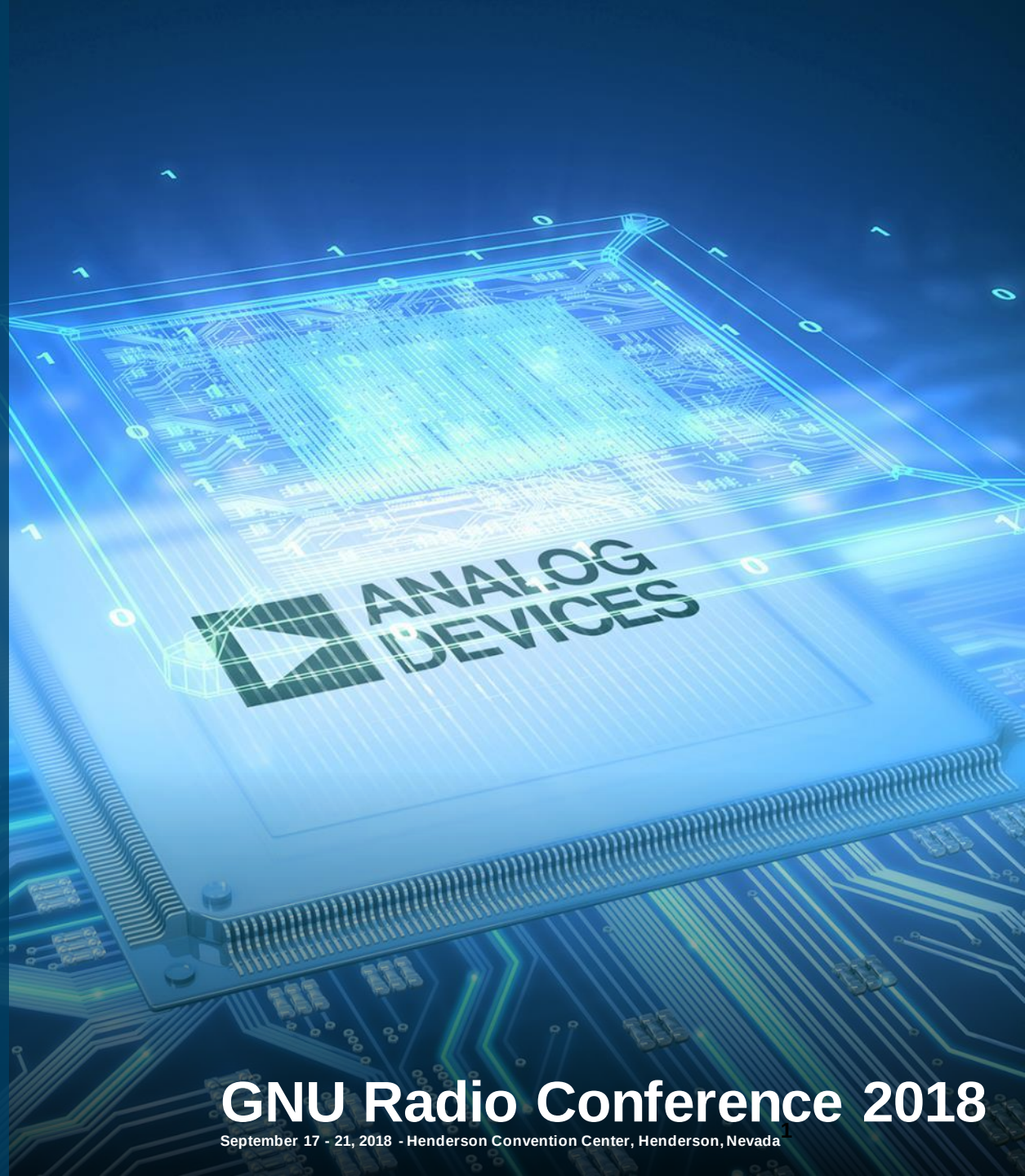
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

GNURadio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

ARM PlutoSDR With Custom Applications

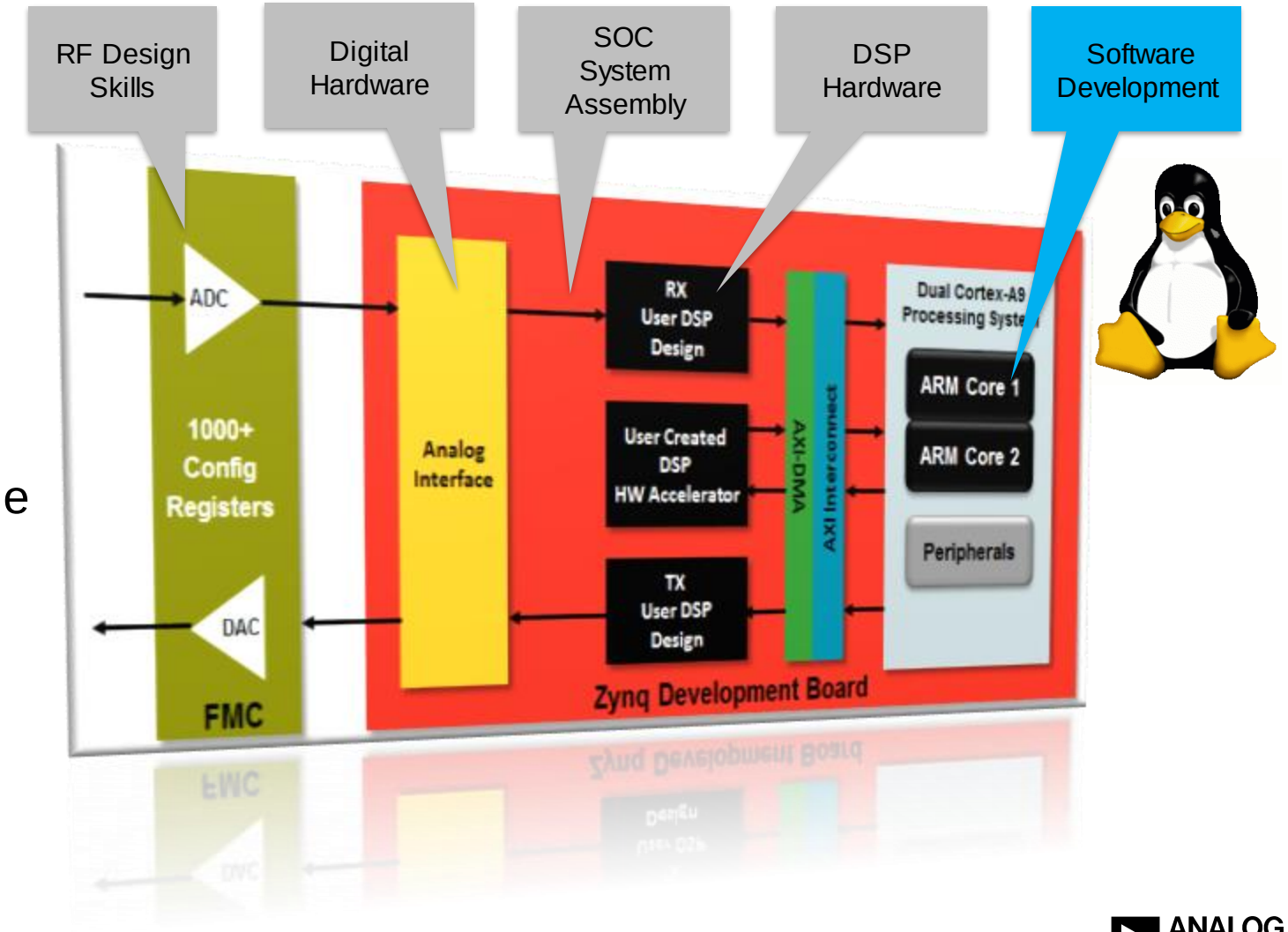
MICHAEL HENNERICH



GNU Radio Conference 2018
September 17 - 21, 2018 - Henderson Convention Center, Henderson, Nevada

Agenda

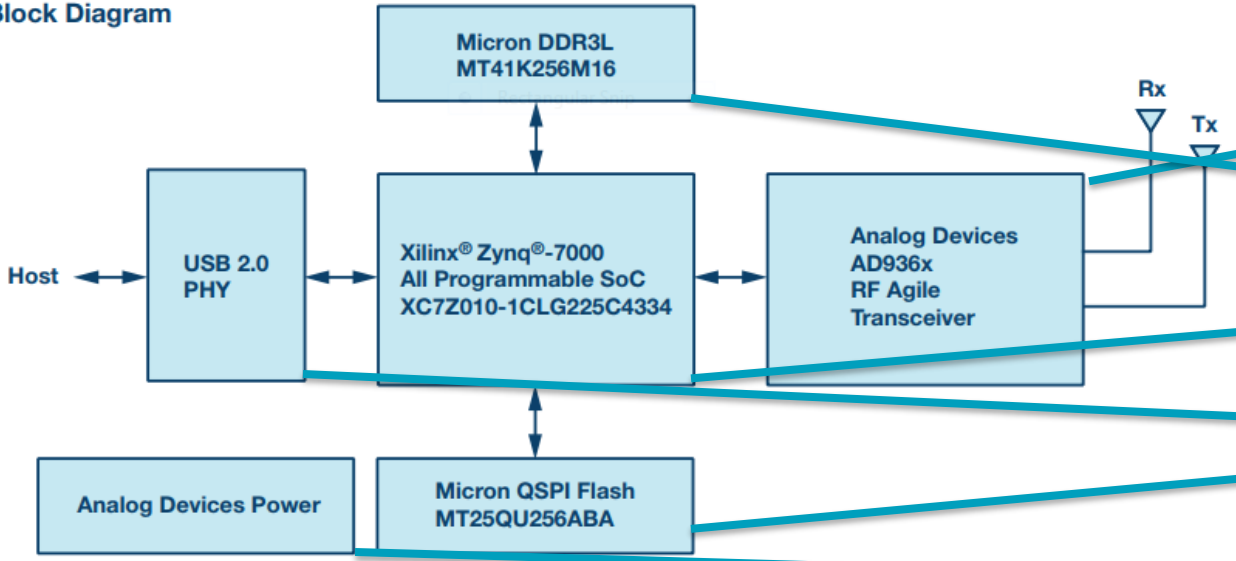
- PlutoSDR overview
  - System components
  - Connectivity options
- IIO introduction
  - Concept and Architecture
  - IIO for SDR
- Custom application libiio C example
- Building the PlutoSDR Firmware Image
- Customizing the PlutoSDR filesystem
- Cross-compiling external applications using sysroot
- GNU Radio *on* the PlutoSDR
- IIO on other COTS SDR transceivers

