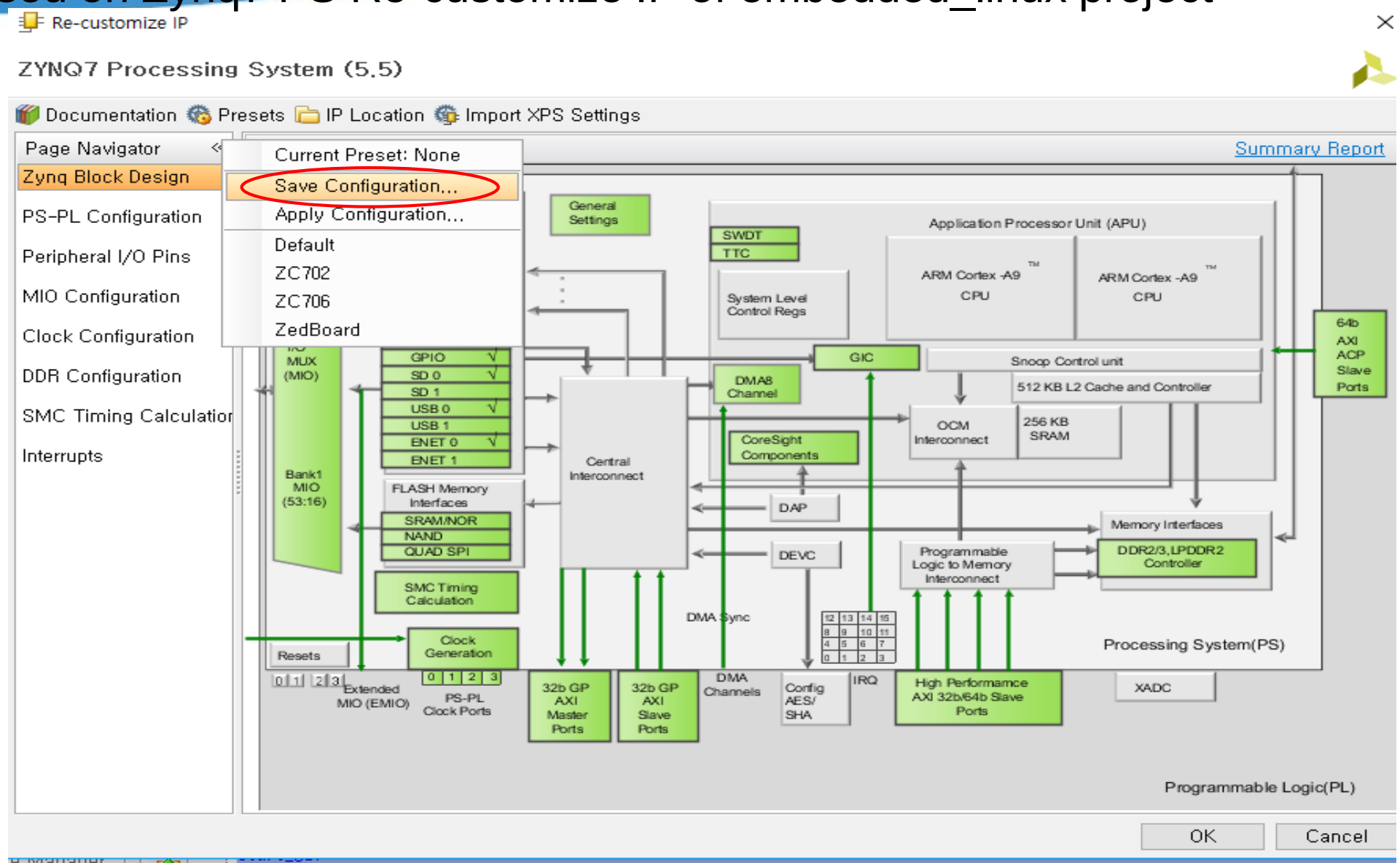
Based on Zynq7 PS Re-customize IP of embedded_linux project

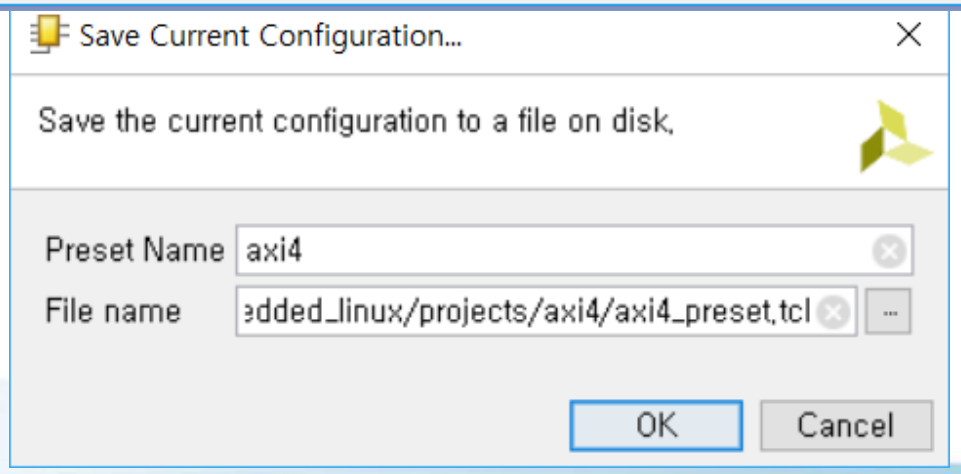


Zynq7 PS Re-customize IP for AXI4

INIPRO

Based on Zynq7 PS Re-customize IP of embedded_linux project





Project for AXI4

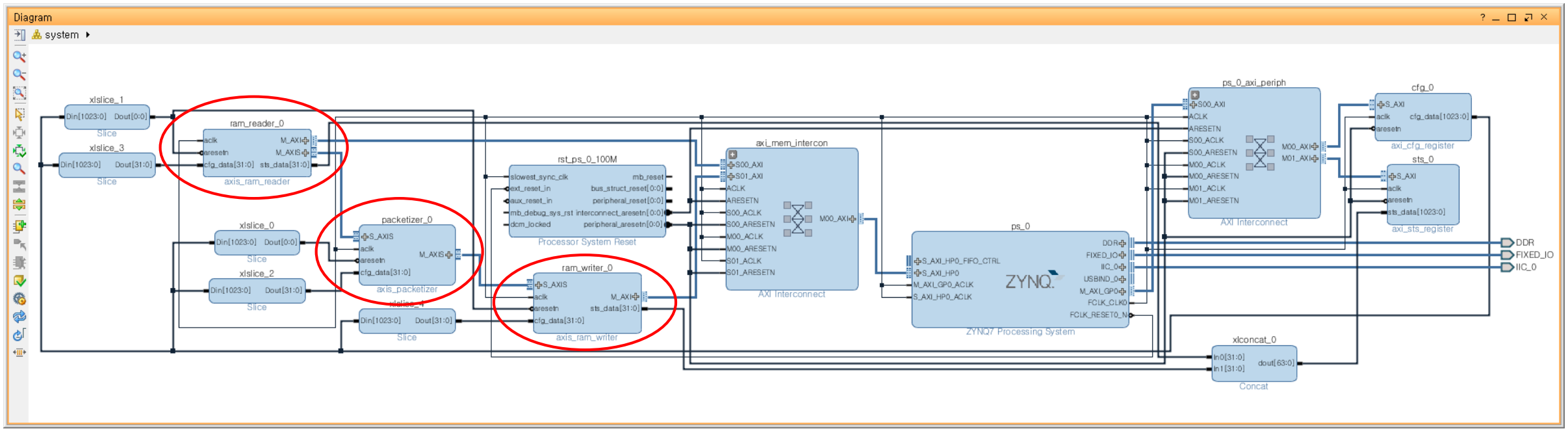
https://github.com/inipro/embedded_linux/projects/axi4

