Developing Embedded Linux for AD-FMCOMMS4-EZB and ZedBoard using Petalinux

By Abdulaziz A. Alasows

King Abdulaziz City for Science and Technology

Hardware:

- AD-FMCOMMS4-EBZ.
- ZedBoard Rev D.
- Host PC: Ubuntu Linux 16.04.

Installed tools:

- Xilinx Vivado 2017.4.
- Xilinx SDK 2014.4.
- Petalinux 2017.4.

1. Create a Petalinux project by specifying the targeted board BSP.

(BSP for ZedBoard, ZC702, and ZC706 can be downloaded from Xilinx. Match versions with Petalinux tool!).

```
$ petalinux-create -t project -n <NAME> -s <BSP-SRC-FILE>
```

Ex:

```
$ petalinux-create -t project -n fmcomms4_zed -s bsp_file/avnet-digilent-zedboard-v2017.4-final.bsp
```

2. Import hardware configuration

Get the needed HDLs from analog devices for the specified boards **(AD-FMCOMMS4-EBZ, ZedBoard)** and generate **HDF** file from Vivado tool.

To initialize a Petalinux project with a new hardware configuration:

```
plnx-proj-dir$ petalinux-config --get-hw-description=<PATH-TO-HDF-DIRECTORY>
```

Ex:

```
plnx-proj-dir $ petalinux-config --get-hw-description=/opt/projs/zynq/hdls/zed_orig/zed_orig.sdk
```

See ug1157 (v2017.4) page 10 for different commands.

A configuration windows shall pop up, we will be working with ADI kernel (xcomm_zynq) rather than Xilinx kernel (linux-xlnx) in order to get the needed drivers for AD9361/AD9364 and others.

Open a new terminal (Ctrl+Alt+t) and download ADI kernel:

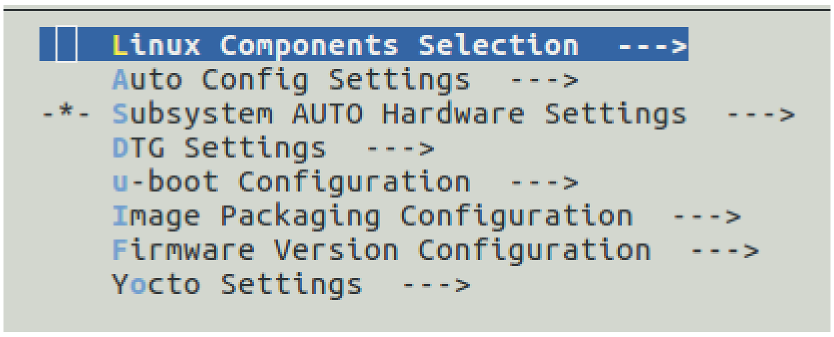
```
$ git clone -b xcomm_zynq --single-branch https://github.com/analogdevicesinc/linux.git
```

NOTE: **xcomm_zynq** kernel version is 4.9 which is compatible with petalinux 2017.4 (it expects 4.9 kernel version). It would fail sanity check if you don't. One can check kernel version by looking at the beginning of the file "Makefile" in ADI kernel root. So make sure which branch you should clone from ADI linux.

Using popped window:

```
Linux Components Selection     >> Linux Kernel
                                         >> Choose "ext-local-src".
                               >> External Linux-kernel local source settings.
                                                   >> Place path to downloaded ADI kernel.
```

Save configurations.



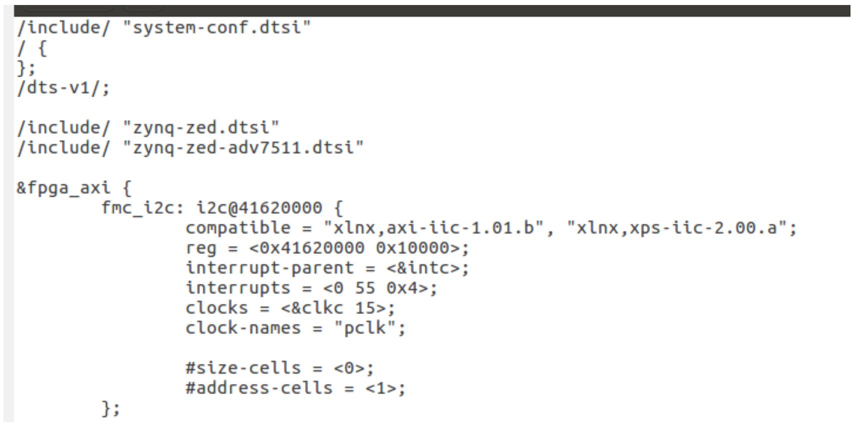
```
    Linux Components Selection  --->
    Auto Config Settings  --->
-*- Subsystem AUTO Hardware Settings  --->
    DTG Settings  --->
    u-boot Configuration  --->
    Image Packaging Configuration  --->
    Firmware Version Configuration  --->
    Yocto Settings  --->
```

3. Configure device tree.

All dts files can be found in “<adi-kernel-root>/arch/arm/boot/dts” (zynq only – arm).

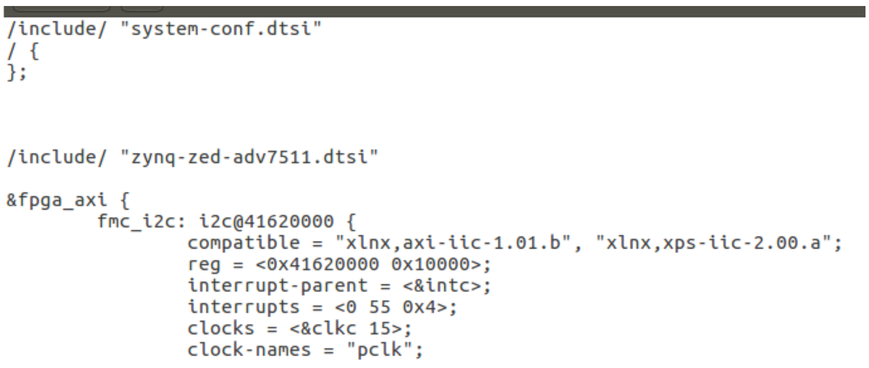
Since we are using **Zedboard** and **AD-FMCOMMS4-EBZ**, we need (zynq-zed-adv7511-ad9364-fmcomms4.dts)

i. Open up the file “system-user.dtsi” found in (<plnx-proj-dir>/project-spec/meta-user/recipes-bsp/device-tree/files) and copy the content of “zynq-zed-adv7511-ad9364-fmcomms4.dts” and place at the end, as shown :



```
/include/ "system-conf.dtsi"
/ {
};
/dts-v1/;

/include/ "zynq-zed.dtsi"
/include/ "zynq-zed-adv7511.dtsi"

&fpga_axi {
        fmc_i2c: i2c@41620000 {
                compatible = "xlnx,axi-iic-1.01.b", "xlnx,xps-iic-2.00.a";
                reg = <0x41620000 0x10000>;
                interrupt-parent = <&intc>;
                interrupts = <0 55 0x4>;
                clocks = <&clkc 15>;
                clock-names = "pclk";

                #size-cells = <0>;
                #address-cells = <1>;
        };
```

ii. Delete lines as shown:

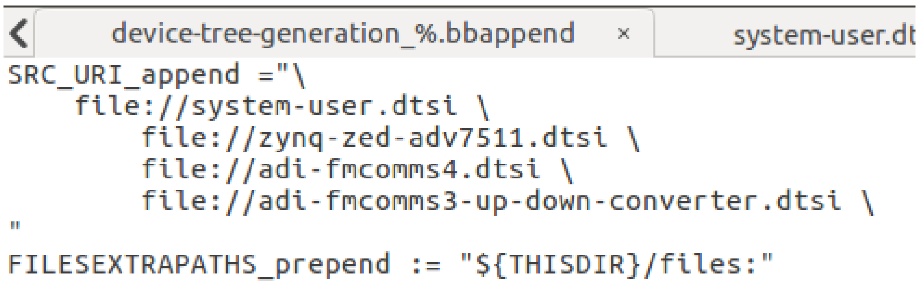


```
/include/ "system-conf.dtsi"
/ {
};


/include/ "zynq-zed-adv7511.dtsi"

&fpga_axi {
        fmc_i2c: i2c@41620000 {
                compatible = "xlnx,axi-iic-1.01.b", "xlnx,xps-iic-2.00.a";
                reg = <0x41620000 0x10000>;
                interrupt-parent = <&intc>;
                interrupts = <0 55 0x4>;
                clocks = <&clkc 15>;
                clock-names = "pclk";
```

iii. Copy the following dts files to (<plnx-proj-dir>/project-spec/meta-user/recipes-bsp/device-tree/files) – why those files, see include statements in “zynq-zed-adv7511-ad9364-fmcomms4.dts”.

a. **zynq-zed-adv7511.dtsi.**
b. **adi-fmcomms4.dtsi.**
c. **adi-fmcomms3-up-down-converter.dtsi.**