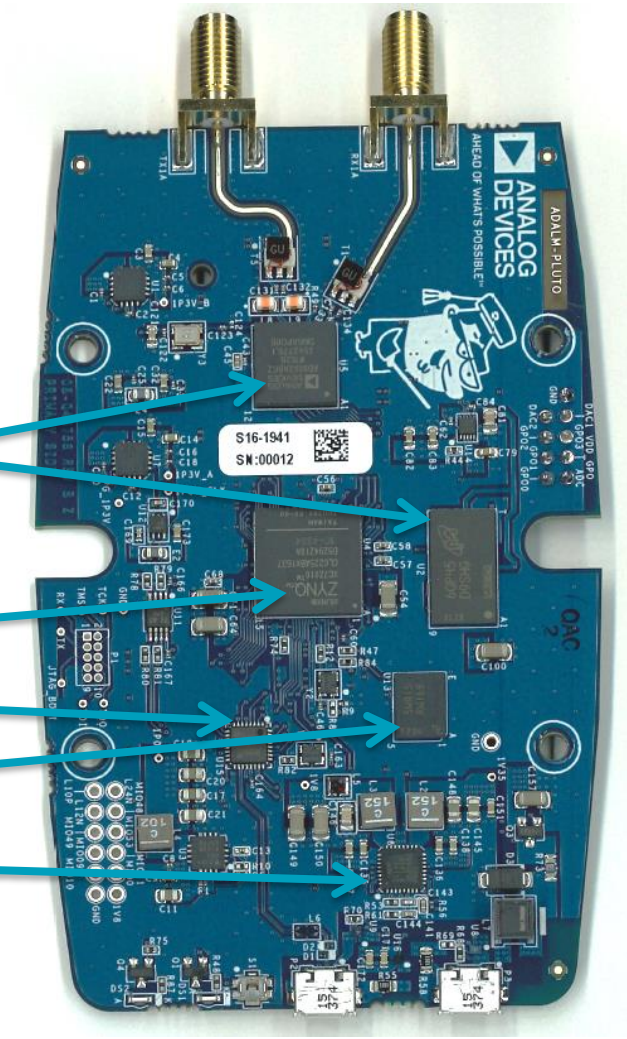


ADALM-PLUTO aka PlutoSDR – What's inside?

- AD9363
- Xilinx Zynq-7010
- 512 MB DDR3
- 32 MB SPI NOR
- USB 2.0 OTG

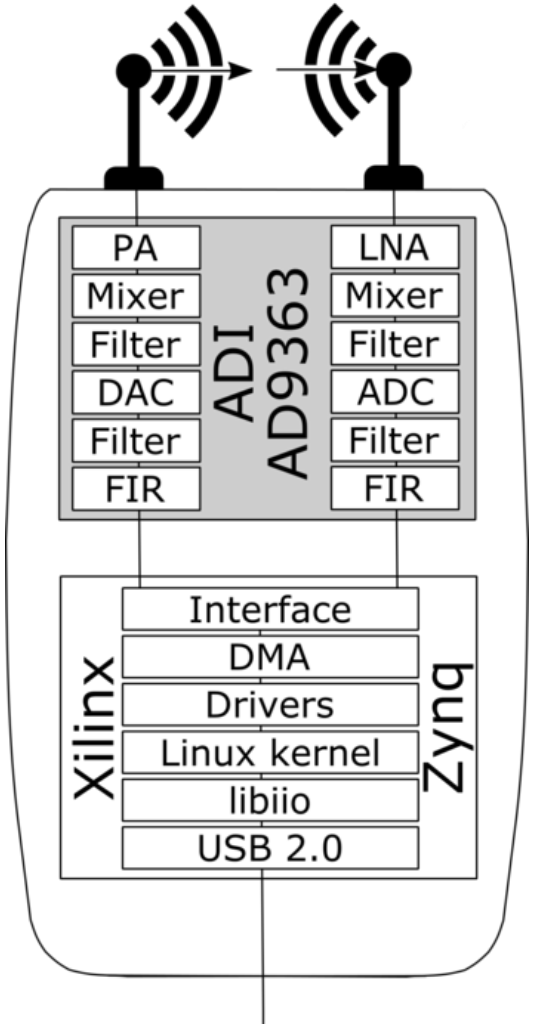
Simplified Block Diagram





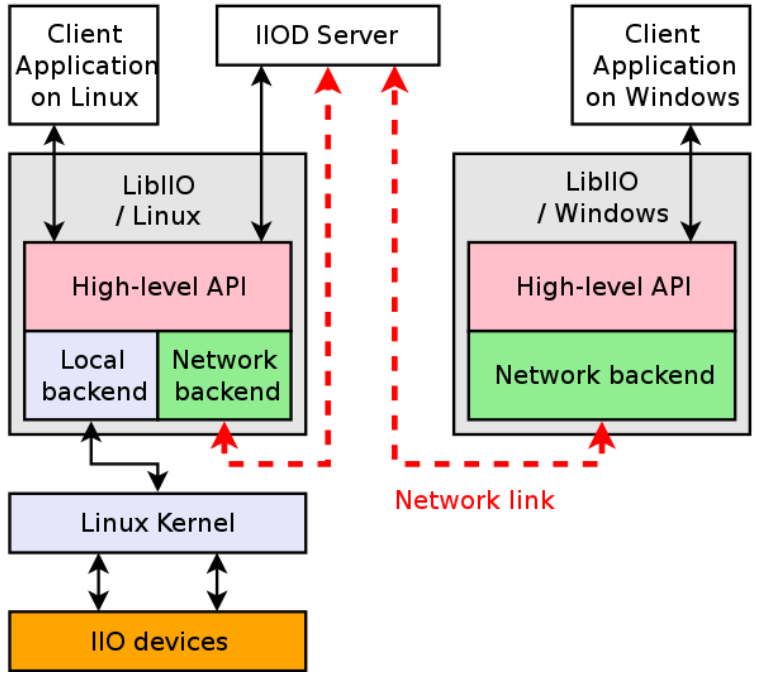
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

ADALM-PLUTO – Software stack



- Runs Linux inside the device
- Uses Linux's IIO framework to expose I/Q data and control
- Multi-Function USB Device
  - Native IIO over USB
  - Serial over USB
    - Kernel console
    - COMx, ttyACMx
  - Ethernet over USB (RNDIS)
  - Mass Storage
  - Device Firmware Update (DFU)
- USB Host
  - USB dongles

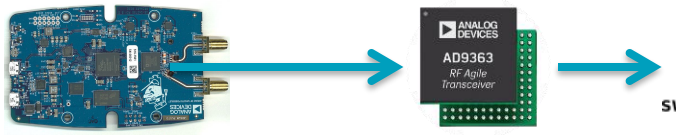
- Cross Platform
  - Windows
  - Linux
  - MAC
- Cross framework
  - Stacked libraries based on libiio



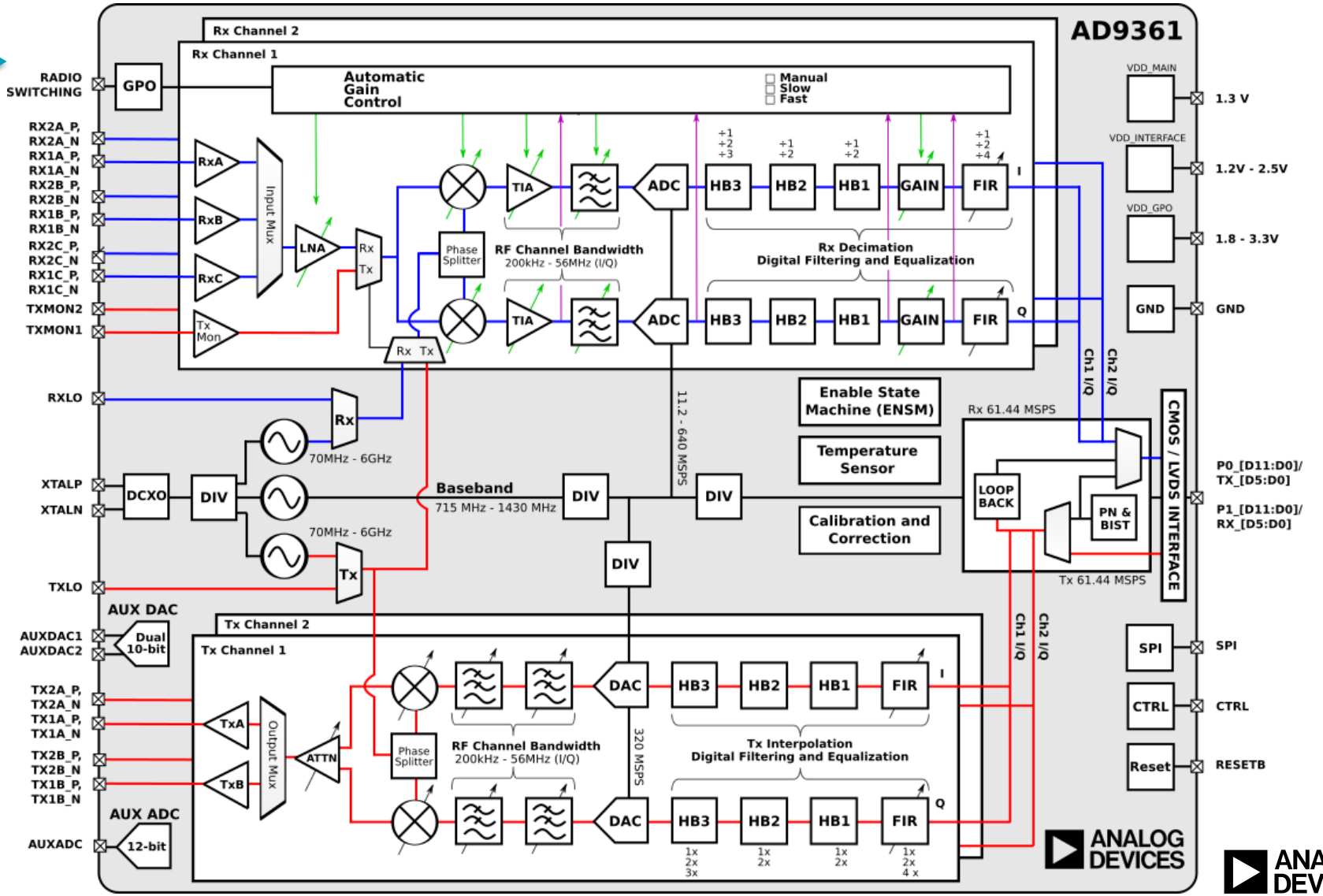
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