In cmd window for Vivado

```
C:\> cd C:\Users\hokim\work\embedded_linux\projects\axi4
C:\> vivado  -nolog -nojournal -mode batch -source axi4.tcl
```

output : axi4\axi4.xpr…

Project for AXI4



Address Editor

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
ps_0					
Data (32 address bits : 0x40000000 [1G])					
cfg_0	S_AXI	reg0	0x4000_0000	64K	0x4000_FFFF
sts_0	S_AXI	reg0	0x4001_0000	64K	0x4001_FFFF
ram_reader_0					
M_AXI (32 address bits : 4G)					
ps_0	S_AXI_HP0	HP0_DDR_L...	0x0000_0000	512M	0x1FFF_FFFF
ram_writer_0					
M_AXI (32 address bits : 4G)					
ps_0	S_AXI_HP0	HP0_DDR_L...	0x0000_0000	512M	0x1FFF_FFFF

IP for AXI4 Project

C:\Users\hokim\work\embedded_linux\projects\axi4\ip\axis_ram_reader_v1_0
→ Ram read to axis stream

hdl\axis_ram_reader.v

```
`timescale 1 ns / 1 ps

module axis_ram_reader #
(
  parameter integer ADDR_WIDTH = 20,
  //parameter integer AXI_ID_WIDTH = 6,
  parameter integer AXI_ADDR_WIDTH = 32,
  parameter integer AXI_DATA_WIDTH = 64,
  parameter integer AXIS_TDATA_WIDTH = 64
)
(
  // System signals
  input  wire                            aclk,
  input  wire                            aresetn,

  input  wire [AXI_ADDR_WIDTH-1:0]       cfg_data,
  output wire [ADDR_WIDTH-1:0]           sts_data,

  // Master side
  //output wire [AXI_ID_WIDTH-1:0]       m_axi_arid,      // AXI master: Read address ID
  output wire [AXI_ADDR_WIDTH-1:0]       m_axi_araddr,    // AXI master: Read address
  output wire [3:0]                      m_axi_arlen,     // AXI master: Read burst length
  output wire [2:0]                      m_axi_arsize,    // AXI master: Read burst size
  output wire [1:0]                      m_axi_arburst,   // AXI master: Read burst type
  output wire [3:0]                      m_axi_arcache,   // AXI master: Read cache type
  output wire                            m_axi_arvalid,   // AXI master: Read address valid
  input  wire                            m_axi_arready,   // AXI master: Read address ready
```

IP for AXI4 Project

hdl\axis_ram_reader.v

```
  input  wire [AXI_DATA_WIDTH-1:0]          m_axi_rdata,          // AXI master: Read data
  input  wire                               m_axi_rlast,          // AXI master: Read last
  input  wire                               m_axi_rvalid,         // AXI master: Read valid
  output wire                               m_axi_rready,         // AXI master: Read ready

  input  wire                               m_axis_tready,
  output wire [AXIS_TDATA_WIDTH-1:0]        m_axis_tdata,
  output wire                               m_axis_tvalid
);

  function integer clogb2 (input integer value);
    for(clogb2 = 0; value > 0; clogb2 = clogb2 + 1) value = value >> 1;
  endfunction

  localparam integer ADDR_SIZE = clogb2((AXI_DATA_WIDTH/8)-1);

  reg int_arvalid_reg, int_arvalid_next;
  reg int_rready_reg, int_rready_next;
  reg [ADDR_WIDTH-1:0] int_addr_reg, int_addr_next;
  //reg [AXI_ID_WIDTH-1:0] int_arid_reg, int_arid_next;

  wire int_full_wire, int_empty_wire, int_rden_wire;
  wire int_tvalid_wire;
  wire [71:0] int_rdata_wire;

  assign int_rden_wire = m_axis_tready & ~int_empty_wire;
```

IP for AXI4 Project

hdl\axis_ram_reader.v

```
FIFO36E1 #(
  .FIRST_WORD_FALL_THROUGH("TRUE"),
                .ALMOST_FULL_OFFSET(13'h1f1),
                .DATA_WIDTH(72),
                .FIFO_MODE("FIFO36_72")
) fifo_0 (
                .ALMOSTFULL(int_full_wire),
                .EMPTY(int_empty_wire),
                .RST(~aresetn),
                .WRCLK(aclk),
                .WREN(int_rready_reg & m_axi_rvalid),
                .DI({{(72-AXI_DATA_WIDTH){1'b0}}, m_axi_rdata}),
                .RDCLK(aclk),
                .RDEN(int_rden_wire),
                .DO(int_rdata_wire)
 );

always @(posedge aclk)
 begin
                if (~aresetn)
                begin
                  int_arvalid_reg <= 1'b0;
     int_rready_reg <= 1'b0;
                  int_addr_reg <= {(ADDR_WIDTH){1'b0}};
     //int_arid_reg <= {(AXI_ID_WIDTH){1'b0}};
                end
                else
                begin
                  int_arvalid_reg <= int_arvalid_next;
                  int_rready_reg <= int_rready_next;
                  int_addr_reg <= int_addr_next;
                  //int_arid_reg <= int_arid_next;
                end
 end
```

IP for AXI4 Project

hdl\axis_ram_reader.v

```
always @*
 begin
  int_arvalid_next = int_arvalid_reg;
                int_rready_next = int_rready_reg;
                int_addr_next = int_addr_reg;
                //int_arid_next = int_arid_reg;

                if (~int_full_wire & ~int_arvalid_reg & ~int_rready_reg)
                begin
                  int_arvalid_next = 1'b1;
                  int_rready_next = 1'b1;
                end

                if (m_axi_arready & int_arvalid_reg)
                begin
                  int_arvalid_next = 1'b0;
                end

                if (int_rready_reg & m_axi_rlast)
                begin
                  int_addr_next = int_addr_reg + 5'd16;
                  //int_arid_next = int_arid_reg + 1'b1;
                  if (int_full_wire)
                                int_rready_next = 1'b0;
                  else
                  begin
                    int_arvalid_next = 1'b1;
                  end
                end
end
```

IP for AXI4 Project

INIPRO

hdl\axis_ram_reader.v

```
 assign sts_data = int_addr_reg;

  //assign m_axi_arid = int_arid_reg;
  assign m_axi_araddr = cfg_data + {int_addr_reg,
{(ADDR_SIZE){1'b0}}};
  assign m_axi_arlen = 4'd15;
  assign m_axi_arsize = ADDR_SIZE;
  assign m_axi_arburst = 2'b01;
  assign m_axi_arcache = 4'b0011;
  assign m_axi_arvalid = int_arvalid_reg;
  assign m_axi_rready = int_rready_reg;
  assign m_axis_tvalid = int_rden_wire;
  assign m_axis_tdata = int_rdata_wire[AXI_DATA_WIDTH-1:0];

endmodule
```