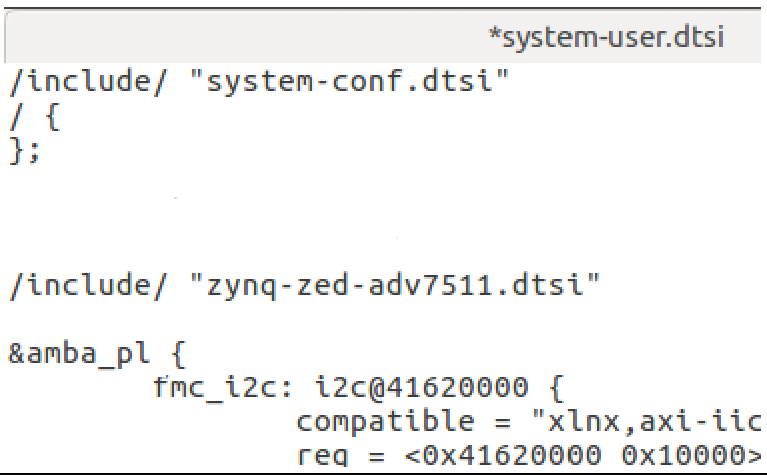
iv. Edit “device-tree-generation_%.bbappend” found in (<plnx-proj-dir>/project-spec/meta-user/recipes-bsp/device-tree) as shown:



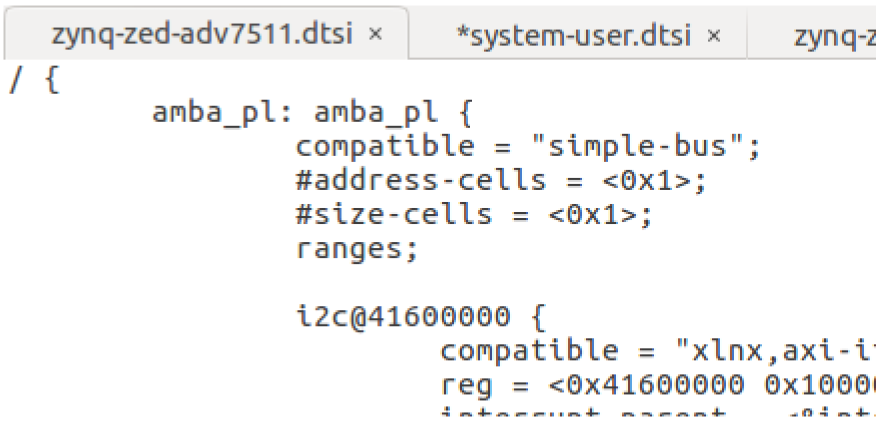
```
SRC_URI_append ="\
    file://system-user.dtsi \
        file://zynq-zed-adv7511.dtsi \
        file://adi-fmcomms4.dtsi \
        file://adi-fmcomms3-up-down-converter.dtsi \
"
FILESEXTRAPATHS_prepend := "${THISDIR}/files:"
```

Simply, we are including the previously copied files.

v. Replace “**fpga_axi**” in files (system-user.dtsi, zynq-zed-adv7511.dtsi) with “**amba_pl**”, in order to overwrite node entry that is auto-generated by petalinux in file “pl.dtsi” found in (<plnx-proj-dir>/components/plnx_workspace/device-tree/device-tree-generation) – **Note: you need to build the project to see the update pl.dtsi that reflect your recently imported hardware configuration.**



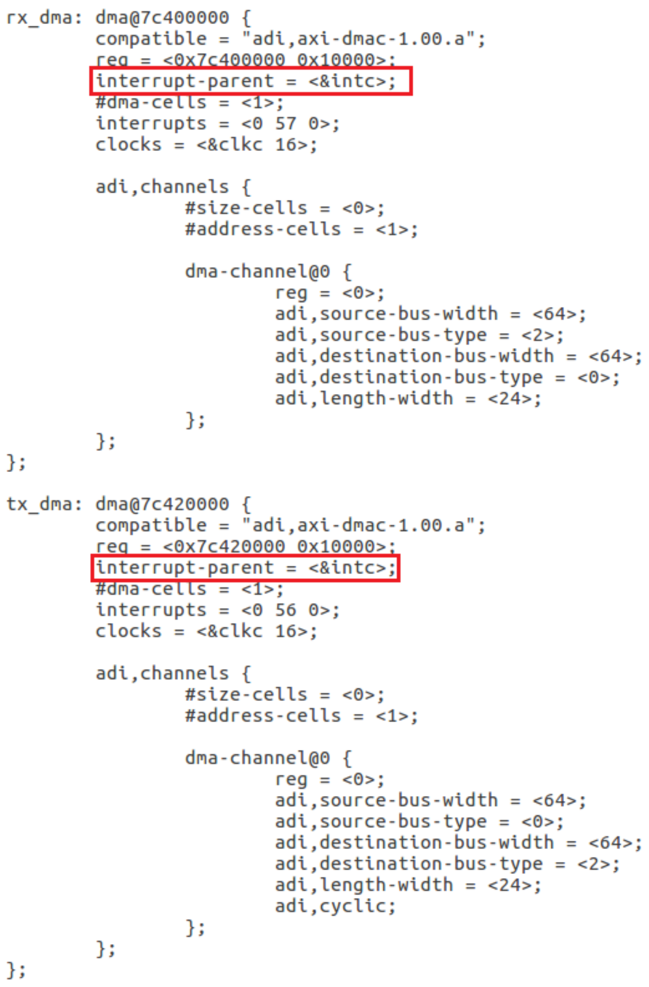
```
/include/ "system-conf.dtsi"
/ {
};



/include/ "zynq-zed-adv7511.dtsi"

&amba_pl {
        fmc_i2c: i2c@41620000 {
                compatible = "xlnx,axi-iic
                req = <0x41620000 0x10000>
```



```
/ {
        amba_pl: amba_pl {
                compatible = "simple-bus";
                #address-cells = <0x1>;
                #size-cells = <0x1>;
                ranges;

                i2c@41600000 {
                        compatible = "xlnx,axi-i
                        reg = <0x41600000 0x1000
```

vi. Add to rx_dma and tx_dma in file “system-user.dtsi” (see thread) the following:

```
interrupt-parent = <&intc>;
```



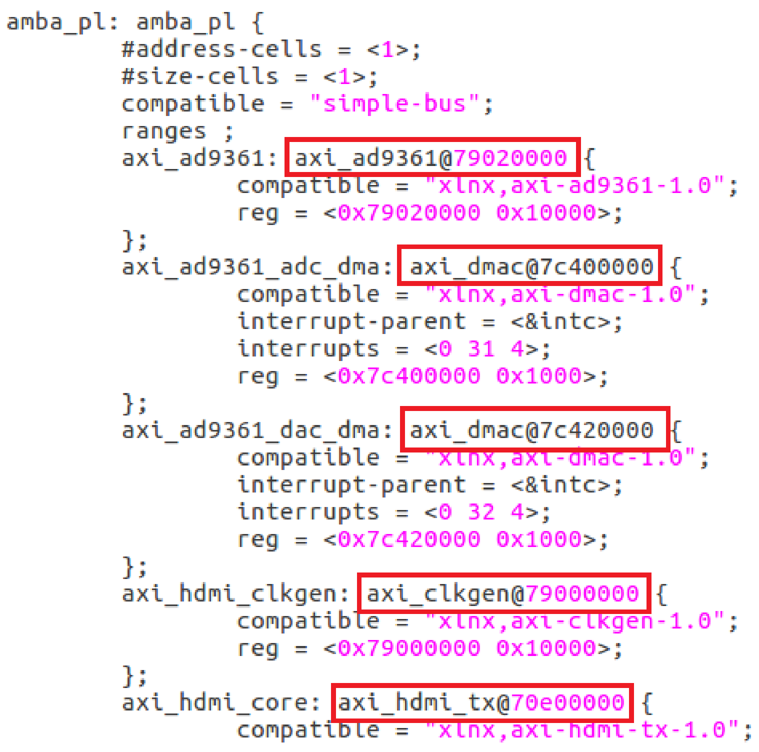
```
rx_dma: dma@7c400000 {
        compatible = "adi,axi-dmac-1.00.a";
        reg = <0x7c400000 0x10000>;
        interrupt-parent = <&intc>;
        #dma-cells = <1>;
        interrupts = <0 57 0>;
        clocks = <&clkc 16>;

        adi,channels {
                #size-cells = <0>;
                #address-cells = <1>;

                dma-channel@0 {
                        reg = <0>;
                        adi,source-bus-width = <64>;
                        adi,source-bus-type = <2>;
                        adi,destination-bus-width = <64>;
                        adi,destination-bus-type = <0>;
                        adi,length-width = <24>;
                };
        };
};

tx_dma: dma@7c420000 {
        compatible = "adi,axi-dmac-1.00.a";
        reg = <0x7c420000 0x10000>;
        interrupt-parent = <&intc>;
        #dma-cells = <1>;
        interrupts = <0 56 0>;
        clocks = <&clkc 16>;

        adi,channels {
                #size-cells = <0>;
                #address-cells = <1>;

                dma-channel@0 {
                        reg = <0>;
                        adi,source-bus-width = <64>;
                        adi,source-bus-type = <0>;
                        adi,destination-bus-width = <64>;
                        adi,destination-bus-type = <2>;
                        adi,length-width = <24>;
                        adi,cyclic;
                };
        };
};
```

vii. Delete unused auto-generated nodes in pl.dtsi file by passing device tree commands to “system-user.dtsi” file.