AD936x – Under the Hood

For more information:
http://www.analog.com/ad9361
http://www.analog.com/ad9364
http://www.analog.com/ad9363

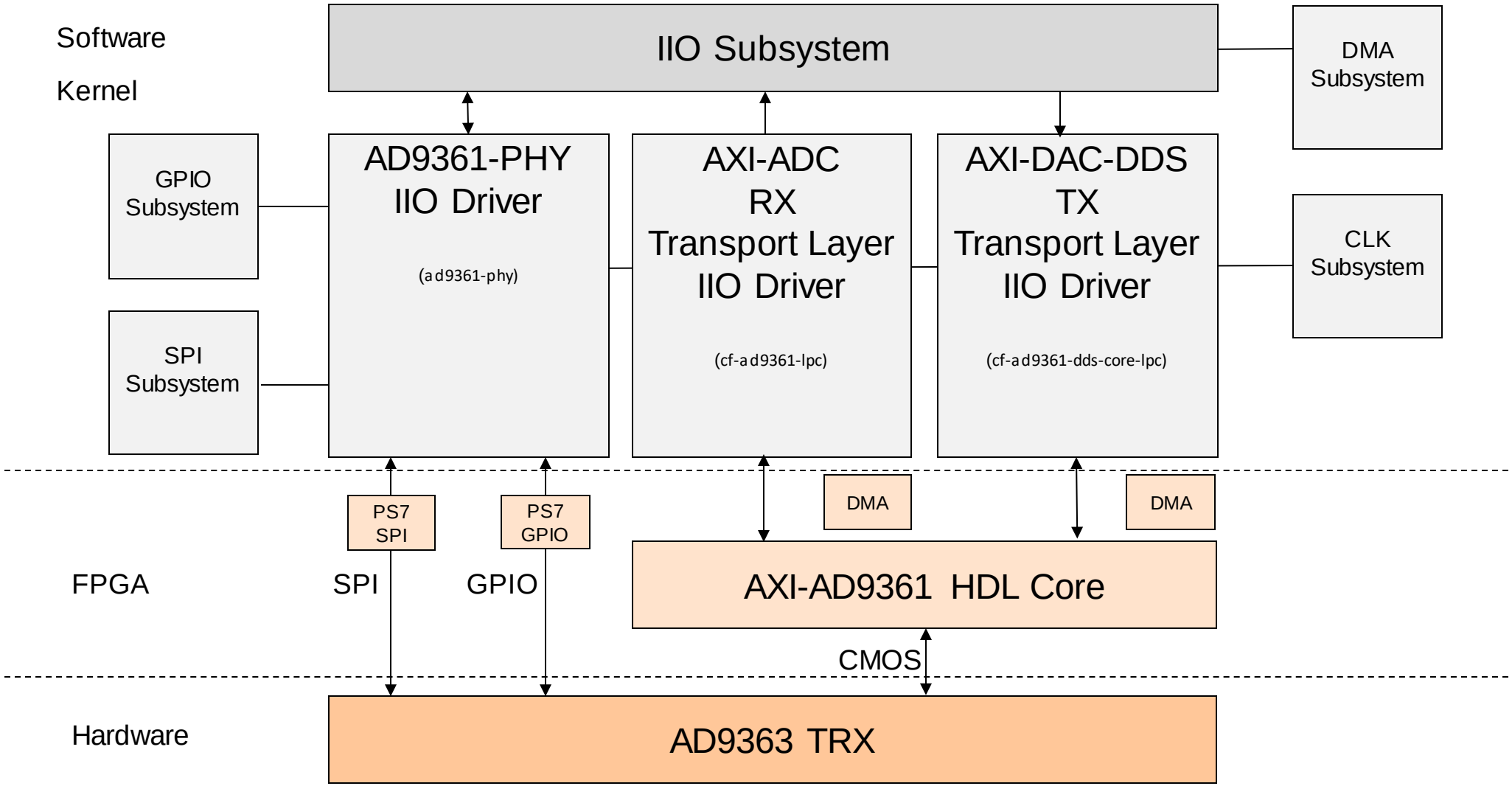


- AD9361: 2 Rx + 2 Tx
- AD9364: 1 Rx + 1 Tx
- AD9363: 2 Rx + 2 Tx
- Major sections:
  - RF input/output paths
  - RF PLL/LO
  - Clock generation
  - ADC/DAC
  - Digital filters
  - Digital interface
  - Enable state machine
  - RX Gain (AGC)
  - TX Attenuation
  - Aux DAC/ADC and GPOs
  - Analog and Digital Correction/Calibration

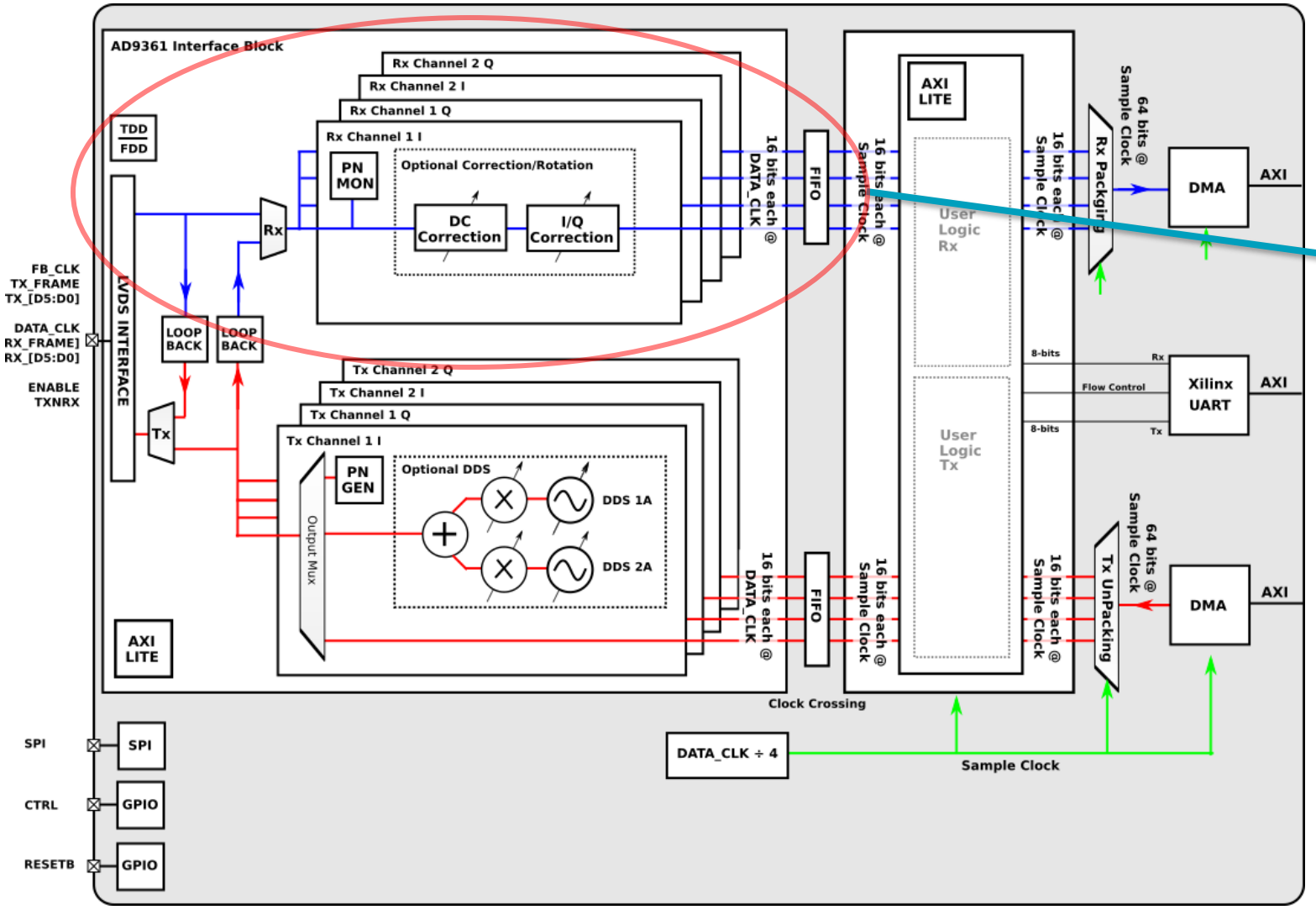


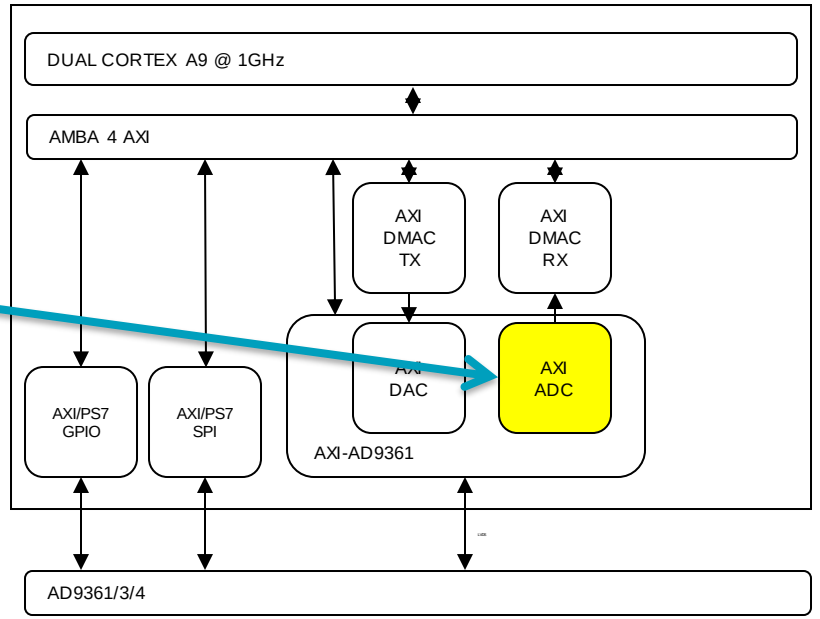
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Software, Programmable Logic & Hardware