IP for AXI4 Project

INIPRO

axis_ram_reader.tcl

```
set ip_name "axis_ram_reader"
file delete -force component.xml xgui

# Collect the names of the HDL source files that are used by this IP here.
set file_list [list "hdl/axis_ram_reader.v"]

# Create a new Vivado project for this IP and add the source files.
create_project $ip_name . -force
set proj_dir [get_property directory [current_project]]
set proj_name [get_projects $ip_name]
set proj_fileset [get_filesets sources_1]
add_files -norecurse -scan_for_includes -fileset $proj_fileset $file_list
set_property "top" "$ip_name" $proj_fileset
ipx::package_project -root_dir .

# Set the IP core information properties.
set core [ipx::current_core]
set_property vendor {hokim} $core
set_property library {ip} $core
set_property version {1.0} $core
set_property vendor_display_name {Hyunok Kim} $core
set_property company_url {https://github.com/hokim72} $core
set_property supported_families {{zynq} {Production}} $core
set_property name $ip_name $core
set_property display_name $ip_name $core
set_property description $ip_name $core

set parameter [ipx::get_user_parameters AXI_DATA_WIDTH -of_objects $core]
set_property DISPLAY_NAME {AXI DATA WIDTH} $parameter
set_property DESCRIPTION {Width of the AXI data bus.} $parameter

set parameter [ipgui::get_guiparamspec -name AXI_DATA_WIDTH -component $core]
set_property DISPLAY_NAME {AXI DATA WIDTH} $parameter
set_property TOOLTIP {Width of the AXI data bus.} $parameter
```

IP for AXI4 Project

axis_ram_reader.tcl

```
set parameter [ipx::get_user_parameters AXI_ADDR_WIDTH -of_objects $core]
set_property DISPLAY_NAME {AXI ADDR WIDTH} $parameter
set_property DESCRIPTION {Width of the AXI address bus.} $parameter

set parameter [ipgui::get_guiparamspec -name AXI_ADDR_WIDTH -component $core]
set_property DISPLAY_NAME {AXI ADDR WIDTH} $parameter
set_property TOOLTIP {Width of the AXI address bus.} $parameter

#set parameter [ipx::get_user_parameters AXI_ID_WIDTH -of_objects $core]
#set_property DISPLAY_NAME {AXI ID WIDTH} $parameter
#set_property DESCRIPTION {Width of the AXI ID bus.} $parameter

#set parameter [ipgui::get_guiparamspec -name AXI_ID_WIDTH -component $core]
#set_property DISPLAY_NAME {AXI ID WIDTH} $parameter
#set_property TOOLTIP {Width of the AXI ID bus.} $parameter

set parameter [ipx::get_user_parameters AXIS_TDATA_WIDTH -of_objects $core]
set_property DISPLAY_NAME {AXIS TDATA WIDTH} $parameter
set_property DESCRIPTION {Width of the S_AXIS data bus.} $parameter

set parameter [ipgui::get_guiparamspec -name AXIS_TDATA_WIDTH -component $core]
set_property DISPLAY_NAME {AXIS TDATA WIDTH} $parameter
set_property TOOLTIP {Width of the S_AXIS data bus.} $parameter

set parameter [ipx::get_user_parameters ADDR_WIDTH -of_objects $core]
set_property DISPLAY_NAME {ADDR WIDTH} $parameter
set_property DESCRIPTION {Width of the address.} $parameter

set parameter [ipgui::get_guiparamspec -name ADDR_WIDTH -component $core]
set_property DISPLAY_NAME {ADDR WIDTH} $parameter
set_property TOOLTIP {Width of the address.} $parameter

set address [ipx::get_address_spaces m_axi -of_objects $core]
set_property NAME M_AXI $address
```

IP for AXI4 Project

INIPRO

axis_ram_reader.tcl

```
set bus [ipx::get_bus_interfaces -of_objects $core m_axi]
set_property NAME M_AXI $bus
set_property INTERFACE_MODE master $bus
set_property MASTER_ADDRESS_SPACE_REF M_AXI $bus

set bus [ipx::get_bus_interfaces -of_objects $core m_axis]
set_property NAME M_AXIS $bus
set_property INTERFACE_MODE master $bus

set bus [ipx::get_bus_interfaces aclk]
set parameter [ipx::get_bus_parameters -of_objects $bus ASSOCIATED_BUSIF]
set_property VALUE M_AXI:M_AXIS $parameter

# Generate the XGUI files to accompany this IP core.
ipx::create_xgui_files $core

# Save the IP core.
ipx::save_core $core

# Close the current project.
close_project

file delete -force $ip_name.cache $ip_name.hw $ip_name.ip_user_files $ip_name.xpr
```

```
C:\> cd C:\Users\hokim\work\embedded_linux\projects\axi4\ip\axis_ram_reader_v1_0
C:\> vivado -nolog -nojournal -mode batch -source axis_ram_reader.tcl
```

IP for AXI4 Project

INIPRO

C:\Users\hokim\work\embedded_linux\projects\axi4\ip\axis_ram_writer_v1_0
→ axis stream to ram write

hdl\axis_ram_writer.v

```
`timescale 1 ns / 1 ps

module axis_ram_writer #
(
  parameter integer ADDR_WIDTH = 20,
  //parameter integer AXI_ID_WIDTH = 6,
  parameter integer AXI_ADDR_WIDTH = 32,
  parameter integer AXI_DATA_WIDTH = 64,
  parameter integer AXIS_TDATA_WIDTH = 64
)
(
  // System signals
  input  wire                            aclk,
  input  wire                            aresetn,

  input  wire [AXI_ADDR_WIDTH-1:0]       cfg_data,
  output wire [ADDR_WIDTH-1:0]           sts_data,

  // Master side
  //output wire [AXI_ID_WIDTH-1:0]       m_axi_awid,    // AXI master: Write address ID
  output wire [AXI_ADDR_WIDTH-1:0]       m_axi_awaddr,  // AXI master: Write address
  output wire [3:0]                      m_axi_awlen,   // AXI master: Write burst length
  output wire [2:0]                      m_axi_awsize,  // AXI master: Write burst size
  output wire [1:0]                      m_axi_awburst, // AXI master: Write burst type
  output wire [3:0]                      m_axi_awcache, // AXI master: Write memory type
  output wire                            m_axi_awvalid, // AXI master: Write address valid
  input  wire                            m_axi_awready, // AXI master: Write address ready
  //output wire [AXI_ID_WIDTH-1:0]       m_axi_wid,     // AXI master: Write data ID
```