Since dts files, provided by ADI, take care of defining all hardware in programmable loigic (pl), we need to delete any auto-generated nodes by petalinux to avoid any unpredictable kernel behavior by adding the following lines in “system-user.dtsi”:

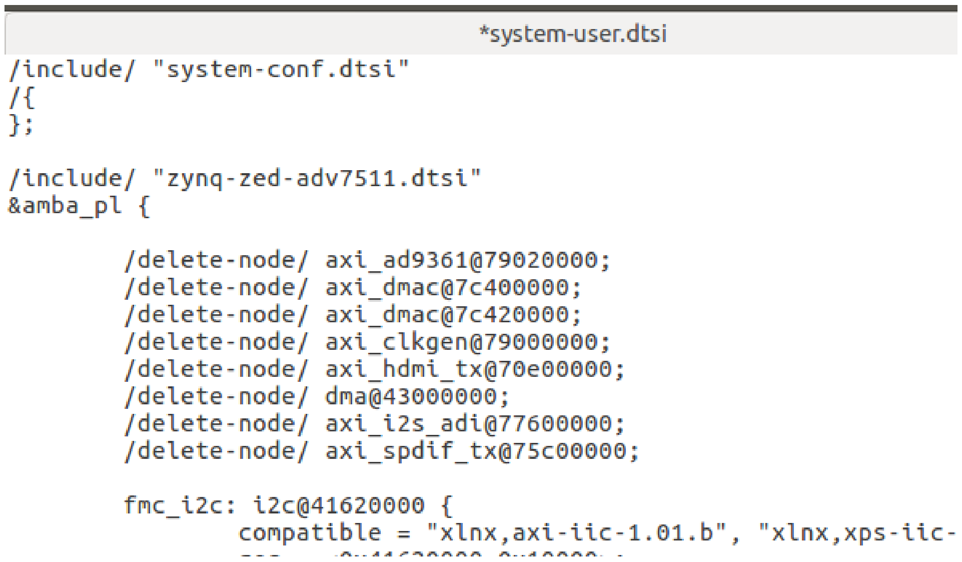
```
/delete-node/ axi_ad9361@79020000;

/delete-node/ axi_dmac@7c400000;

/delete-node/ axi_dmac@7c420000;

/delete-node/ axi_clkgen@79000000;

/delete-node/ axi_hdmi_tx@70e00000;

/delete-node/ dma@43000000;

/delete-node/ axi_i2s_adi@77600000;

/delete-node/ axi_spdif_tx@75c00000;
```

Nodes names are taken from pl.dtsi, see the following figure:


```
amba_pl: amba_pl {
        #address-cells = <1>;
        #size-cells = <1>;
        compatible = "simple-bus";
        ranges ;
        axi_ad9361: axi_ad9361@79020000 {
                compatible = "xlnx,axi-ad9361-1.0";
                reg = <0x79020000 0x10000>;
        };
        axi_ad9361_adc_dma: axi_dmac@7c400000 {
                compatible = "xlnx,axi-dmac-1.0";
                interrupt-parent = <&intc>;
                interrupts = <0 31 4>;
                reg = <0x7c400000 0x1000>;
        };
        axi_ad9361_dac_dma: axi_dmac@7c420000 {
                compatible = "xlnx,axi-dmac-1.0";
                interrupt-parent = <&intc>;
                interrupts = <0 32 4>;
                reg = <0x7c420000 0x1000>;
        };
        axi_hdmi_clkgen: axi_clkgen@79000000 {
                compatible = "xlnx,axi-clkgen-1.0";
                reg = <0x79000000 0x10000>;
        };
        axi_hdmi_core: axi_hdmi_tx@70e00000 {
                compatible = "xlnx,axi-hdmi-tx-1.0";
```

Note: the updated pl.dtsi that reflect your recently imported hardware won't be updated unless you build the project.

End result will be:


*system-user.dtsi

```
/include/ "system-conf.dtsi"
/{
};

/include/ "zynq-zed-adv7511.dtsi"
&amba_pl {

        /delete-node/ axi_ad9361@79020000;
        /delete-node/ axi_dmac@7c400000;
        /delete-node/ axi_dmac@7c420000;
        /delete-node/ axi_clkgen@79000000;
        /delete-node/ axi_hdmi_tx@70e00000;
        /delete-node/ dma@43000000;
        /delete-node/ axi_i2s_adi@77600000;
        /delete-node/ axi_spdif_tx@75c00000;

        fmc_i2c: i2c@41620000 {
                compatible = "xlnx,axi-iic-1.01.b", "xlnx,xps-iic-
```

4. Configure kernel:

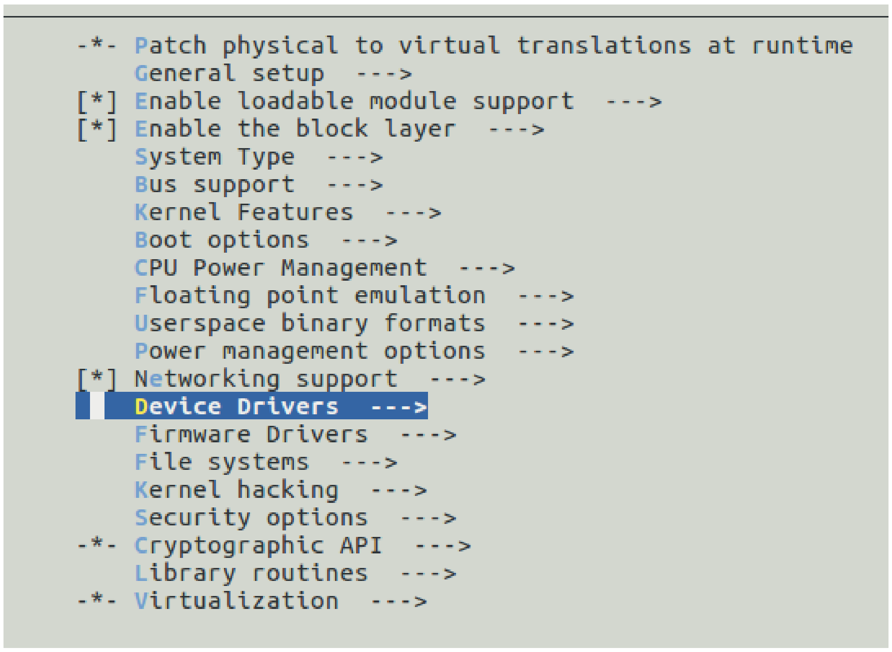
```
plnx-proj-dir$  petalinux-config -c kernel --defconfig zynq_xcomm_adv7511_defconfig
```

Note: The file "zynq_xcomm_adv7511_defconfig" will be loaded from <adi-kernel-root>/arch/arm/configs.

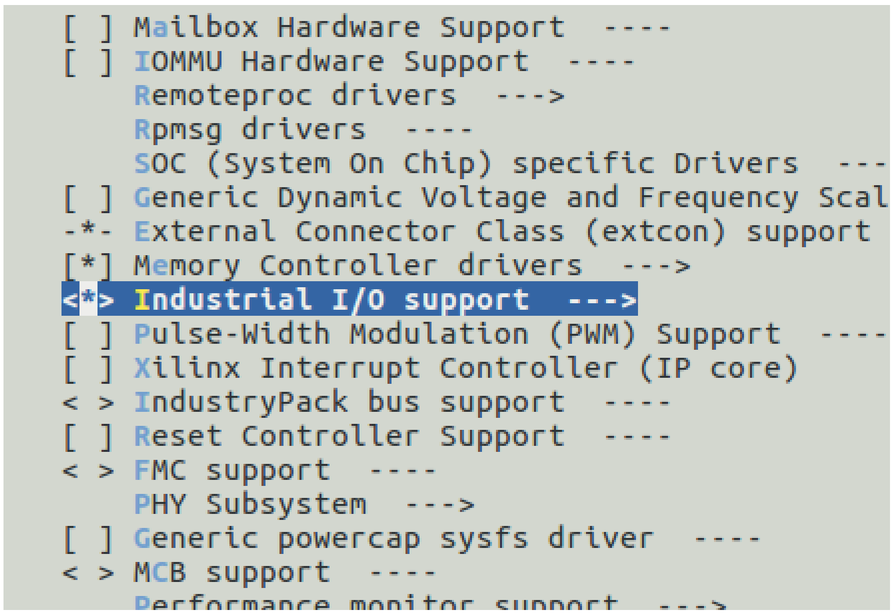
It is good idea to double check if kernel configured correctly by running:

```
plnx-proj-dir$  petalinux-config -c kernel
```

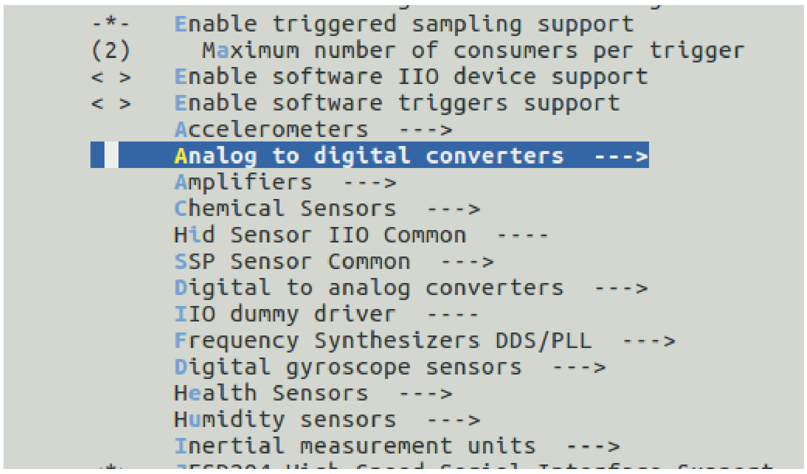
[Device Drivers >> Industrial I/O support >> Analog to Digital Converter >> AD9361/AD9364] is checked.
Note: There are times I had to re-configure kernel (repeat step 4) after configuring rootfs because kernel lost its configuration!