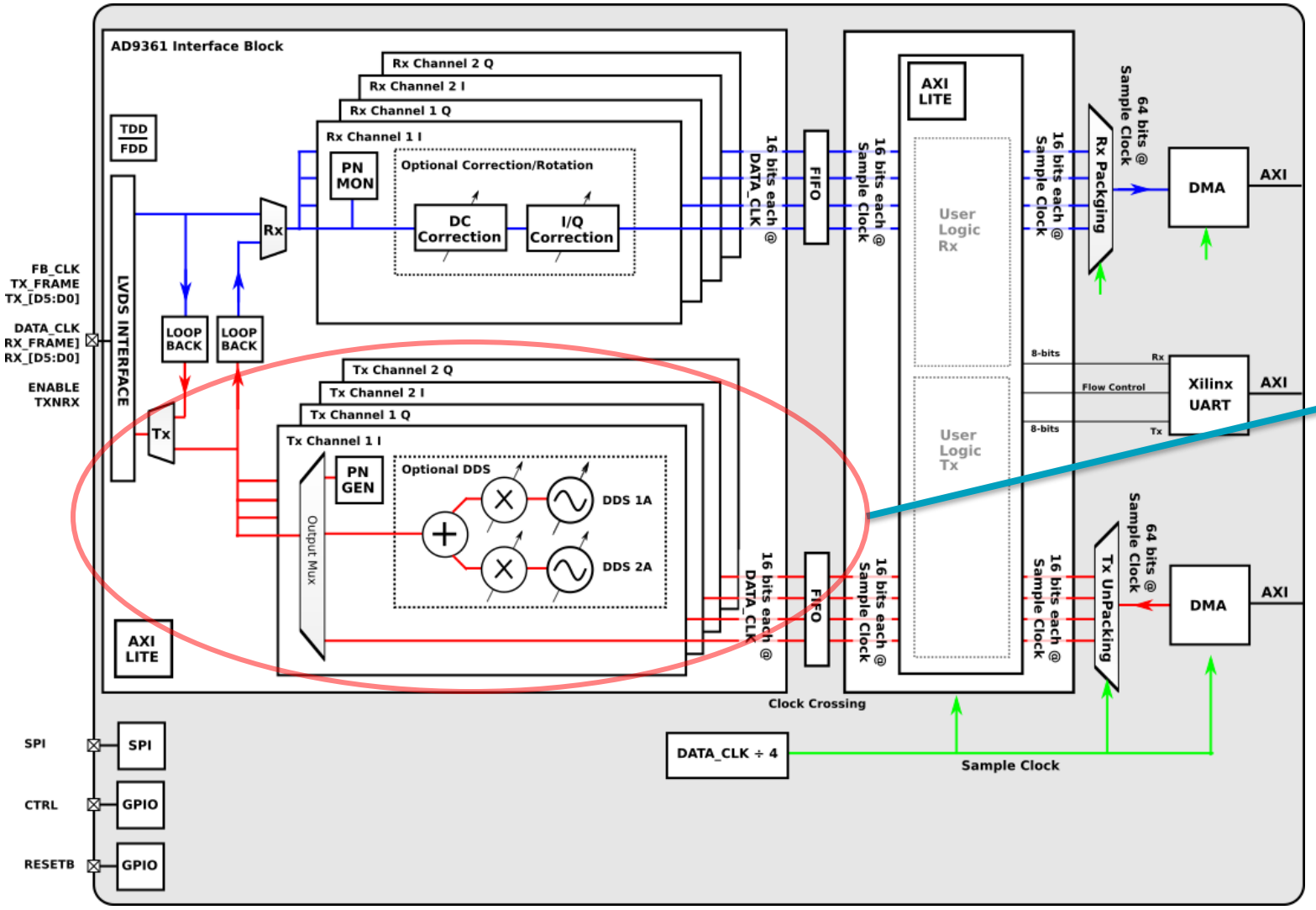
FPGA HDL Cores – RX

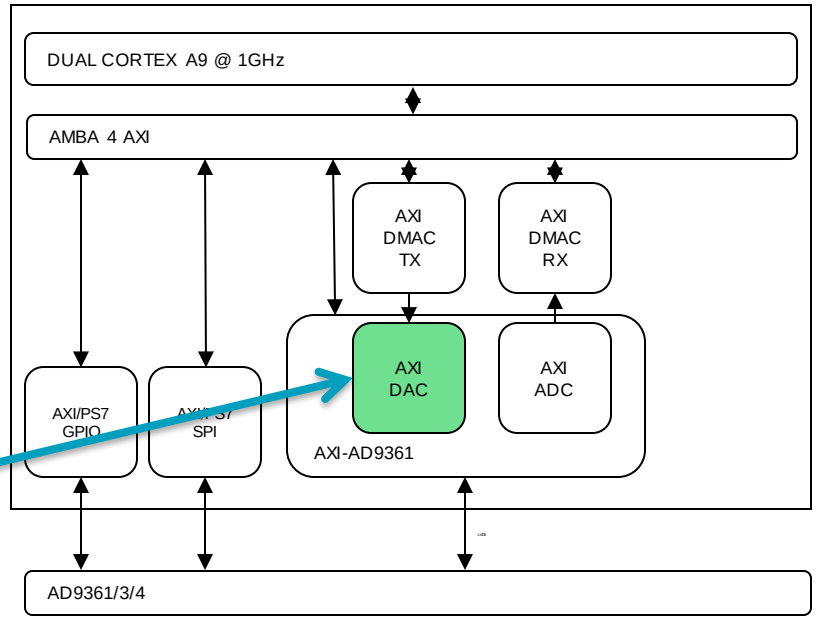




ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

FPGA HDL Cores – TX

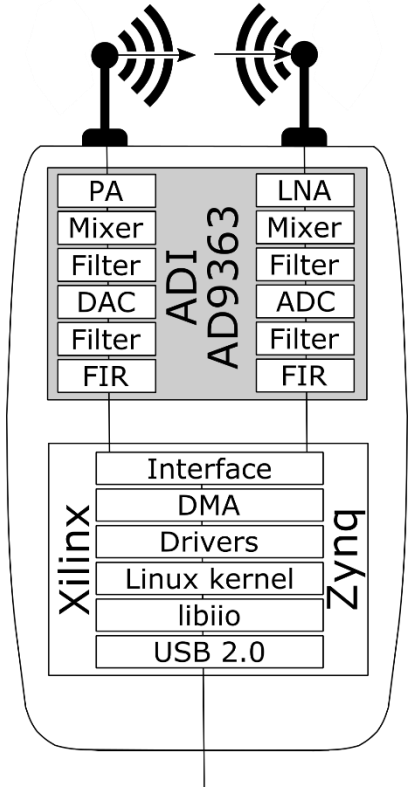




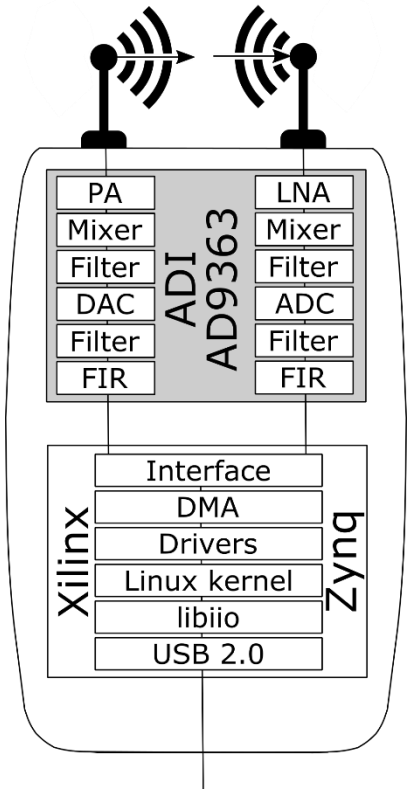
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