IP for AXI4 Project

hdl\axis_ram_writer.v

```
 output wire [AXI_DATA_WIDTH-1:0]          m_axi_wdata,   // AXI master: Write data
  output wire [AXI_DATA_WIDTH/8-1:0]       m_axi_wstrb,   // AXI master: Write strobes
  output wire                              m_axi_wlast,   // AXI master: Write last
  output wire                              m_axi_wvalid,  // AXI master: Write valid
  input  wire                              m_axi_wready,  // AXI master: Write ready
  input  wire                              m_axi_bvalid,  // AXI master: Write response valid
  output wire                              m_axi_bready,  // AXI master: Write response ready

  // Slave side
  output wire                              s_axis_tready,
  input  wire [AXIS_TDATA_WIDTH-1:0]       s_axis_tdata,
  input  wire                              s_axis_tvalid
);

  function integer clogb2 (input integer value);
    for(clogb2 = 0; value > 0; clogb2 = clogb2 + 1) value = value >> 1;
  endfunction

  localparam integer ADDR_SIZE = clogb2((AXI_DATA_WIDTH/8)-1);

 reg int_awvalid_reg, int_awvalid_next;
  reg int_wvalid_reg, int_wvalid_next;
  reg [ADDR_WIDTH-1:0] int_addr_reg, int_addr_next;
  //reg [AXI_ID_WIDTH-1:0] int_wid_reg, int_wid_next;

  wire int_full_wire, int_empty_wire, int_rden_wire;
  wire int_wlast_wire, int_tready_wire;
  wire [71:0] int_wdata_wire;

  assign int_tready_wire = ~int_full_wire;
  assign int_wlast_wire = &int_addr_reg[3:0];
  assign int_rden_wire = m_axi_wready & int_wvalid_reg;
```

IP for AXI4 Project

INIPRO

hdl\axis_ram_writer.v

```
FIFO36E1 #(
   .FIRST_WORD_FALL_THROUGH("TRUE"),
   .ALMOST_EMPTY_OFFSET(13'hf),
   .DATA_WIDTH(72),
   .FIFO_MODE("FIFO36_72")
 ) fifo_0 (
   .FULL(int_full_wire),
   .ALMOSTEMPTY(int_empty_wire),
   .RST(~aresetn),
   .WRCLK(aclk),
   .WREN(int_tready_wire & s_axis_tvalid),
   .DI({{(72-AXIS_TDATA_WIDTH){1'b0}}, s_axis_tdata}),
   .RDCLK(aclk),
   .RDEN(int_rden_wire),
   .DO(int_wdata_wire)
);

always @(posedge aclk)
begin
  if(~aresetn)
  begin
    int_awvalid_reg <= 1'b0;
    int_wvalid_reg <= 1'b0;
    int_addr_reg <= {(ADDR_WIDTH){1'b0}};
    //int_wid_reg <= {(AXI_ID_WIDTH){1'b0}};
  end
  else
  begin
    int_awvalid_reg <= int_awvalid_next;
    int_wvalid_reg <= int_wvalid_next;
    int_addr_reg <= int_addr_next;
    //int_wid_reg <= int_wid_next;
  end
end
```

IP for AXI4 Project

INIPRO

hdl\axis_ram_writer.v

```
always @*
 begin
  int_awvalid_next = int_awvalid_reg;
  int_wvalid_next = int_wvalid_reg;
  int_addr_next = int_addr_reg;
  //int_wid_next = int_wid_reg;

  if(~int_empty_wire & ~int_awvalid_reg & ~int_wvalid_reg)
  begin
    int_awvalid_next = 1'b1;
    int_wvalid_next = 1'b1;
  end

  if(m_axi_awready & int_awvalid_reg)
  begin
    int_awvalid_next = 1'b0;
  end

  if(int_rden_wire)
  begin
    int_addr_next = int_addr_reg + 1'b1;
  end

  if(m_axi_wready & int_wlast_wire)
  begin
    //int_wid_next = int_wid_reg + 1'b1;
    if(int_empty_wire)
    begin
      int_wvalid_next = 1'b0;
    end
    else
    begin
      int_awvalid_next = 1'b1;
    end
  end
end
```

IP for AXI4 Project

hdl\axis_ram_writer.v

```
  assign sts_data = int_addr_reg;

  //assign m_axi_awid = int_wid_reg;
  assign m_axi_awaddr = cfg_data + {int_addr_reg,
{(ADDR_SIZE){1'b0}}};
  assign m_axi_awlen = 4'd15;
  assign m_axi_awsize = ADDR_SIZE;
  assign m_axi_awburst = 2'b01;
  assign m_axi_awcache = 4'b0011;
  assign m_axi_awvalid = int_awvalid_reg;
  //assign m_axi_wid = int_wid_reg;
  assign m_axi_wdata = int_wdata_wire[AXI_DATA_WIDTH-1:0];
  assign m_axi_wstrb = {(AXI_DATA_WIDTH/8){1'b1}};
  assign m_axi_wlast = int_wlast_wire;
  assign m_axi_wvalid = int_wvalid_reg;
  assign m_axi_bready = 1'b1;

  assign s_axis_tready = int_tready_wire;

endmodule
```

IP for AXI4 Project

axis_ram_writer.tcl

```
set ip_name "axis_ram_writer"
file delete -force component.xml xgui

# Collect the names of the HDL source files that are used by this IP here.
set file_list [list "hdl/axis_ram_writer.v"]

# Create a new Vivado project for this IP and add the source files.
create_project $ip_name . -force
set proj_dir [get_property directory [current_project]]
set proj_name [get_projects $ip_name]
set proj_fileset [get_filesets sources_1]
add_files -norecurse -scan_for_includes -fileset $proj_fileset $file_list
set_property "top" "$ip_name" $proj_fileset
ipx::package_project -root_dir .

# Set the IP core information properties.
set core [ipx::current_core]
set_property vendor {hokim} $core
set_property library {ip} $core
set_property version {1.0} $core
set_property vendor_display_name {Hyunok Kim} $core
set_property company_url {https://github.com/hokim72} $core
set_property supported_families {{zynq} {Production}} $core
set_property name $ip_name $core
set_property display_name $ip_name $core
set_property description $ip_name $core

set parameter [ipx::get_user_parameters AXI_DATA_WIDTH -of_objects $core]
set_property DISPLAY_NAME {AXI DATA WIDTH} $parameter
set_property DESCRIPTION {Width of the AXI data bus.} $parameter

set parameter [ipgui::get_guiparamspec -name AXI_DATA_WIDTH -component $core]
set_property DISPLAY_NAME {AXI DATA WIDTH} $parameter
set_property TOOLTIP {Width of the AXI data bus.} $parameter
```

IP for AXI4 Project

axis_ram_writer.tcl

```
set parameter [ipx::get_user_parameters AXI_ADDR_WIDTH -of_objects $core]
set_property DISPLAY_NAME {AXI ADDR WIDTH} $parameter
set_property DESCRIPTION {Width of the AXI address bus.} $parameter

set parameter [ipgui::get_guiparamspec -name AXI_ADDR_WIDTH -component $core]
set_property DISPLAY_NAME {AXI ADDR WIDTH} $parameter
set_property TOOLTIP {Width of the AXI address bus.} $parameter

#set parameter [ipx::get_user_parameters AXI_ID_WIDTH -of_objects $core]
#set_property DISPLAY_NAME {AXI ID WIDTH} $parameter
#set_property DESCRIPTION {Width of the AXI ID bus.} $parameter

#set parameter [ipgui::get_guiparamspec -name AXI_ID_WIDTH -component $core]
#set_property DISPLAY_NAME {AXI ID WIDTH} $parameter
#set_property TOOLTIP {Width of the AXI ID bus.} $parameter

set parameter [ipx::get_user_parameters AXIS_TDATA_WIDTH -of_objects $core]
set_property DISPLAY_NAME {AXIS TDATA WIDTH} $parameter
set_property DESCRIPTION {Width of the S_AXIS data bus.} $parameter

set parameter [ipgui::get_guiparamspec -name AXIS_TDATA_WIDTH -component $core]
set_property DISPLAY_NAME {AXIS TDATA WIDTH} $parameter
set_property TOOLTIP {Width of the S_AXIS data bus.} $parameter

set parameter [ipx::get_user_parameters ADDR_WIDTH -of_objects $core]
set_property DISPLAY_NAME {ADDR WIDTH} $parameter
set_property DESCRIPTION {Width of the address.} $parameter

set parameter [ipgui::get_guiparamspec -name ADDR_WIDTH -component $core]
set_property DISPLAY_NAME {ADDR WIDTH} $parameter
set_property TOOLTIP {Width of the address.} $parameter

set address [ipx::get_address_spaces m_axi -of_objects $core]
set_property NAME M_AXI $address
```

