Developing embedded Linux for AD-FMCOMMS4-EZB and Zed boards 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 get HDF file by generating it from Vivado tool.

   To initialize a Petalinux project with 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:

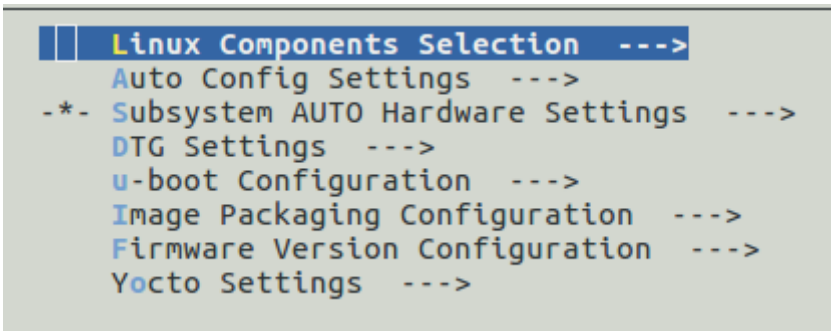
```
$ git clone -b xcomm_zynq --single-branch https://github.com/analogdevicesinc/linux.git
```

   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.



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:

device-tree-generation_%.bbappend × system-user.dt

```
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 it.**

*system-user.dtsi

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


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

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

---

zynq-zed-adv7511.dtsi × *system-user.dtsi × zynq-z

```
/ {
        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;
                };
        };
};
```

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.**

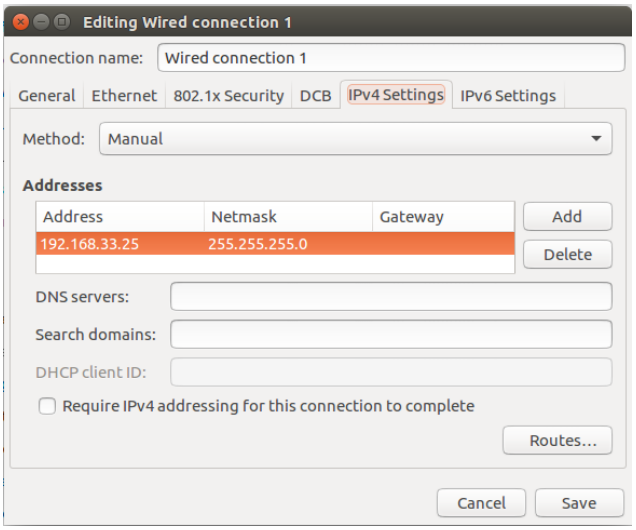
a. Establish a network between target Linux and host PC.

i. Connect Ethernet cable.

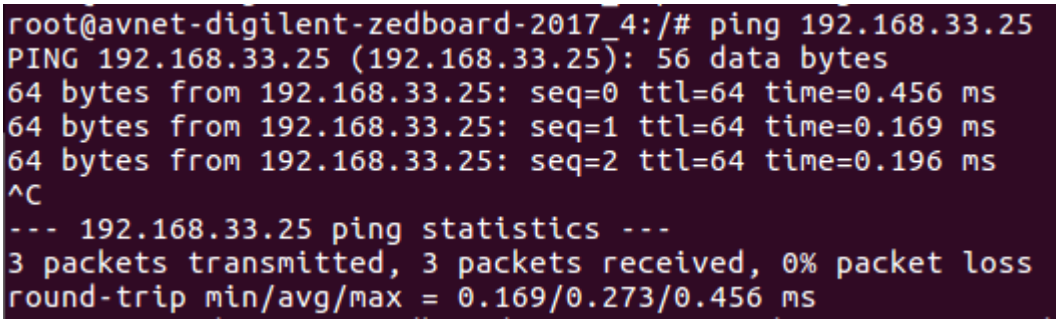
ii. Configure IP for the target Linux:

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

iii. Configure IP for the host PC.



iv. Ping host IP:



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.