ADALM-PLUTO USB OTG Connectivity Options

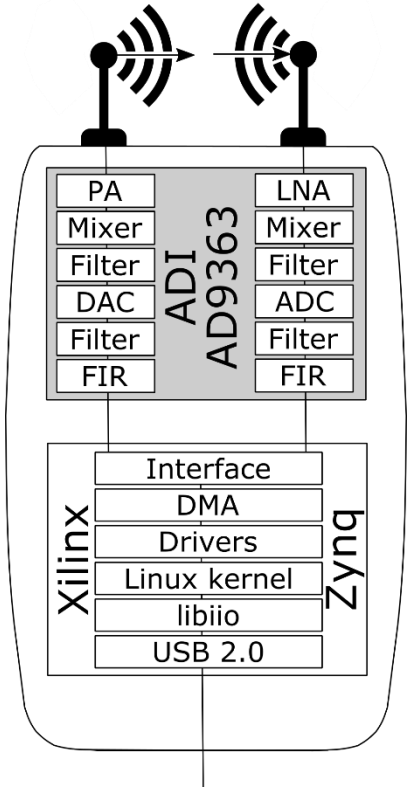
Connect to host



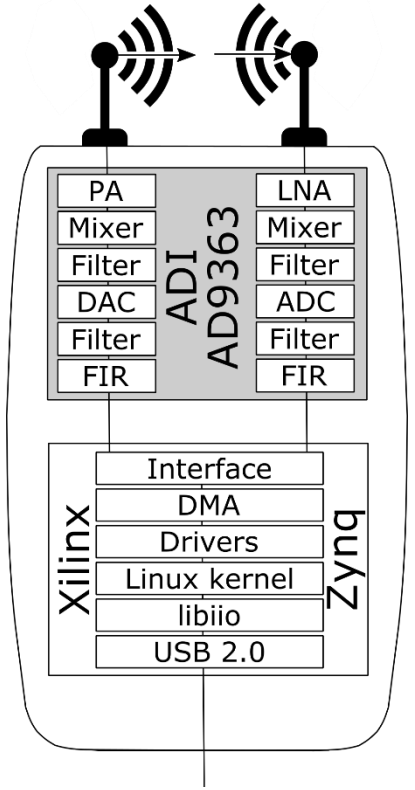
USB Thumb Drive



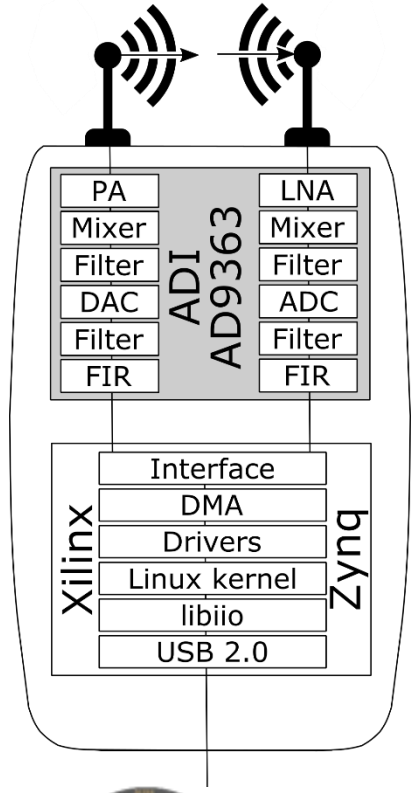
USB LAN



USB WiFi



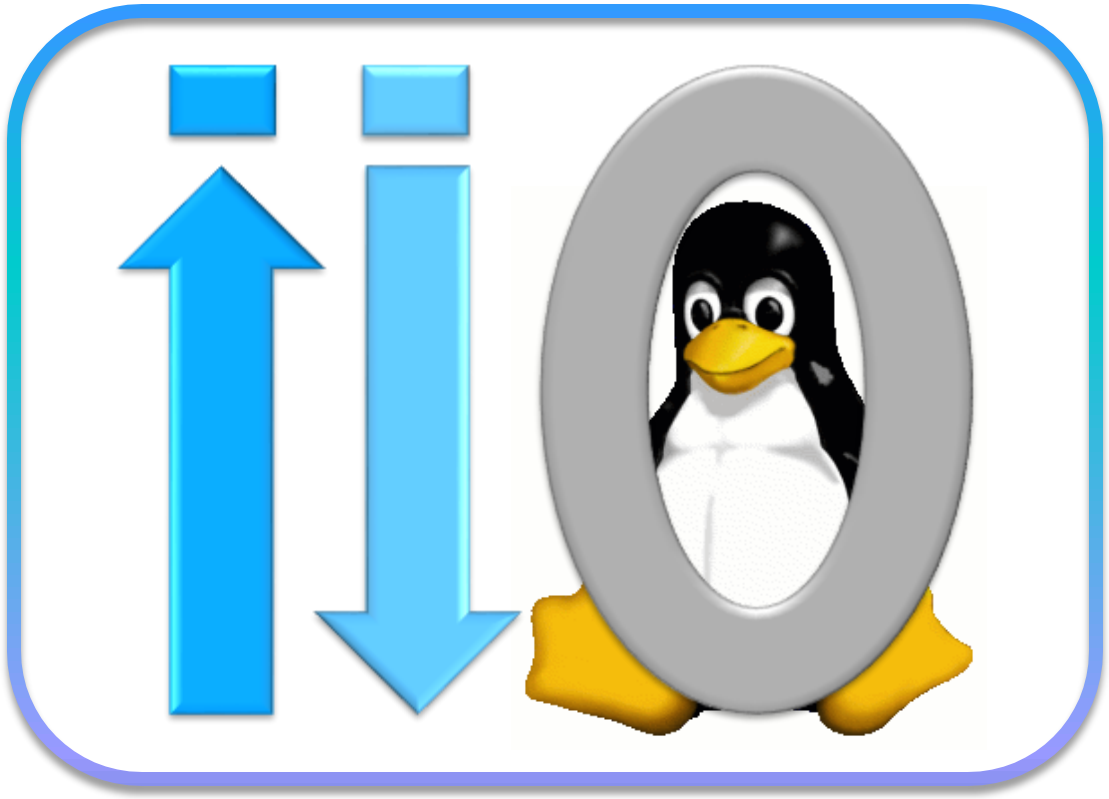
USB Audio



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

What is IIO?

- Linux kernel **I**ndustrial **I**nput / **O**utput framework
  - Not really just for Industrial IO
  - All non-HID IO
  - ADC, DAC, light, accelerometer, gyro, magnetometer, humidity, temperature, pressure, rotation, angular momentum, chemical, health, proximity, counters, etc.
- In the upstream Linux kernel for 10 years.
- Mailing list:
  - linux-iio@vger.kernel.org



https://www.kernel.org/doc/html/latest/driver-api/iio/index.html

Why use IIO for SDR?

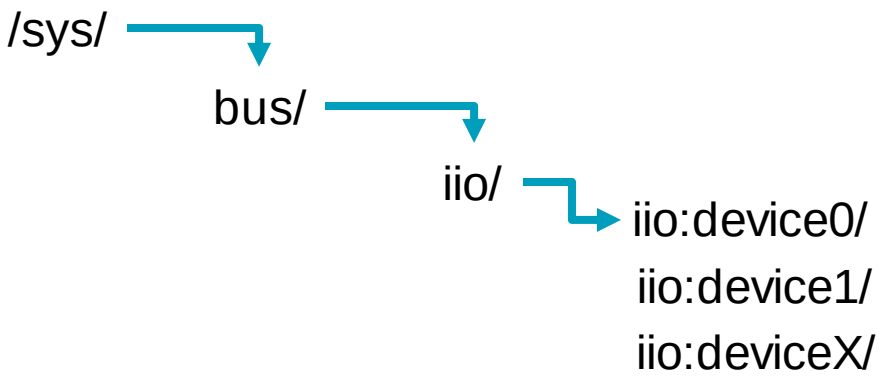
- Provides hardware abstraction layer
  - Allows sharing of infrastructure
  - Allows developers to focus on the solution
  - Allows application re-use

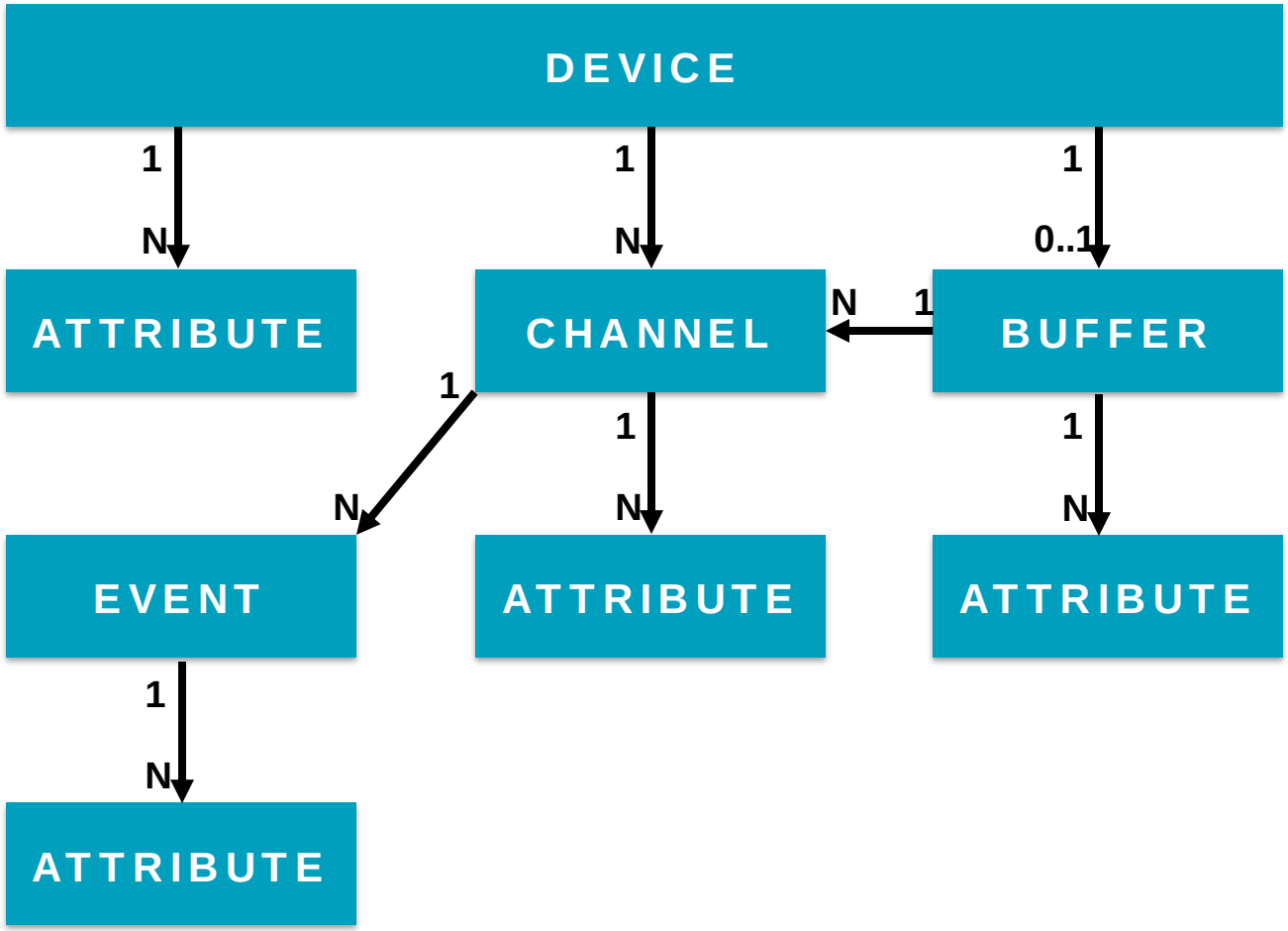
- Kernel drivers have low-level & low-latency access to hardware
  - MMIO
  - Interrupts
  - DMA
  - Memory

- IIO provides fast and efficient data transport
  - From device to application
  - From application to device
  - From device to network/storage

IIO – Devices

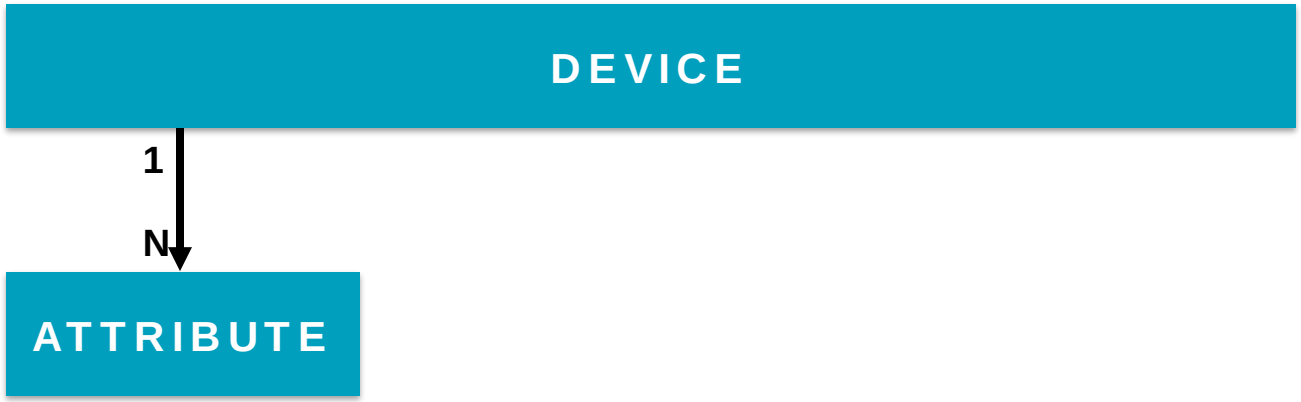
- Main structure
- Typically corresponds to a single physical hardware device
- Represented as directories in sysfs