IP for AXI4 Project

INIPRO

axis_ram_writer.tcl

```
set bus [ipx::get_bus_interfaces -of_objects $core m_axi]
set_property NAME M_AXI $bus
set_property INTERFACE_MODE master $bus
set_property MASTER_ADDRESS_SPACE_REF M_AXI $bus

set bus [ipx::get_bus_interfaces -of_objects $core s_axis]
set_property NAME S_AXIS $bus
set_property INTERFACE_MODE slave $bus

set bus [ipx::get_bus_interfaces aclk]
set parameter [ipx::get_bus_parameters -of_objects $bus ASSOCIATED_BUSIF]
set_property VALUE M_AXI:S_AXIS $parameter

# Generate the XGUI files to accompany this IP core.
ipx::create_xgui_files $core

# Save the IP core.
ipx::save_core $core

# Close the current project.
close_project

file delete -force $ip_name.cache $ip_name.hw $ip_name.ip_user_files $ip_name.xpr
```

```
C:\> cd C:\Users\hokim\work\embedded_linux\projects\axi4\ip\axis_ram_writer_v1_0
C:\> vivado -nolog -nojournal -mode batch -source axis_ram_writer.tcl
```

IP for AXI4 Project

INIPRO

C:\Users\hokim\work\embedded_linux\projects\axi4\ip\axis_packetizer_v1_0
→ axis stream to axis stream packet

hdl\axis_packetizer.v

```
`timescale 1 ns / 1 ps

module axis_packetizer #
(
  parameter integer AXIS_TDATA_WIDTH = 32,
  parameter integer CNTR_WIDTH = 32,
  parameter         CONTINUOUS = "FALSE"
)
(
  // System signals
  input  wire                                aclk,
  input  wire                                aresetn,

  input  wire [CNTR_WIDTH-1:0]               cfg_data,

  // Slave side
  output wire                                s_axis_tready,
  input  wire [AXIS_TDATA_WIDTH-1:0]         s_axis_tdata,
  input  wire                                s_axis_tvalid,

  // Master side
  input  wire                                m_axis_tready,
  output wire [AXIS_TDATA_WIDTH-1:0]         m_axis_tdata,
  output wire                                m_axis_tvalid,
  output wire                                m_axis_tlast
);
```

IP for AXI4 Project

hdl\axis_packetizer.v

```
reg [CNTR_WIDTH-1:0] int_cntr_reg, int_cntr_next;
reg int_enbl_reg, int_enbl_next;

wire int_comp_wire, int_tvalid_wire, int_tlast_wire;

always @(posedge aclk)
begin
  if(~aresetn)
  begin
    int_cntr_reg <= {(CNTR_WIDTH){1'b0}};
    int_enbl_reg <= 1'b0;
  end
  else
  begin
    int_cntr_reg <= int_cntr_next;
    int_enbl_reg <= int_enbl_next;
  end
end

assign int_comp_wire = int_cntr_reg < cfg_data-1;
assign int_tvalid_wire = int_enbl_reg & s_axis_tvalid;
assign int_tlast_wire = ~int_comp_wire;

generate
  if(CONTINUOUS == "TRUE")
  begin : CONT
    always @*
    begin
      int_cntr_next = int_cntr_reg;
      int_enbl_next = int_enbl_reg;
```

IP for AXI4 Project

INIPRO

hdl\axis_packetizer.v

```
if(~int_enbl_reg & int_comp_wire)
      begin
        int_enbl_next = 1'b1;
      end

      if(m_axis_tready & int_tvalid_wire & int_comp_wire)
      begin
        int_cntr_next = int_cntr_reg + 1'b1;
      end

      if(m_axis_tready & int_tvalid_wire & int_tlast_wire)
      begin
        int_cntr_next = {(CNTR_WIDTH){1'b0}};
      end
    end
  end
end
else
begin : STOP
  always @*
  begin
    int_cntr_next = int_cntr_reg;
    int_enbl_next = int_enbl_reg;

    if(~int_enbl_reg & int_comp_wire)
    begin
      int_enbl_next = 1'b1;
    end

    if(m_axis_tready & int_tvalid_wire & int_comp_wire)
    begin
      int_cntr_next = int_cntr_reg + 1'b1;
    end
```

IP for AXI4 Project

INIPRO

hdl\axis_packetizer.v

```
 if(m_axis_tready & int_tvalid_wire & int_tlast_wire)
        begin
          int_enbl_next = 1'b0;
        end
      end
    end
  endgenerate

  assign s_axis_tready = int_enbl_reg & m_axis_tready;
  assign m_axis_tdata = s_axis_tdata;
  assign m_axis_tvalid = int_tvalid_wire;
  assign m_axis_tlast = int_enbl_reg & int_tlast_wire;

endmodule
```

IP for AXI4 Project

axis_packetizer.tcl

```
set ip_name "axis_packetizer"
file delete -force component.xml xgui

# Collect the names of the HDL source files that are used by this IP here.
set file_list [list "hdl/axis_packetizer.v"]

# Create a new Vivado project for this IP and add the source files.
create_project $ip_name . -force
set proj_dir [get_property directory [current_project]]
set proj_name [get_projects $ip_name]
set proj_fileset [get_filesets sources_1]
add_files -norecurse -scan_for_includes -fileset $proj_fileset $file_list
set_property "top" "$ip_name" $proj_fileset
ipx::package_project -root_dir .

# Set the IP core information properties.
set core [ipx::current_core]
set_property vendor {hokim} $core
set_property library {ip} $core
set_property version {1.0} $core
set_property vendor_display_name {Hyunok Kim} $core
set_property company_url {https://github.com/hokim72} $core
set_property supported_families {{zynq} {Production}} $core
set_property name $ip_name $core
set_property display_name $ip_name $core
set_property description $ip_name $core


set parameter [ipx::get_user_parameters AXIS_TDATA_WIDTH -of_objects $core]
set_property DISPLAY_NAME {AXIS TDATA WIDTH} $parameter
set_property DESCRIPTION {Width of the M_AXIS and S_AXIS data buses.} $parameter

set parameter [ipgui::get_guiparamspec -name AXIS_TDATA_WIDTH -component $core]
set_property DISPLAY_NAME {AXI TDATA WIDTH} $parameter
set_property TOOLTIP {Width of the M_AXIS and S_AXIS data buses.} $parameter
```