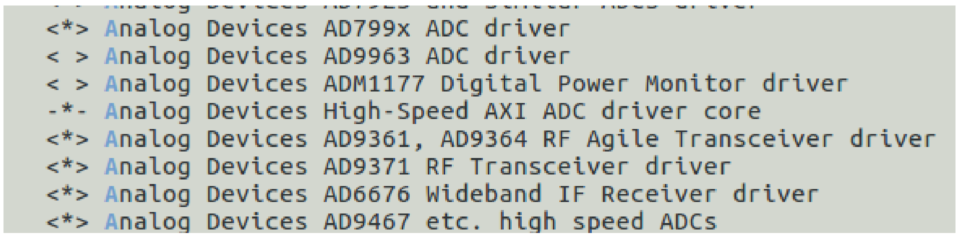
```
    -*- Patch physical to virtual translations at runtime
        General setup  --->
    [*] Enable loadable module support  --->
    [*] Enable the block layer  --->
        System Type  --->
        Bus support  --->
        Kernel Features  --->
        Boot options  --->
        CPU Power Management  --->
        Floating point emulation  --->
        Userspace binary formats  --->
        Power management options  --->
    [*] Networking support  --->
        Device Drivers  --->
        Firmware Drivers  --->
        File systems  --->
        Kernel hacking  --->
        Security options  --->
    -*- Cryptographic API  --->
        Library routines  --->
    -*- Virtualization  --->
```



```
  [ ] Mailbox Hardware Support  ----
  [ ] IOMMU Hardware Support  ----
      Remoteproc drivers  --->
      Rpmsg drivers  ----
      SOC (System On Chip) specific Drivers  ---
  [ ] Generic Dynamic Voltage and Frequency Scal
  -*- External Connector Class (extcon) support
  [*] Memory Controller drivers  --->
  <*> Industrial I/O support  --->
  [ ] Pulse-Width Modulation (PWM) Support  ----
  [ ] Xilinx Interrupt Controller (IP core)
  < > IndustryPack bus support  ----
  [ ] Reset Controller Support  ----
  < > FMC support  ----
      PHY Subsystem  --->
  [ ] Generic powercap sysfs driver  ----
  < > MCB support  ----
      Performance monitor support  --->
```



```
    -*-   Enable triggered sampling support
    (2)     Maximum number of consumers per trigger
    < >   Enable software IIO device support
    < >   Enable software triggers support
          Accelerometers  --->
          Analog to digital converters  --->
          Amplifiers  --->
          Chemical Sensors  --->
          Hid Sensor IIO Common  ----
          SSP Sensor Common  --->
          Digital to analog converters  --->
          IIO dummy driver  ----
          Frequency Synthesizers DDS/PLL  --->
          Digital gyroscope sensors  --->
          Health Sensors  --->
          Humidity sensors  --->
          Inertial measurement units  --->
```



```
<*> Analog Devices AD799x ADC driver
< > Analog Devices AD9963 ADC driver
< > Analog Devices ADM1177 Digital Power Monitor driver
-*- Analog Devices High-Speed AXI ADC driver core
<*> Analog Devices AD9361, AD9364 RF Agile Transceiver driver
<*> Analog Devices AD9371 RF Transceiver driver
<*> Analog Devices AD6676 Wideband IF Receiver driver
<*> Analog Devices AD9467 etc. high speed ADCs
```

5. **Build the project:**

```
plnx-proj-dir$ petalinux-build
```

Note: I personally use sstate cache, which can be downloaded from Xilinx, in order build the project without internet (faster for me). You need internet later in the step of adding **libiio** to petalinux rootfs.

6. **Boot the petalinux image.**
There are different ways to boot petalinux image on hardware. I personally use tftpboot during embedded linux development. See ug1144(v2017.4) to setup tftpboot.
You need also to setup tftp server see the following link here or search "setup tftpboot"

Note: You can skip Package Prebuilt image and in step 8 (ug1144-v2017.4 – page 37) run the following command instead:

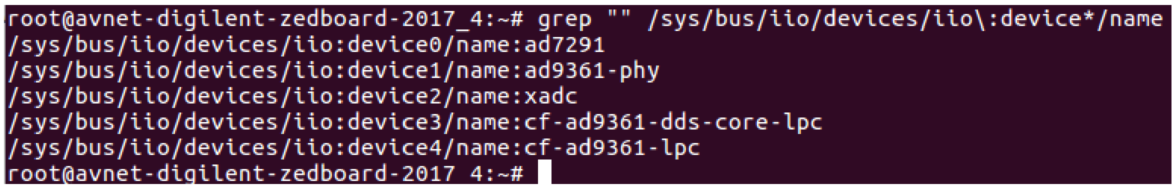
```
plnx-proj-dir$ petalinux-boot --jtag --fpga --u-boot
```

It is going to download the following
- **bitstream** <plnx-proj-root>/images/linux/system_top.bit
- **fsbl** <plnx-proj-root>/images/linux/zynq_fsbl.elf
- **U-Boot** <plnx-proj-root>/images/linux/u-boot.elf.

Follow the remaining steps in ug1144(v2017.4) page 37.

7. **Check IIO devices are found**, in target linux terminal run the following:

```
root@avnet-digilent-zedboard-2017_4:~#  grep "" /sys/bus/iio/devices/iio\:device*/name
```



```
root@avnet-digilent-zedboard-2017_4:~# grep ""  /sys/bus/iio/devices/iio\:device*/name
/sys/bus/iio/devices/iio:device0/name:ad7291
/sys/bus/iio/devices/iio:device1/name:ad9361-phy
/sys/bus/iio/devices/iio:device2/name:xadc
/sys/bus/iio/devices/iio:device3/name:cf-ad9361-dds-core-lpc
/sys/bus/iio/devices/iio:device4/name:cf-ad9361-lpc
root@avnet-digilent-zedboard-2017_4:~# 
```

8. To connect remotely to IIO Oscilloscope we need to **install libiio** to our root file system.
   a. Create a new app:

```
plnx-proj-dir$ petalinux-create -t apps -n libiio --template install --enable
```

   The app will be at (<plnx-proj-dir>/project-spec/meta-user/recipes-apps/libiio).

   b. Edit libiio.bb (download script) and copy the following:

```
DESCRIPTION = "Analog Devices Libiio"
LICENSE = "LGPLv2.1"

LIC_FILES_CHKSUM = "file://COPYING.txt;md5=7c13b3376cea0ce68d2d2da0a1b3a72c"

SRCREV = "6ecff5d46e1b12c2859f0b63a73282940e3402bb"
SRCBRANCH = "master"

PVBASE := "${PV}"

FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}-${PVBASE}:"

PV = "${PVBASE}.${SRCPV}"

SRC_URI = "git://github.com/analogdevicesinc/libiio.git;protocol=https;branch=${SRCBRANCH}"

S = "${WORKDIR}/git"

inherit autotools cmake

DEPENDS += "libxml2 bison flex ncurses avahi"

PARALLEL_MAKE = ""
```

It is very important to change SRCREV if new commit have been made:

- Browse to Analog Device Libiio github.
  https://github.com/analogdevicesinc/libiio
- Make sure the branch selected is “master” or else as long as you change it from the code (SRCBRANCH = "master")
- Click commits.
- Choose the most recent commit.
- Copy hash on the link and paste it in SRCREV.