IIO – Attributes

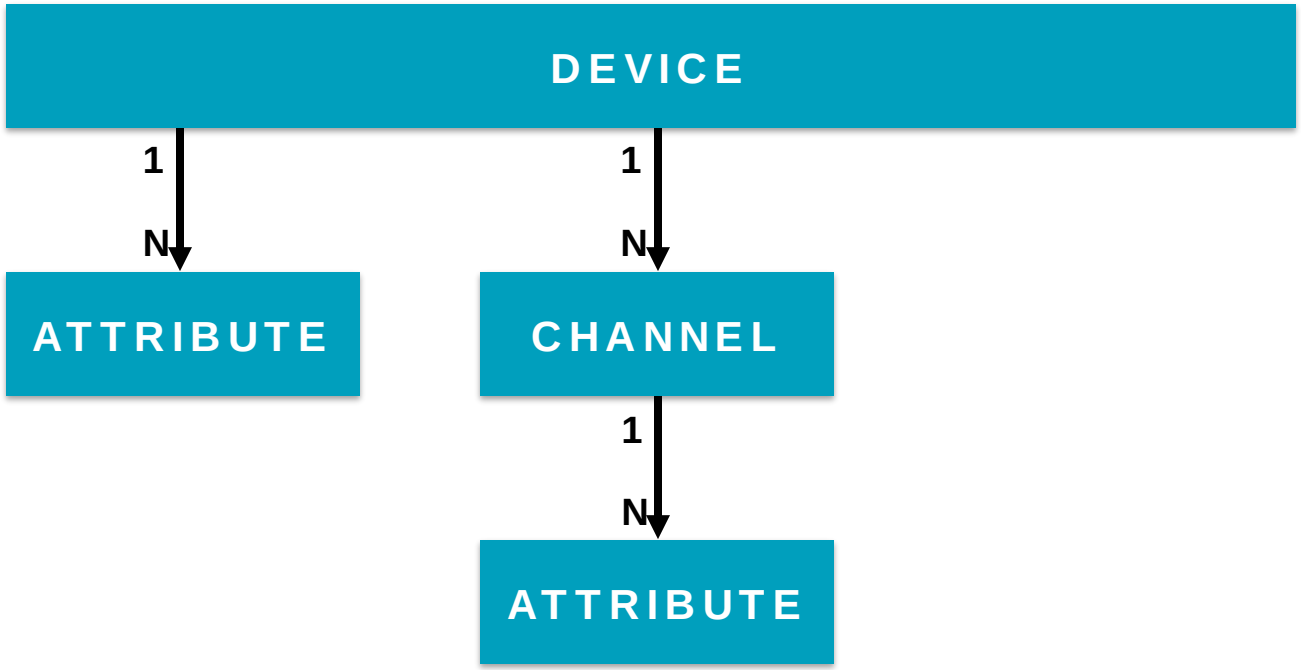
- Describe hardware capabilities
- Allow to configure hardware features
  - SAMPLING_FREQUENCY
  - POWERDOWN
  - PLL_LOCKED
  - SYNC_DIVIDERS
  - etc.
- Represented as files in sysfs




```
# ls /sys/bus/iio/devices/
iio:device0  iio:device1  iio:device2  iio:device3  iio:device4
# cat  /sys/bus/iio/devices/*/name
adm1177
ad9361-phy
xadc
cf-ad9361-dds-core-lpc
cf-ad9361-lpc
#
```

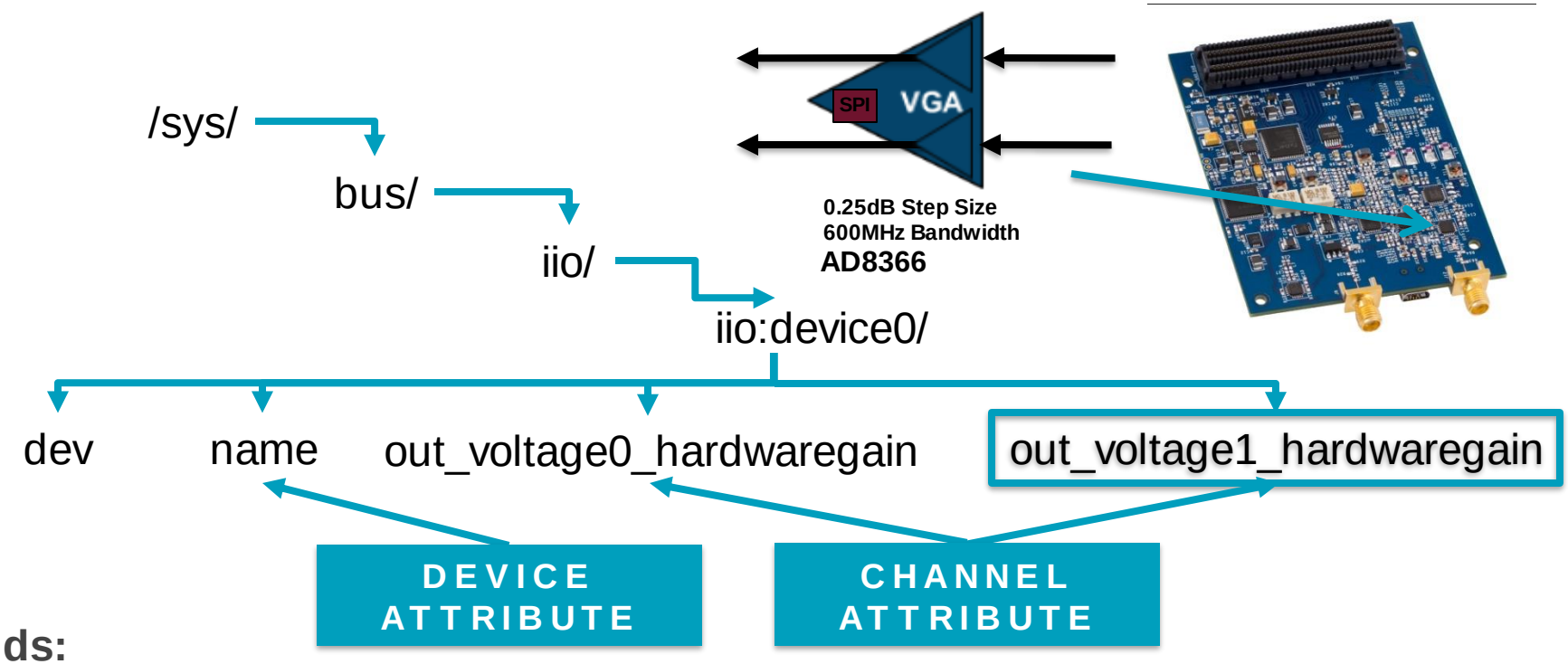
IIO – Channels

- Representation of a data channel
- Has direction, type, index and modifier
  - Direction
    - IN
    - OUT
  - Type
    - IIO_VOLTAGE
    - IIO_TEMP, etc.
  - Index
    - 0..N
  - Modifier
    - IIO_MOD_I, IIO_MOD_Q
- Channel Attributes provide additional information
  - RAW
  - SCALE
  - OFFSET
  - FREQUENCY
  - PHASE
  - HARDWAREGAIN
  - etc.



- Example: Read voltage from ADC Channel X in mV
  - *VoltageX_mV = (in_voltageX_raw + in_voltageX_offset) * in_voltageX_scale*

Example Device: AD8366 VGA/PGA Gain Control

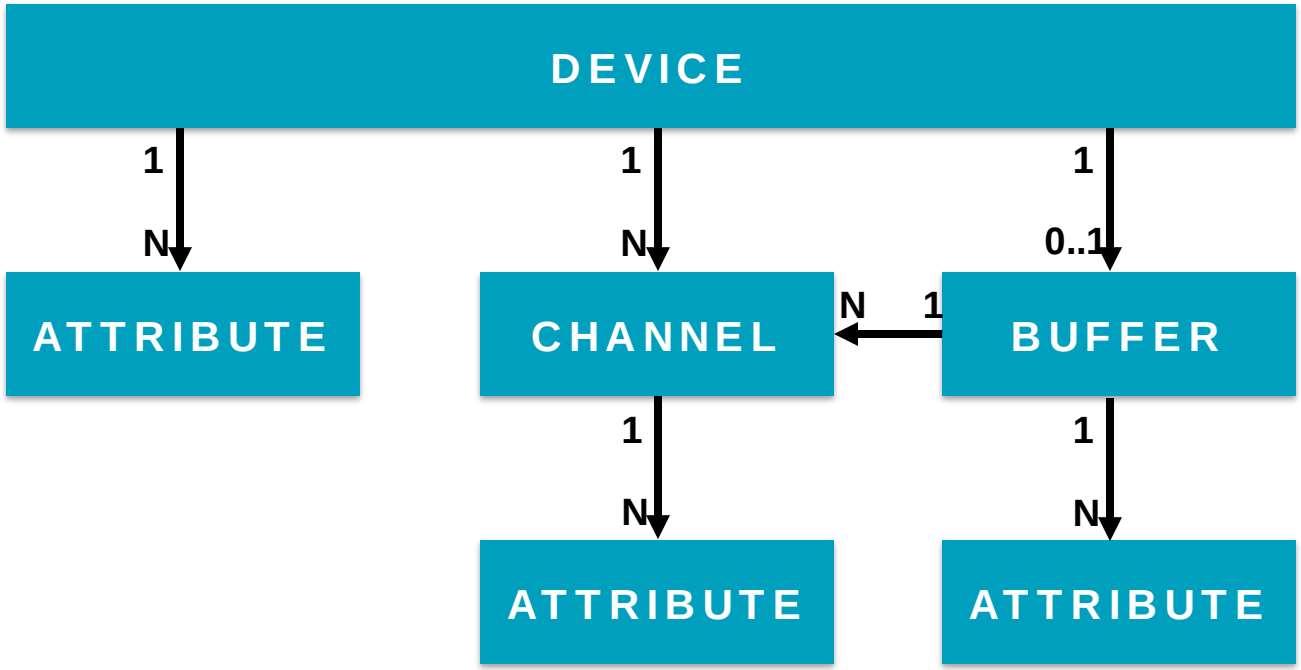


Shell Commands:

```
/sys/bus/iio/iio:device0 # cat name
ad8366-lpc
/sys/bus/iio/iio:device0 # echo 6 > out_voltage1_hardwaregain
/sys/bus/iio/iio:device0 # cat out_voltage1_hardwaregain
5.765000 dB
```