IP for AXI4 Project

axis_packetizer.tcl

```
set parameter [ipx::get_user_parameters CNTR_WIDTH -of_objects $core]
set_property DISPLAY_NAME {CNTR WIDTH} $parameter
set_property DESCRIPTION {Width of the counter register.} $parameter

set parameter [ipgui::get_guiparamspec -name CNTR_WIDTH -component $core]
set_property DISPLAY_NAME {CNTR WIDTH} $parameter
set_property TOOLTIP {Width of the counter register.} $parameter

set parameter [ipx::get_user_parameters CONTINUOUS -of_objects $core]
set_property DISPLAY_NAME {CONTINUOUS} $parameter
set_property DESCRIPTION {If TRUE, packetizer runs continuously.} $parameter

set parameter [ipgui::get_guiparamspec -name CONTINUOUS -component $core]
set_property DISPLAY_NAME {CONTINUOUS} $parameter
set_property TOOLTIP {If TRUE, packetizer runs continuously.} $parameter

set bus [ipx::get_bus_interfaces -of_objects $core m_axis]
set_property NAME M_AXIS $bus
set_property INTERFACE_MODE master $bus

set bus [ipx::get_bus_interfaces -of_objects $core s_axis]
set_property NAME S_AXIS $bus
set_property INTERFACE_MODE slave $bus

set bus [ipx::get_bus_interfaces aclk]
set parameter [ipx::get_bus_parameters -of_objects $bus ASSOCIATED_BUSIF]
set_property VALUE M_AXIS:S_AXIS $parameter
```

IP for AXI4 Project

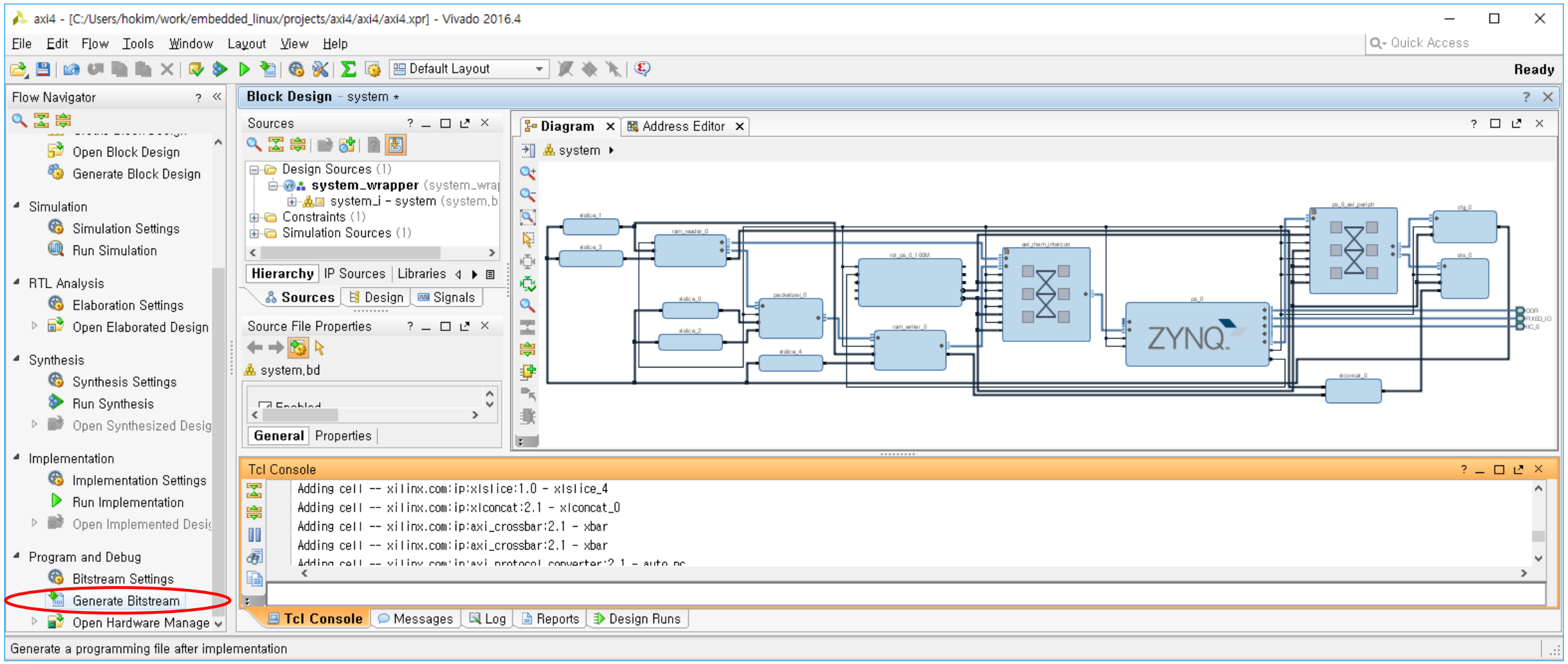
INIPRO

axis_packetizer.tcl

```
# Generate the XGUI files to accompany this IP core.
ipx::create_xgui_files $core

# Save the IP core.
ipx::save_core $core

# Close the current project.
close_project

file delete -force $ip_name.cache $ip_name.hw $ip_name.ip_user_files $ip_name.xpr
```

```
C:\> cd C:\Users\hokim\work\embedded_linux\projects\axi4\ip\axis_packetizer_v1_0
C:\> vivado -nolog -nojournal -mode batch -source axis_packetizer.tcl
```