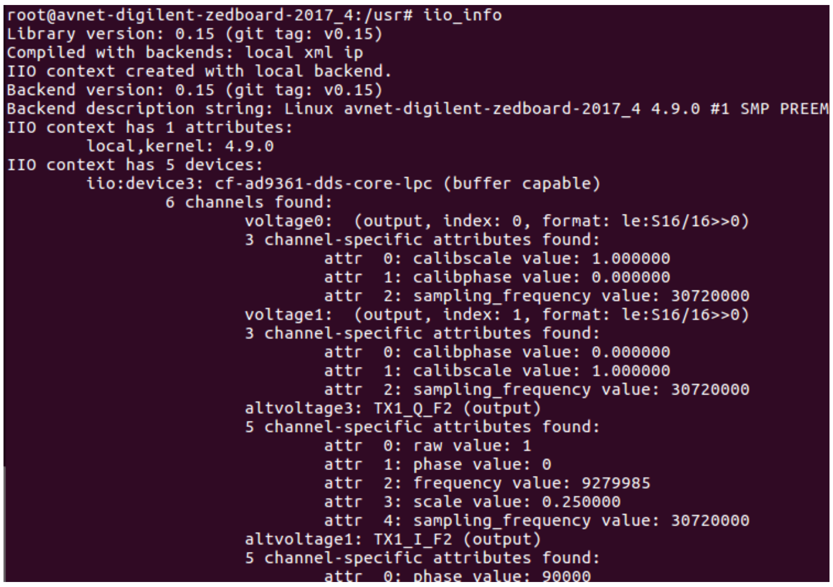
https://github.com/analogdevicesinc/libiio/commit/6ecff5d46e1b12c2859f0b63a73282940e3402bb

9. Build the project:

```
plnx-proj-dir$ petalinux-build
```

10. Boot the petalinux image (step 6).

11. Type “iio_info” or “iiod --version” to check if libiio is installed successfully.



```
root@avnet-digilent-zedboard-2017_4:/usr# iio_info
Library version: 0.15 (git tag: v0.15)
Compiled with backends: local xml ip
IIO context created with local backend.
Backend version: 0.15 (git tag: v0.15)
Backend description string: Linux avnet-digilent-zedboard-2017_4 4.9.0 #1 SMP PREEM
IIO context has 1 attributes:
        local,kernel: 4.9.0
IIO context has 5 devices:
        iio:device3: cf-ad9361-dds-core-lpc (buffer capable)
                6 channels found:
                        voltage0:  (output, index: 0, format: le:S16/16>>0)
                        3 channel-specific attributes found:
                                attr  0: calibscale value: 1.000000
                                attr  1: calibphase value: 0.000000
                                attr  2: sampling_frequency value: 30720000
                        voltage1:  (output, index: 1, format: le:S16/16>>0)
                        3 channel-specific attributes found:
                                attr  0: calibphase value: 0.000000
                                attr  1: calibscale value: 1.000000
                                attr  2: sampling_frequency value: 30720000
                        altvoltage3: TX1_Q_F2 (output)
                        5 channel-specific attributes found:
                                attr  0: raw value: 1
                                attr  1: phase value: 0
                                attr  2: frequency value: 9279985
                                attr  3: scale value: 0.250000
                                attr  4: sampling_frequency value: 30720000
                        altvoltage1: TX1_I_F2 (output)
                        5 channel-specific attributes found:
                                attr  0: phase value: 90000
```

12. **Connecting** our device (ad9364) remotely **to IIO Oscilloscope.**

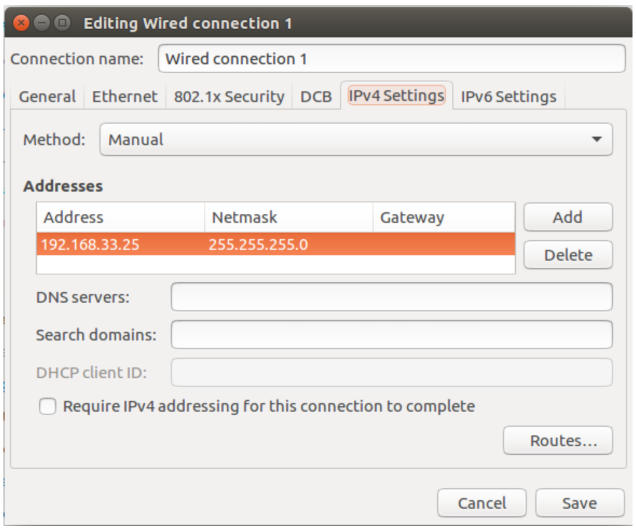
a. Establish a network between target Linux and host PC.

  i. Connect Ethernet cable.

  ii. Configure IP for the target Linux:

```
root@avnet-digilent-zedboard-2017_4:/# ifconfig eth0 192.168.33.30 up
```

  iii. Configure IP for the host PC.


Editing Wired connection 1

Connection name: Wired connection 1

Method: Manual

Addresses

Address	Netmask	Gateway
192.168.33.25	255.255.255.0	

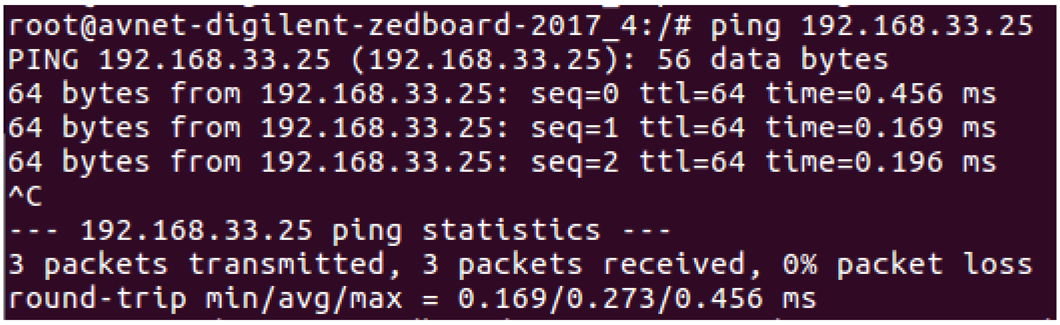
DNS servers:

Search domains:

DHCP client ID:

☐ Require IPv4 addressing for this connection to complete

  iv. Ping host IP:

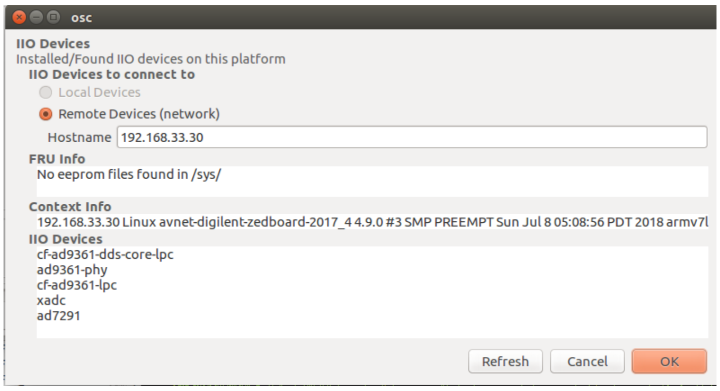

```
root@avnet-digilent-zedboard-2017_4:/# ping 192.168.33.25
PING 192.168.33.25 (192.168.33.25): 56 data bytes
64 bytes from 192.168.33.25: seq=0 ttl=64 time=0.456 ms
64 bytes from 192.168.33.25: seq=1 ttl=64 time=0.169 ms
64 bytes from 192.168.33.25: seq=2 ttl=64 time=0.196 ms
^C
--- 192.168.33.25 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.169/0.273/0.456 ms
```

b. Install **IIO Oscilloscope** on host PC and run it.
See: https://wiki.analog.com/resources/tools-software/linux-software/iio_oscilloscope

c. Start **IIOD Daemon**. Type in target Linux virtual terminal:

```
root@avnet-digilent-zedboard-2017_4:/# iiod &
```

d. Connect to target hardware.

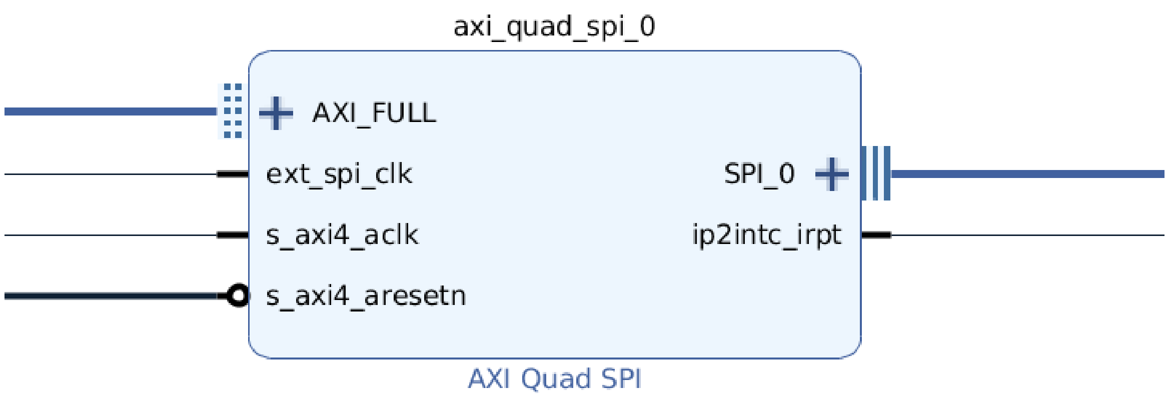

osc

IIO Devices
Installed/Found IIO devices on this platform
IIO Devices to connect to
○ Local Devices
● Remote Devices (network)
Hostname 192.168.33.30
FRU Info
No eeprom files found in /sys/
Context Info
192.168.33.30 Linux avnet-digilent-zedboard-2017_4 4.9.0 #3 SMP PREEMPT Sun Jul 8 05:08:56 PDT 2018 armv7l
IIO Devices
cf-ad9361-dds-core-lpc
ad9361-phy
cf-ad9361-lpc
xadc
ad7291