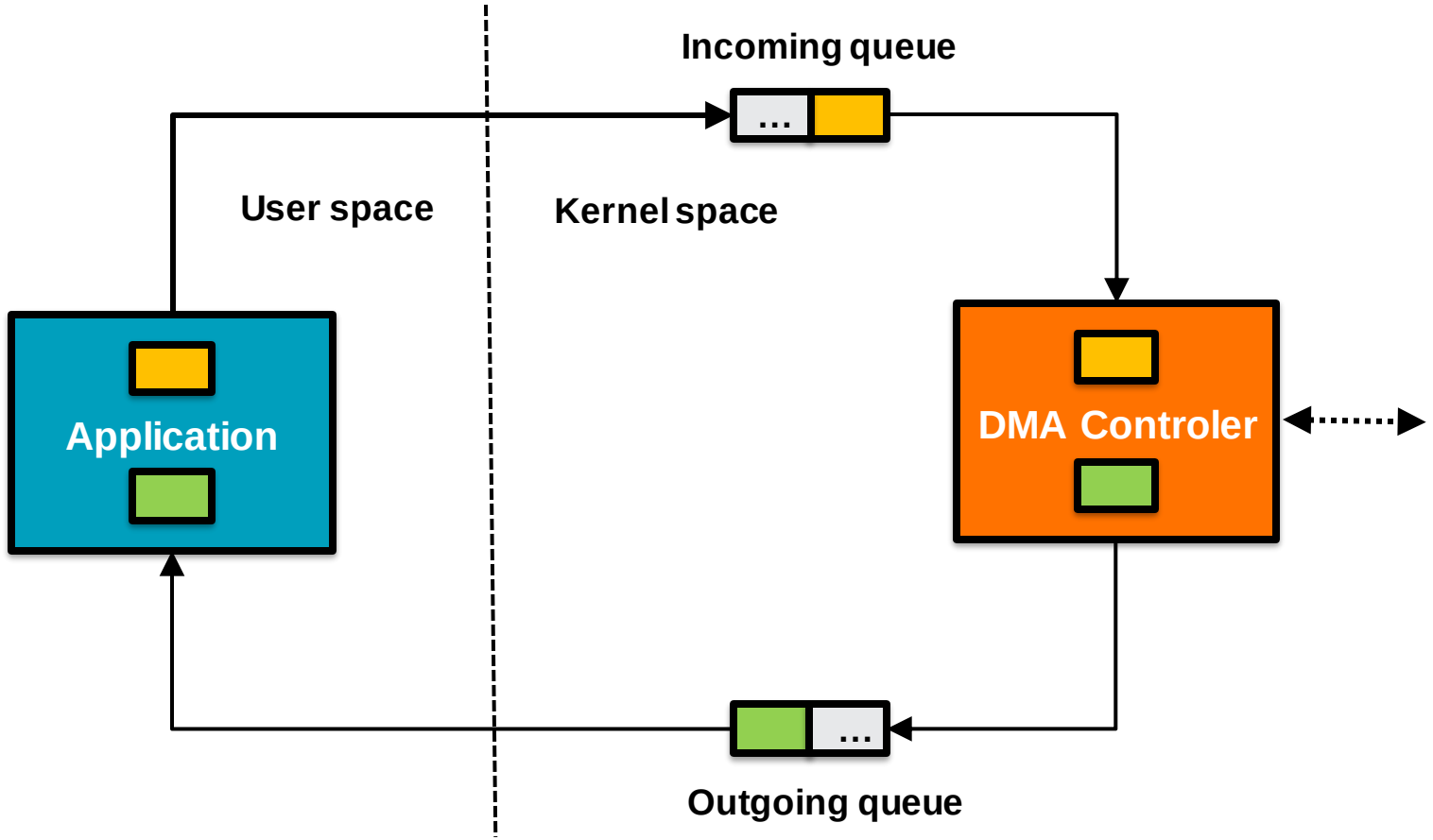
IIO – Buffers

- Used for continuous data capture/transmit
- Channels can be enabled/disabled
- Channels specify their data layout
  - [be|le]:[s|u]bits/storagebitsXrepeat[>>shift]
- /dev/iio:deviceX allows read()/write() access
- Configuration using sysfs files
- Support for different buffer implementations
  - Software FIFO
  - DMA Buffer
  - Device specific buffer



IIO – DMA buffer

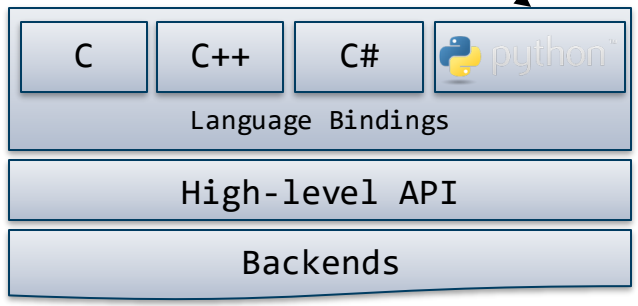
- DMA is used to copy data from device to memory
- mmap() is used to make data available in the application
- Allows low overhead high-speed data capture
- Data is grouped into chunks (called DMA blocks) to manage ownership
  - Either application or driver/hardware owns a block
  - Samples per block are configurable
  - Number of blocks are configurable



IIO – libiio

- System library
- Abstracts away low level details of the IIO kernel ABI
  - Kernel ABI is designed to be simple and efficient
  - libiio focuses on ease of use
- Provides high-level C, C++, C# or Python programming interface to IIO (Language bindings)
  - Write your IIO application in your favorite language
- Cross Platform (Linux, Windows, MacOS X, BSD)
- Available as
  - Official DEBIAN package
  - RPM package
  - OpenEmbedded Layer meta-oe/libiio
  - Buildtroot package
  - Windows or Mac OS X installer
  - Etc.

```
#!/usr/bin/env python

import iio

ctx = iio.Context()

for dev in ctx.devices:
   print dev.name
```



For more information:

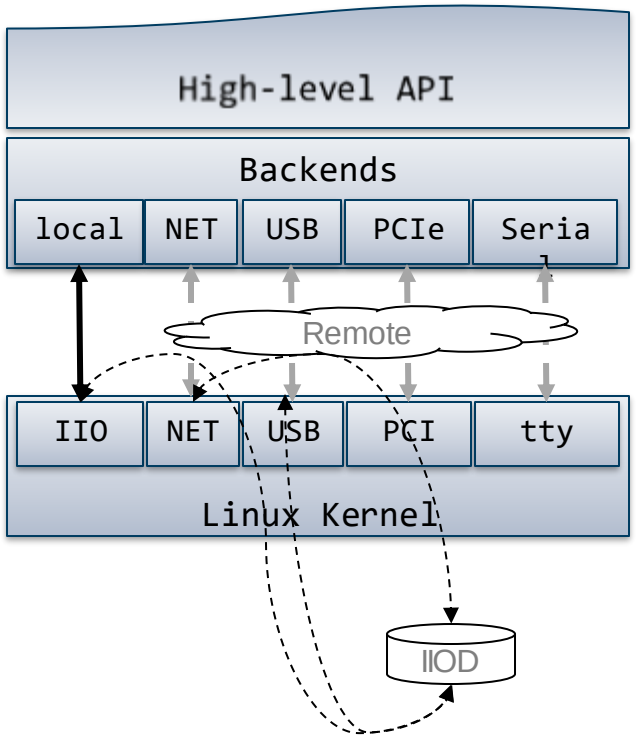
https://github.com/analogdevicesinc/libiio

http://wiki.analog.com/resources/tools-software/linux-software/libiio_internals

http://analogdevicesinc.github.io/libiio/

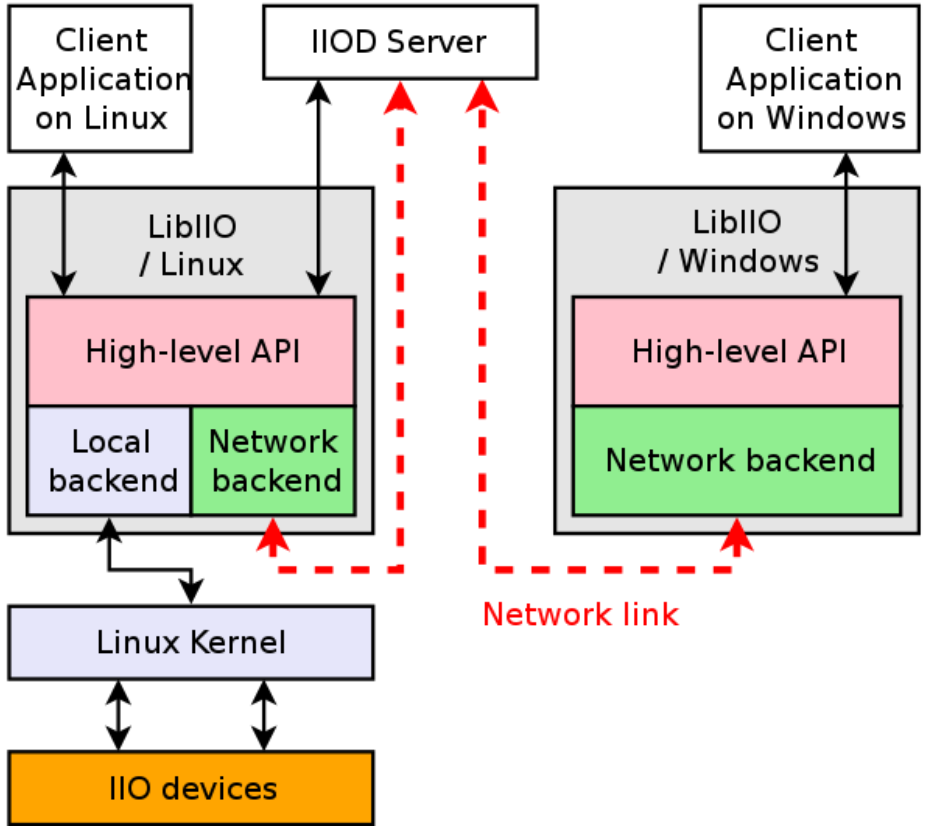
IIO – libiio – Backends

- Support for backends
  - Backend takes care of low-level communication details
  - Provide the same API for applications
  - Transparent from the applications point of view
- Multiple backends
  - Local, directly uses the Linux kernel IIO ABI
  - Network, uses network protocol to talk to (remote) iiod server which uses it's local backend
  - USB, SERIAL
- Allows to create flexible and portable applications
  - Write once, deploy everywhere
  - E.g. develop application on PC, deploy on embedded system (SoC, FPGA)