Bit Generation for AXI4



Boot.bin Build/DT Generation for AXI4

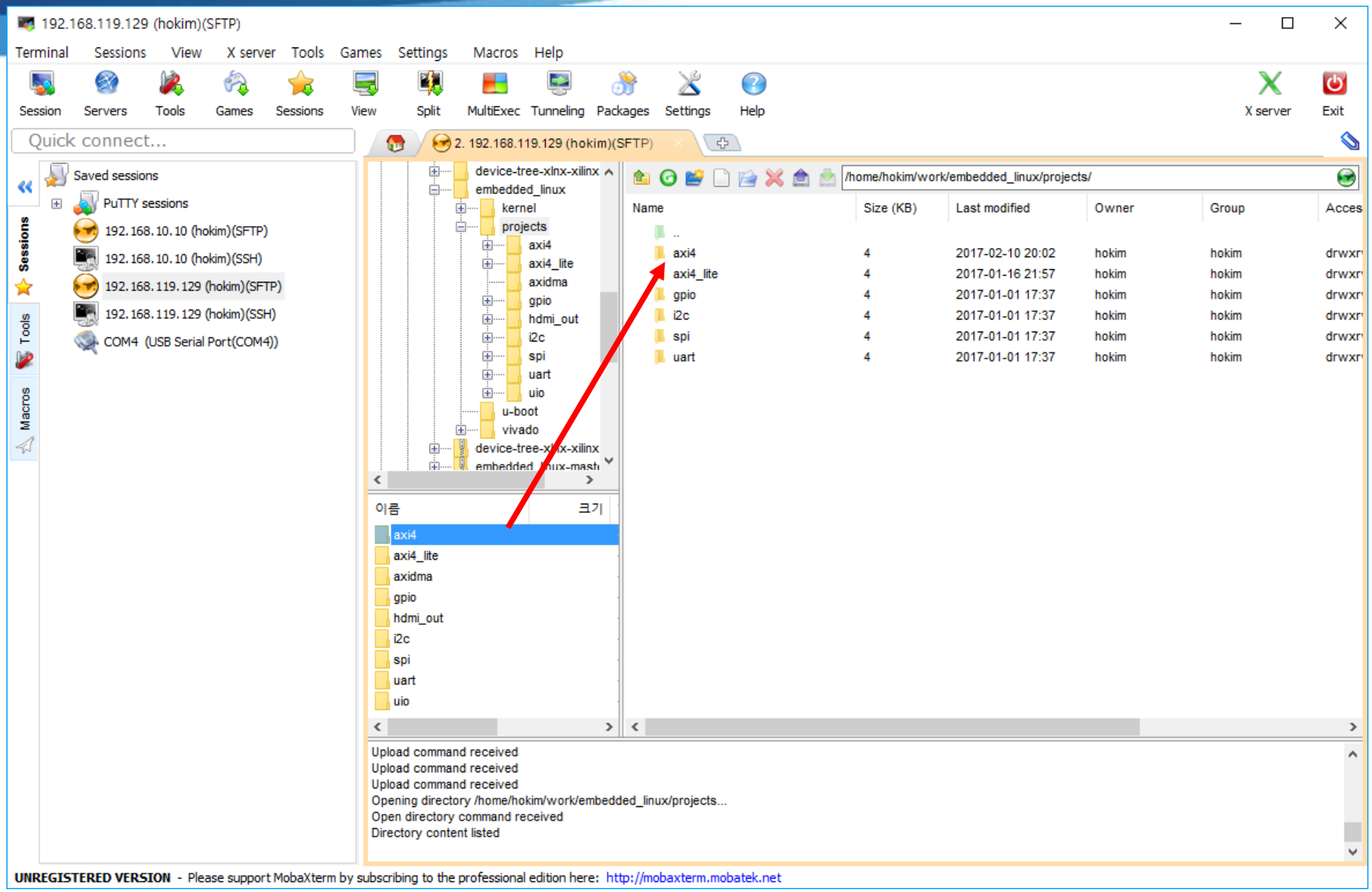
INIPRO

C:\Users\hokim\work\embedded_linux\projects\axi4\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\axi4
C:\> all
```

Device Tree Compile for AXI4




```
$ cd ~/work/embedded_linux/projects/axi4
$ cp axi4/axi4.tree/system.dts  system_axi4.dts
$ nano system_axi4.dts
```

Device Tree Compile for AXI4

system_axi4.dts

```
/dts-v1/;
/include/ "zynq-7000.dtsi"
/include/ "pl.dtsi“
…….
    chosen {
-       bootargs = "console=ttyPS0,115200
root=/dev/mmcblk0p2 ro rootfstype=ext4 earlyprintk rootwait";
+       bootargs = "console=ttyPS0,115200
root=/dev/mmcblk0p2 ro rootfstype=ext4 earlyprintk rootwait
uio_pdrv_genirq.of_id=generic-uio mem=480M";
};

…….
&clkc {
    fclk-enable = <0x1>;
    ps-clk-frequency = <50000000>;
};
+&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>;
+};

+&cfg_0 {
+               compatible = "generic-uio";
+};
+&sts_0 {
+               compatible = "generic-uio";
+               interrupt-parent =<&intc>;
+               interrupts = <0x0 0x1d 0x1>;
+};
+ / {
+    ram_reader {
+        compatible = "generic-uio";
+        reg = <0x1E000000 0x1000000>;
+    };
+    ram_writer {
+        compatible = "generic-uio";
+        reg = <0x1F000000 0x1000000>;
+    };
+};
```

Device Tree Compile for AXI4

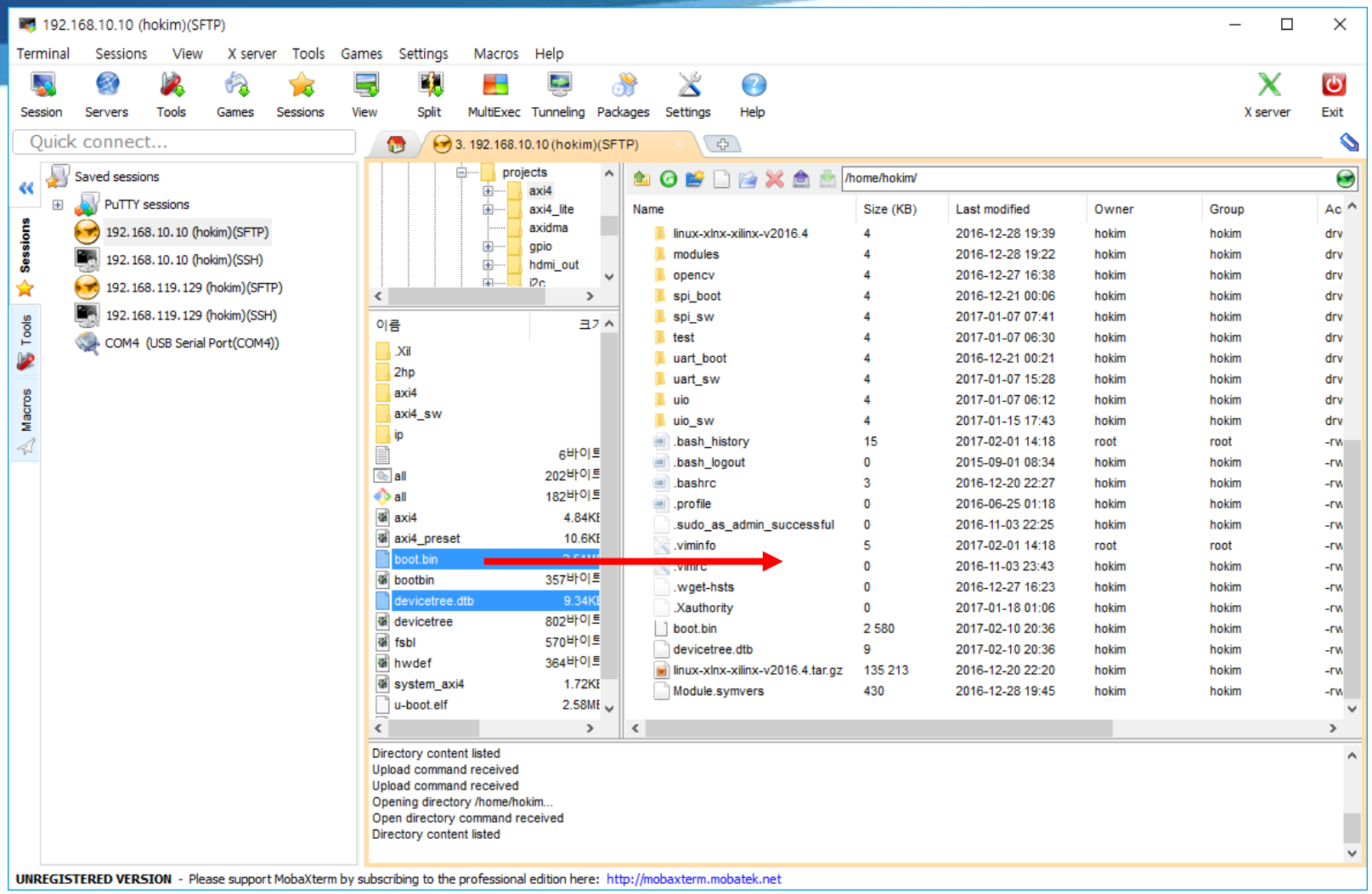
INIPRO

axi4/axi4.tree/*pl.dtsi*