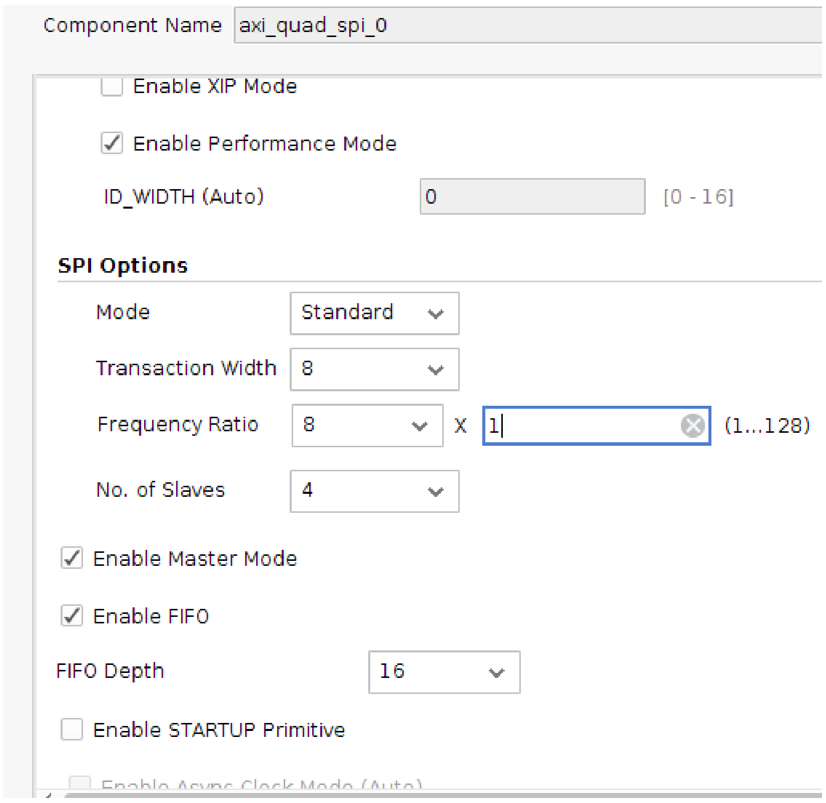
Adding AXI Quad SPI to ADI HDL and using SPIDEV Userspace Driver.

A. Modifying HDL.

1. Open up the original HDL design found here (FMCOMMS4+Zed).
2. Open up the Block Design and add AXI Quad SPI IP.



3. Edit AXI Quad SPI IP for your preferred use, mine is as follows:



Component Name axi_quad_spi_0

☐ Enable XIP Mode

✓ Enable Performance Mode

ID_WIDTH (Auto) 0 [0 - 16]

SPI Options

Mode	Standard
Transaction Width	8
Frequency Ratio	8 X 1 (1...128)
No. of Slaves	4

✓ Enable Master Mode

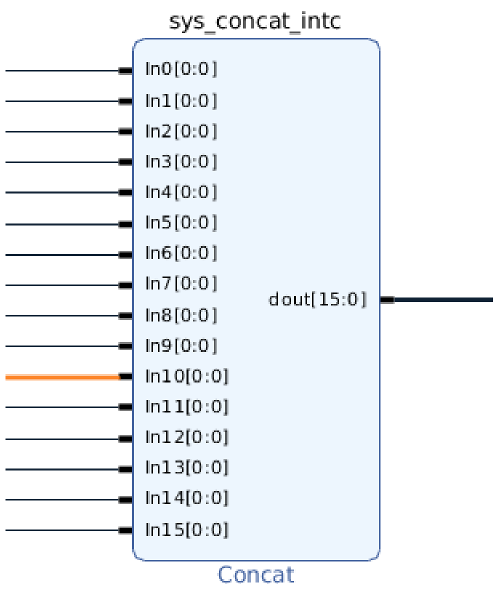
✓ Enable FIFO

FIFO Depth 16

☐ Enable STARTUP Primitive

☐ Enable Async Clock Mode (Auto)

4. Make the necessary connections by running "Run Connection Automation".
5. Connect (ip2intc_irqt) of axi_quad_spi ip to irq bus **in10** as shown:



It is very important to know exactly where the irq signal of the ip is connected (interrupt ID) and reflect that value in the device tree later on or you might get transfer timed out issues when trying to send and receive.

IRQ_F2P[15:0] of ZYNQ7 Processing System has the following ID [91:84] ,[68:61] with MSB of IRQ bus (sys_concat_intc in HDL block design) assigned to highest interrupt ID of 91.

6. Make a new constraint and connect SPI signal to whatever I/O pin you like:

```
set_property PACKAGE_PIN H15 [get_ports {spi_rtl_ss_io[0]}]
set_property IOSTANDARD LVCMOS25 [get_ports {spi_rtl_ss_io[0]}]
set_property PACKAGE_PIN K15 [get_ports {spi_rtl_ss_io[1]}]
set_property IOSTANDARD LVCMOS25 [get_ports {spi_rtl_ss_io[1]}]

# "XADC-AD0N-R"
set_property PACKAGE_PIN E16 [get_ports {spi_rtl_ss_io[2]}]
set_property IOSTANDARD LVCMOS25 [get_ports {spi_rtl_ss_io[2]}]

# "XADC-AD8N-R"
set_property PACKAGE_PIN D17 [get_ports {spi_rtl_ss_io[3]}]
set_property IOSTANDARD LVCMOS25 [get_ports {spi_rtl_ss_io[3]}]

# MOSI
set_property PACKAGE_PIN R15 [get_ports spi_rtl_io0_io]
set_property IOSTANDARD LVCMOS25 [get_ports spi_rtl_io0_io]

set_property PACKAGE_PIN J15 [get_ports spi_rtl_sck_io]
set_property IOSTANDARD LVCMOS25 [get_ports spi_rtl_sck_io]
```

Note: I have dropped spi_rtl_io1_io (MISO signal) , user need to modify the generated hdl files by commenting unused lines, also user need to modify zed_system_constr.xdc and comment used pins by the early added constraint file. In my case:

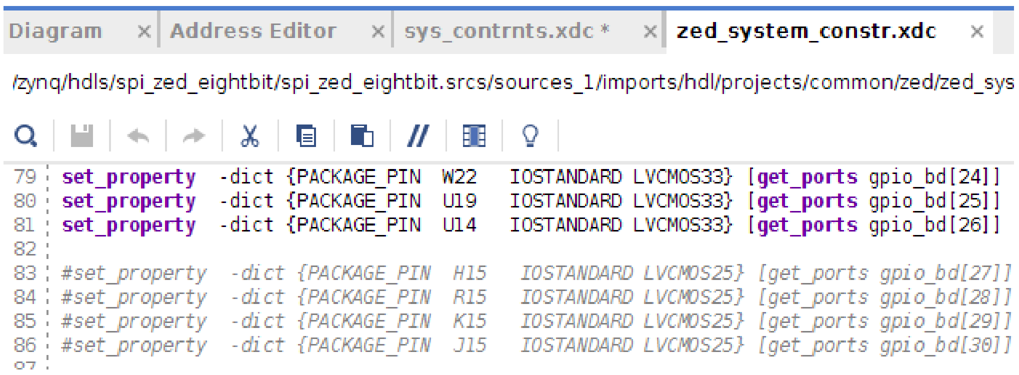


Diagram
Address Editor
sys_contrnts.xdc *
zed_system_constr.xdc
/zynq/hdls/spi_zed_eightbit/spi_zed_eightbit.srcs/sources_1/imports/hdl/projects/common/zed/zed_sys

```
set_property  -dict {PACKAGE_PIN  W22   IOSTANDARD LVCMOS33} [get_ports gpio_bd[24]]
set_property  -dict {PACKAGE_PIN  U19   IOSTANDARD LVCMOS33} [get_ports gpio_bd[25]]
set_property  -dict {PACKAGE_PIN  U14   IOSTANDARD LVCMOS33} [get_ports gpio_bd[26]]

#set_property  -dict {PACKAGE_PIN  H15   IOSTANDARD LVCMOS25} [get_ports gpio_bd[27]]
#set_property  -dict {PACKAGE_PIN  R15   IOSTANDARD LVCMOS25} [get_ports gpio_bd[28]]
#set_property  -dict {PACKAGE_PIN  K15   IOSTANDARD LVCMOS25} [get_ports gpio_bd[29]]
#set_property  -dict {PACKAGE_PIN  J15   IOSTANDARD LVCMOS25} [get_ports gpio_bd[30]]
```

7. Generate bit stream and export HDL.

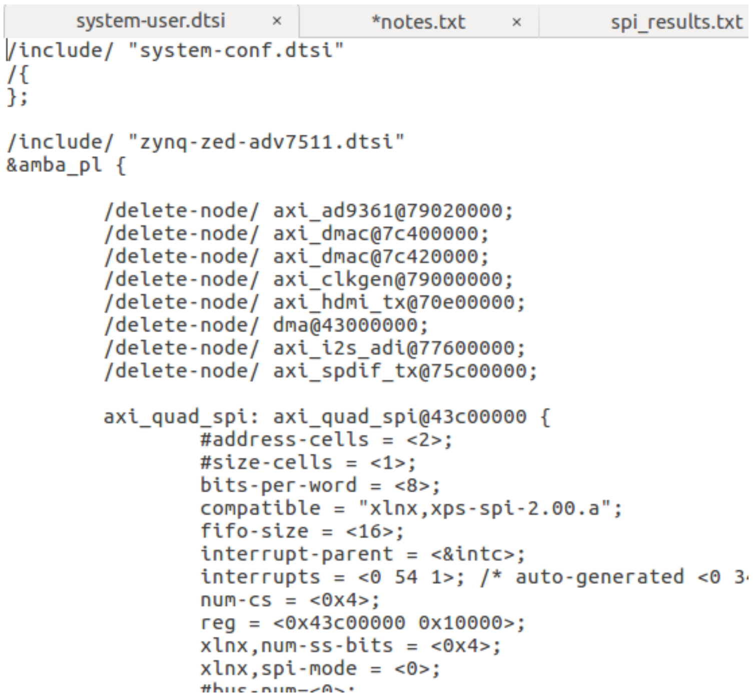
B. Building kernel.

1. Create a new Petalinux project and import recently generated hardware (HDF) [step 1, 2 in embedded Linux].

   Note: ADI xcomm_zynq (4.9) is used as an external Linux kernel in this design.

2. Configure device tree as in step 3 in embedded Linux development and add AXI Quad SPI node definition to system-user.dtsi file (see appendix A or download dts files here).

   Note: AXI Quad SPI node definition in system-user.dtsi will override the node definition in pl.dtsi, you can always double check by taking a look at plnx_arm-system.dts found on (<plnx-proj dir>/components/plnx_workspace/device-tree/device-tree-generation).



```
/include/ "system-conf.dtsi"
/{
};

/include/ "zynq-zed-adv7511.dtsi"
&amba_pl {

        /delete-node/ axi_ad9361@79020000;
        /delete-node/ axi_dmac@7c400000;
        /delete-node/ axi_dmac@7c420000;
        /delete-node/ axi_clkgen@79000000;
        /delete-node/ axi_hdmi_tx@70e00000;
        /delete-node/ dma@43000000;
        /delete-node/ axi_i2s_adi@77600000;
        /delete-node/ axi_spdif_tx@75c00000;

        axi_quad_spi: axi_quad_spi@43c00000 {
                #address-cells = <2>;
                #size-cells = <1>;
                bits-per-word = <8>;
                compatible = "xlnx,xps-spi-2.00.a";
                fifo-size = <16>;
                interrupt-parent = <&intc>;
                interrupts = <0 54 1>; /* auto-generated <0 3
                num-cs = <0x4>;
                reg = <0x43c00000 0x10000>;
                xlnx,num-ss-bits = <0x4>;
                xlnx,spi-mode = <0>;
```

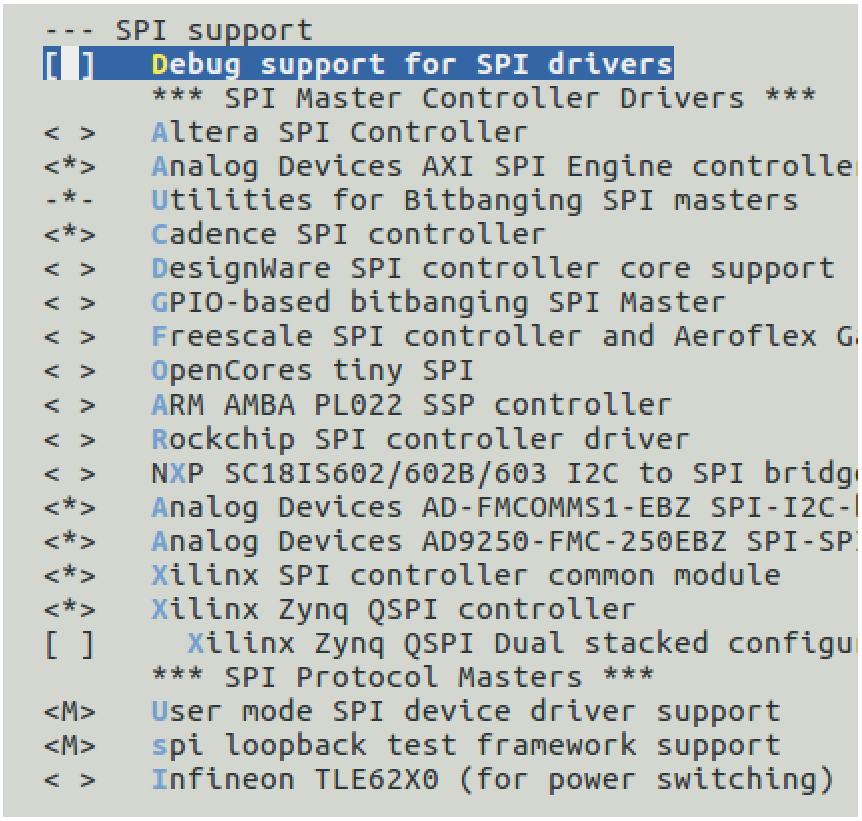
IRQ # in vivado	91 MSB	90	89	88	87	86	85	84	68	67	66	65	64	63	62	61 LSB
IRQ # in device-tree	59	58	57	56	55	54	53	52	36	35	34	33	32	31	30	29
	HDMI VDMA	I2C main	ADC RX DMA	DAC TX DMA	I2C FMC	AXI Quad SPI	U	U	U	U	U	U	U	U	U	U

Table 1: IRQ ID numbers in zynq7. U for unused.

It is also very important to understand interrupt specifier in device tree and its relation to INTC (interrupt controller) of zynq7 processing system in Vivado tool (see table 1). There is a 32 difference between IRQ ID in vivado and device tree (see this link). User shouldn't rely on auto generated device tree which provided an interrupt specifier that won't work well with analog device dts files. We have to change the interrupt specifier to (54). We would face a timed out issue with our SPI if we don't match device tree interrupt information with our hardware.

3. Configure kernel [step 4 in embedded Linux].

   Make sure (user mode SPI device driver support) is selected.
   Device Drivers -> SPI support -> user mode SPI device driver support.



```
--- SPI support
[ ]   Debug support for SPI drivers
      *** SPI Master Controller Drivers ***
< >   Altera SPI Controller
<*>   Analog Devices AXI SPI Engine controlle
-*-   Utilities for Bitbanging SPI masters
<*>   Cadence SPI controller
< >   DesignWare SPI controller core support
< >   GPIO-based bitbanging SPI Master
< >   Freescale SPI controller and Aeroflex G
< >   OpenCores tiny SPI
< >   ARM AMBA PL022 SSP controller
< >   Rockchip SPI controller driver
< >   NXP SC18IS602/602B/603 I2C to SPI bridg
<*>   Analog Devices AD-FMCOMMS1-EBZ SPI-I2C-
<*>   Analog Devices AD9250-FMC-250EBZ SPI-SP
<*>   Xilinx SPI controller common module
<*>   Xilinx Zynq QSPI controller
[ ]     Xilinx Zynq QSPI Dual stacked configu
      *** SPI Protocol Masters ***
<M>   User mode SPI device driver support
<M>   spi loopback test framework support
< >   Infineon TLE62X0 (for power switching)
```

4. Patch Xilinx SPI Linux driver.

   I faced a trouble (bug) related to SPI device setup (error -13) when booting xcomm_zynq (4.9), the only walk around to solve the issue is by using the older version of spi-xilinx.c from xcomm_zynq_4_6.

   - Replace spi-xilinx.c found in "<adi-kernel-root>/drivers/spi/" with spi-xilinx.c from ADI kernel "xcomm_zynq_4_6" or download it here.

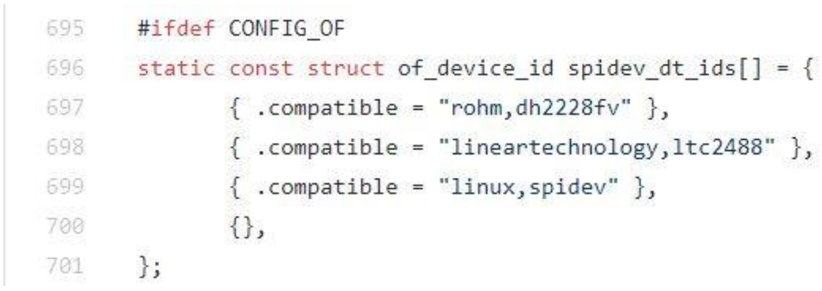
   Also, user would face the warning "buggy DT: spidev listed directly in DT" because of a compatibility we provided in device tree file (system-user.dtsi):

```
compatible = "linux,spidev";
```

   We need to modify spidev.c and add in line 699 the following or download file here:

```
        { .compatible = "linux,spidev" },
```

As shown here:



```
#ifdef CONFIG_OF
static const struct of_device_id spidev_dt_ids[] = {
        { .compatible = "rohm,dh2228fv" },
        { .compatible = "lineartechnology,ltc2488" },
        { .compatible = "linux,spidev" },
        {},
};
```

NOTE: If you modify kernel source we need to run the following command in petalinux and then configure kernel again (step 3) before building.

```
plnx-proj-dir$ petalinux-build –x mrproper
```

So the project can build the newly modified sources from our kernel (spi-xilinx.c, spidev.c). If you don't modify them or any of the kernel, the command could be skipped.