```
/ {
  amba_pl: amba_pl {
    #address-cells = <1>;
    #size-cells = <1>;
    compatible = "simple-bus";
    ranges ;
    cfg_0: axi_cfg_register@40000000 {
      compatible = "xlnx,axi-cfg-register-1.0";
      reg = <0x40000000 0x10000>;
    };
    sts_0: axi_sts_register@40010000 {
      compatible = "xlnx,axi-sts-register-1.0";
      reg = <0x40010000 0x10000>;
    };
  };
};
```

```
$ dtc -O dtb -I dts -i axi4/axi4.tree/ -o devicetree.dtb system_axi4.dts
```

Update boot.bin, devicetree.dtb, uEnv.txt for AXI4



login into zybo

```
$ sudo –s
# cd /boot
# rm boot.bin devicetree.dtb
# mv ~hokim/boot.bin .
# mv ~hokim/devicetree.dtb
# vi  /boot/uEnv.txt
```

Update boot.bin, devicetree.dtb, uEnv.txt for AXI4

INIPRO

```
kernel_image=uImage

devicetree_image=devicetree.dtb

kernel_load_address=0x2080000

devicetree_load_address=0x2000000

-#fdt_high=0x1e000000
+fdt_high=0x1e000000


bootcmd=mmcinfo && fatload mmc 0 ${kernel_load_address} ${kernel_image} && fatload mmc 0
${devicetree_load_address} ${devicetree_image} && bootm ${kernel_load_address} - ${devicetree_load_address}
```

```
# sync
# reboot
```

AXI4 sw & test

```
hokim@zybo:~$ ls /sys/class/uio
uio0  uio1 uio2 uio3
hokim@zybo:~$ cat /sys/class/uio/uio0/name
axi_cfg_register
hokim@zybo:~$ cat /sys/class/uio/uio1/name
axi_sts_register
hokim@zybo:~$ cat /sys/class/uio/uio2/name
ram_reader
hokim@zybo:~$ cat /sys/class/uio/uio3/name
ram_writer
```

axi4_test.c

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/mman.h>
#include <fcntl.h>

#define CONTROL_REG                                    0x0
#define PACKETIZER_REG                 0x4
#define RAM_READER_REG                 0x8
#define RAM_WRITER_REG                 0xC

void uio_write(void *uio_base, uint32_t offset, uint32_t value)
{
    *((volatile uint32_t *)(uio_base + offset)) = value;
}

uint32_t uio_read(void *uio_base, uint32_t offset)
{
    return *((volatile uint32_t *)(uio_base + offset));
}

uint32_t get_memory_size(char *sysfs_path_file)
{
    FILE *size_fp;
    uint32_t size;

    // open the file that describes the memory range size that is based on the
    // reg property of the node in the device tree
    size_fp = fopen(sysfs_path_file, "r");

    if (!size_fp) {
        printf("unable to open the uio size file\n");
        exit(-1);
    }
```

AXI4 sw & test

INIPRO

axi4_test.c

```
    // get the size which is an ASCII string such as 0xXXXXXXXX and then be
stop
    // using the file
    fscanf(size_fp, "0x%08X", &size);
    fclose(size_fp);

    return size;
}

int main()
{
                int cfg0_fd, sts0_fd, ram_reader0_fd, ram_writer0_fd;
                void *cfg0, *sts0, *ram_reader0, *ram_writer0;
                uint32_t cfg0_size, sts0_size, ram_reader0_size,
ram_writer0_size;

                if ((cfg0_fd = open("/dev/uio0", O_RDWR)) < 0) {
                                perror("open cfg0");
                                return -1;
                }

                if ((sts0_fd = open("/dev/uio1", O_RDWR)) < 0) {
                                perror("open sts0");
                                return -1;
                }

                if ((ram_reader0_fd = open("/dev/uio2", O_RDWR)) < 0) {
                                perror("open ram_reader0");
                                return -1;
                }

                if ((ram_writer0_fd = open("/dev/uio3", O_RDWR)) < 0) {
                                perror("open ram_writer0");
                                return -1;
                }
```

AXI4 sw & test

INIPRO

axi4_test.c

```
	cfg0_size =
get_memory_size("/sys/class/uio/uio0/maps/map0/size");
	sts0_size =
get_memory_size("/sys/class/uio/uio1/maps/map0/size");
	ram_reader0_size =
get_memory_size("/sys/class/uio/uio2/maps/map0/size");
	ram_writer0_size =
get_memory_size("/sys/class/uio/uio3/maps/map0/size");

	cfg0 = mmap(NULL, cfg0_size, PROT_READ|PROT_WRITE,
MAP_SHARED, cfg0_fd, 0);

	if (cfg0 == MAP_FAILED) {
		perror("cfg0 mmap call failure");
		return -1;
	}

	sts0 = mmap(NULL, sts0_size, PROT_READ|PROT_WRITE,
MAP_SHARED, sts0_fd, 0);

	if (sts0 == MAP_FAILED) {
		perror("sts0 mmap call failure");
		return -1;
	}

	ram_reader0 = mmap(NULL, ram_reader0_size,
PROT_READ|PROT_WRITE, MAP_SHARED, ram_reader0_fd, 0);

	if (ram_reader0 == MAP_FAILED) {
		perror("ram_reader0 mmap call failure");
		return -1;
	}

	ram_writer0 = mmap(NULL, ram_writer0_size,
PROT_READ|PROT_WRITE, MAP_SHARED, ram_writer0_fd, 0);
```

AXI4 sw & test

INIPRO

axi4_test.c

```
if (ram_writer0 == MAP_FAILED) {
		perror("ram_writer0 mmap call failure");
		return -1;
}

uio_write(cfg0, RAM_READER_REG, 0x1E000000);
uio_write(cfg0, RAM_WRITER_REG, 0x1F000000);
uio_write(cfg0, PACKETIZER_REG, 512);

for(int i=0; i<1026; i++) {
		uio_write(ram_reader0, i*4, i);
		uio_write(ram_writer0, i*4, 0);
}

printf("before...\n");
for(int i=0; i<1026; i++) {
		printf("i = %d, src = %d, dst = %d\n", i,
uio_read(ram_reader0, i*4),
				uio_read(ram_writer0, i*4));
	}

	uio_write(cfg0, CONTROL_REG, 0);
	uio_write(cfg0, CONTROL_REG, uio_read(cfg0, CONTROL_REG)
| 0x1);
	uio_write(cfg0, CONTROL_REG, uio_read(cfg0, CONTROL_REG)
| 0x2);

	printf("after...\n");
	for(int i=0; i<1026; i++) {
		printf("i = %d, src = %d, dst = %d\n", i,
uio_read(ram_reader0, i*4),
				uio_read(ram_writer0, i*4));
	}
```

AXI4 sw & test

axi4_test.c

```
        munmap(cfg0, cfg0_size);
        munmap(sts0, sts0_size);
        munmap(ram_reader0, ram_reader0_size);
        munmap(ram_writer0, ram_writer0_size);

        return 0;
}
```

```
$ cd ~/axi4_sw
$ ls
$ CMakeLists.txt axi4_test.c
$ mkdir build
$ cd build
$ cmake
$ make
$ sudo ./axi4_test
```