5. Creating Linux app to access AXI Quad SPI.

   Our app will use spidev which is a userspace interface to SPI devices.

   - Create a new app:

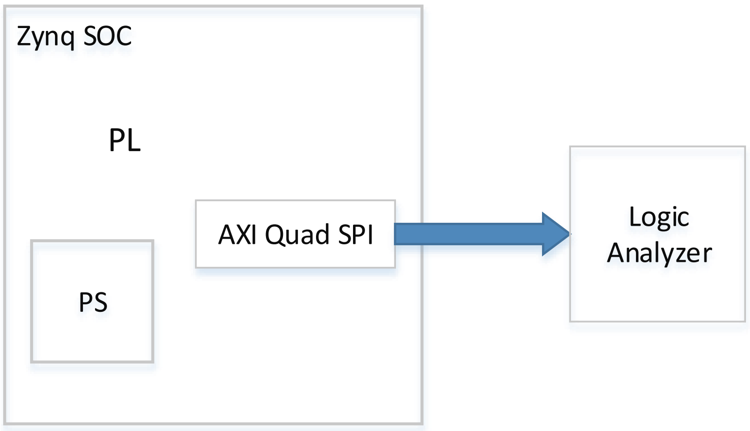
```
plnx-proj-dir$  petalinux-create -t apps -n spidev-test --enable
```

   - Copy the content in "<adi-kernel-root>/tools/spi/spidev_test.c" or download source here to "<plnx-proj-dir>/project-spec/meta-user/recipes-apps/spidev-test/files/spidev-test.c".
   - User could change code for his preferred usage, such as default device and/or data to be transmitted.

6. Build the petalinux project.

C. Measurements:

I have connected the AXI Quad SPI signal with 4 chips select to a Digital logic analyzer.



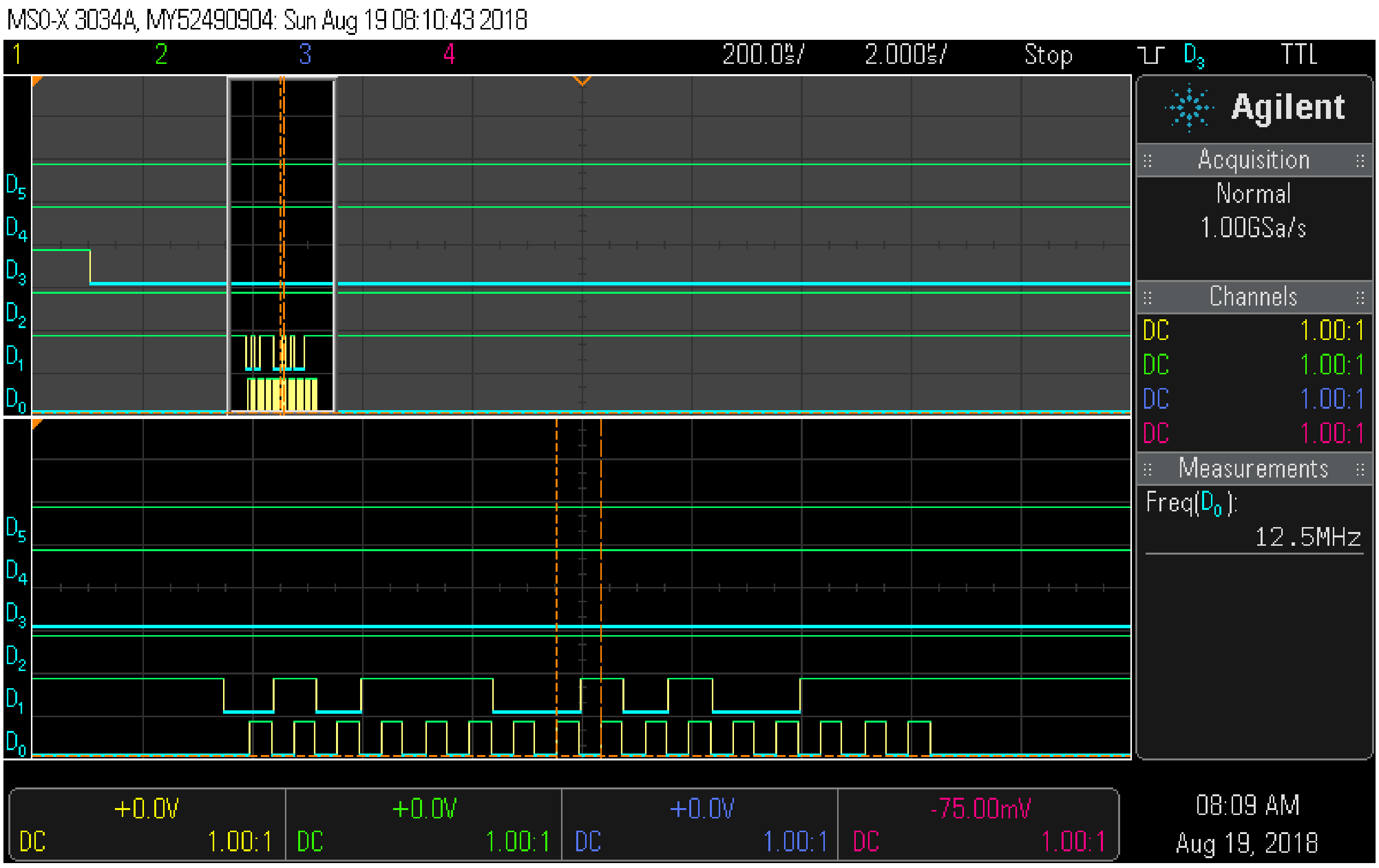
1. Boot linux image (see step 6 in embedded Linux).

2. Run the following command:

```
root@avnet-digilent-zedboard-2017_4:/# spidev-test –D /dev/spidev1.1 –p “\x5C\xA7” –v
```

The command shall do the following:

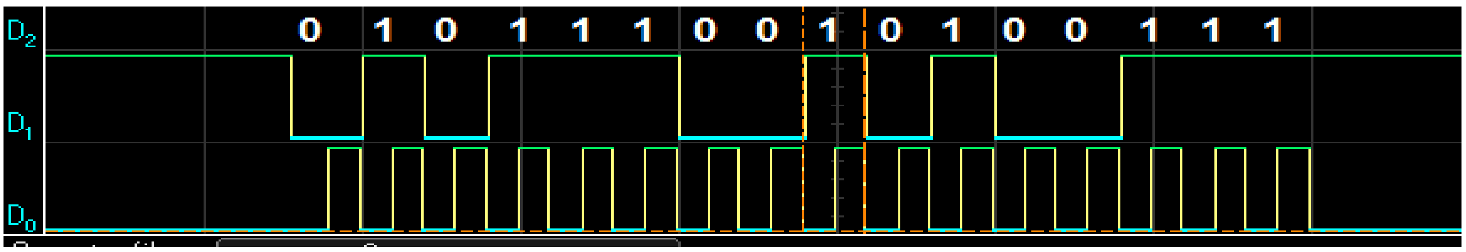
- Select device (CS1) from four devices we have (CS0, CS1, CS2, and CS3).
- Transmit 16 bit word (0x5CA7).


MSO-X 3034A, MY52490904: Sun Aug 19 08:10:43 2018

D0: Clock
D1: MOSI (Data).
D2: chip select 0.
D3: chip select 1.
D4: chip select 2.
D5: chip select 3.

Hex	5				C				A				7			
Bin	0	1	0	1	1	1	0	0	1	0	1	0	0	1	1	1

We can check that the transmitted word is exactly what we have sent and chip select 1 is active low. Also, the frequency is exactly what we have configured it during vivado design – 100 MHz / 8.



Appendix A

```
axi_quad_spi: axi_quad_spi@43c00000 {
                #address-cells = <2>;
                #size-cells = <1>;
                bits-per-word = <8>;
                compatible = "xlnx,xps-spi-2.00.a";
                fifo-size = <16>;
                interrupt-parent = <&intc>;
                interrupts = <0 54 1>; /* auto-generated <0 34 1>; */
                num-cs = <0x4>;
                reg = <0x43c00000 0x10000>;
                xlnx,num-ss-bits = <0x4>;
                xlnx,spi-mode = <0>;
                #bus-num=<0>;
                status = "okay";
                device0: device0@0 {
                        #address-cells = <1>;
                        #size-cells = <0>;
                        reg = <0>;
                        compatible = "linux,spidev";
                         spi-max-frequency = <10000000>;

                };
                device1: device1@1 {
                        #address-cells = <1>;
                        #size-cells = <0>;
                        reg = <1>;
                        compatible = "linux,spidev";
                         spi-max-frequency = <10000000>;

                };
                device2: device2@2 {
                        #address-cells = <1>;
                        #size-cells = <0>;
                        reg = <2>;
                        compatible = "linux,spidev";
                         spi-max-frequency = <10000000>;

                };
                device3: device3@3 {
                        #address-cells = <1>;
                        #size-cells = <0>;
                        reg = <3>;
                        compatible = "linux,spidev";
                         spi-max-frequency = <10000000>;

                };

        };
```