IIO – iiod

- Allows multiplexing between multiple readers/writers
- Provides support for remote clients via:
  - TCP/IP
  - USB
  - Serial
- Applications do not need system level privileges
- Transparent from the applications point of view



IIO – libiio – Command line tools

- **iio_info** : Information about all IIO devices, backends and context attributes
  - `iio_info –u ip:192.168.2.1`
- **iio_attr** : Read and write IIO attributes
  - `iio_attr -c ad9361-phy altvoltage0 frequency 2450000000`
- **iio_readdev** : Read samples from an IIO device
  - `iio_readdev -u usb:1.100.5 -b 100000 cf-ad9361-lpc | pv > /dev/null`
- **iio_writedev** : Write samples to an IIO device
  - `iio_readdev -b 100000 cf-ad9361-lpc | iio_writedev -b 100000 cf-ad9361-dds-core-lpc`
- **iio_reg** : Read or write SPI or I2C registers in an IIO device (useful to debug drivers)
  - `iio_reg adrv9009-phy 0`

ANALOG DEVICES

AHEAD OF WHAT'S POSSIBLE™

Custom Applications

IIO – libiio – example

- Controlling the transceiver
  - The code snippet to the right is a minimalistic example without error checking. It shows how to control the AD936x transceiver via a remote connection.

1. Create IIO IP Network context.
   1. Instead of ip:xxx.xxx.xxx.xxx it'll also accept
      1. local:
      2. usb:XX.XX.X
      3. serial:/dev/ttyAMA0,115200n8
2. Get the AD936x PHY device structure
3. Set the TX LO frequency
4. Set RX baseband rate

```c
#include <iio.h>

int main (int argc, char **argv)
{
        struct iio_context *ctx;
        struct iio_device *phy;

        ctx = iio_create_context_from_uri("ip:192.168.2.1");

        phy = iio_context_find_device(ctx, "ad9361-phy");

        iio_channel_attr_write_longlong(
                iio_device_find_channel(phy, "altvoltage0", true),
                "frequency",
                2400000000); /* RX LO frequency 2.4GHz */

        iio_channel_attr_write_longlong(
                iio_device_find_channel(phy, "voltage0", false),
                "sampling_frequency",
                5000000); /* RX baseband rate 5 MSPS */

        receive(ctx);

        iio_context_destroy(ctx);

        return 0;
}
```

Driver Documentation: https://wiki.analog.com/resources/tools-software/linux-drivers/iio-transceiver/ad9361

IIO – libiio – receive example

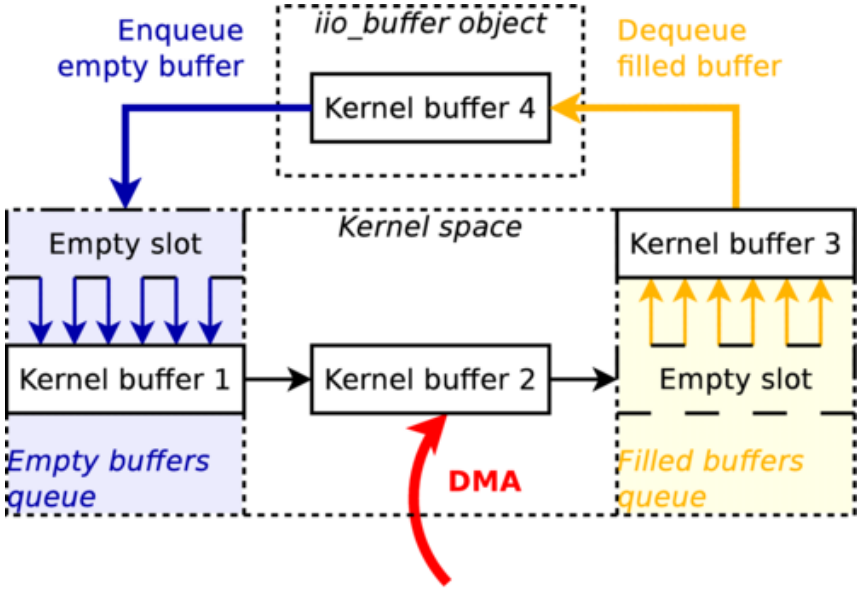
- Receiving data
  1. Get the RX capture device structure
  2. Get the IQ input channels
  3. Enable I and Q channel
  4. Create the RX buffer
  5. Fill the buffer
  6. Get the layout of the buffer
  7. Process samples

```c
int receive(struct iio_context *ctx)
{
        struct iio_device *dev;
        struct iio_channel *rx0_i, *rx0_q;
        struct iio_buffer *rxbuf;

        dev = iio_context_find_device(ctx, "cf-ad9361-lpc");

        rx0_i = iio_device_find_channel(dev, "voltage0", 0);
        rx0_q = iio_device_find_channel(dev, "voltage1", 0);

        iio_channel_enable(rx0_i);
        iio_channel_enable(rx0_q);

        rxbuf = iio_device_create_buffer(dev, 4096, false);
        if (!rxbuf) {
                perror("Could not create RX buffer");
                shutdown();
        }

        while (true) {
                void *p_dat, *p_end, *t_dat;
                ptrdiff_t p_inc;

                iio_buffer_refill(rxbuf);

                p_inc = iio_buffer_step(rxbuf);
                p_end = iio_buffer_end(rxbuf);

                for (p_dat = iio_buffer_first(rxbuf, rx0_i); p_dat < p_end; p_dat += p_inc, t_dat += p_inc) {
                        const int16_t i = ((int16_t*)p_dat)[0]; // Real (I)
                        const int16_t q = ((int16_t*)p_dat)[1]; // Imag (Q)

                        /* Process here */

                }
        }

        iio_buffer_destroy(rxbuf);

}
```

IIO System considerations

- Buffer handling, sizes and counts
  - Typically set to a frame or chunk size suitable for signal processing (e.g. N x FFT_size)
  - Small buffers -> less latency but more overhead
  - Large buffers -> less overhead but more latency
  - Number of discrete buffers are configurable, default is 4.
    - iio_device_set_kernel_buffers_count()
  - Capturing starts as soon as the buffer is created *iio_device_create_buffer()*
  - FIFO like behavior – new data is dropped

- IIO buffer DMA max block size
  - Max buffer size is limited by the **max_block_size** parameter
  - Default 16M
  - Can be adjusted
    - sysfs: /sys/module/industrialio_buffer_dma/parameters/max_block_size
    - Kernel command line: *industrialio_buffer_dma.max_block_size=size_in_bytes*



IIO - System considerations

- Linux Contiguous Memory Allocator (or CMA)
  - Allocation of big, physically-contiguous memory blocks
  - Reserve memory early at boot time
  - Kconfig menu “Device Drivers” -> “Generic Driver Options”-> “Contiguous Memory Allocator”
  - Kernel command line option *cma=size_in_bytes*
  - PlutoSDR default 256M

- IIO context timeout
  - May be triggered by low sample rates and large buffers
  - iio_context_set_timeout() timeout parameter set to 0 disables the timeout

Building the PlutoSDR Firmware Image

- Download and install Xilinx FPGA Tools
  - Vivado HLx 2017.2: WebPACK and Editions - Linux Self Extracting Web Installer
    - During installation check under design tools Software Development Kit (SDK)
    - Under devices SoC make sure Zynq-7000 is selected
    - Xilinx gcc tools are distributed as 32-bit binaries you may need to add 32-bit libs

```
michael@HAL9000:~/devel$ dpkg –add-architecture i386
michael@HAL9000:~/devel$ apt-get update
michael@HAL9000:~/devel$ sudo apt-get install libc6:i386 libstdc++6:i386
```

- Install other build dependencies

```
michael@HAL9000:~/devel$ sudo apt-get install git build-essential fakeroot libncurses5-dev libssl-dev ccache
michael@HAL9000:~/devel$ sudo apt-get install dfu-util u-boot-tools device-tree-compiler libssl1.0-dev mtools
```

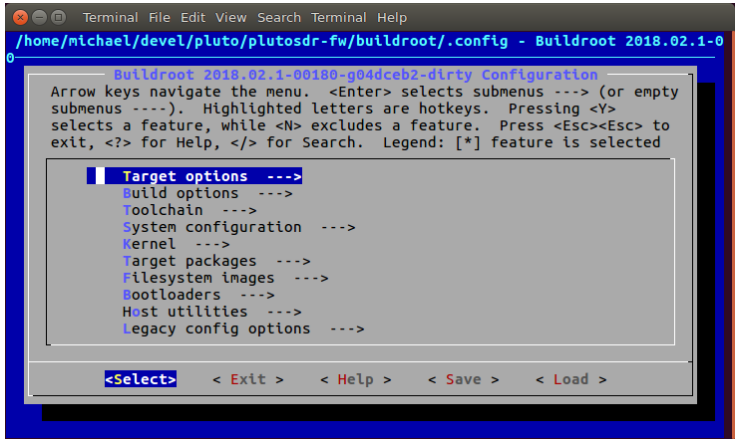
- Clone and build the Firmware image

```
michael@HAL9000:~/devel$ git clone --recursive https://github.com/analogdevicesinc/plutosdr-fw.git
michael@HAL9000:~/devel$ cd plutosdr-fw
michael@HAL9000:~/devel/plutosdr-fw$ export CROSS_COMPILE=arm-xilinx-linux-gnueabi-
michael@HAL9000:~/devel/plutosdr-fw$ export PATH=$PATH:/opt/Xilinx/SDK/2017.2/gnu/arm/lin/bin
michael@HAL9000:~/devel/plutosdr-fw$ export VIVADO_SETTINGS=/opt/Xilinx/Vivado/2017.2/settings64.sh
michael@HAL9000:~/devel/plutosdr-fw$ make
```

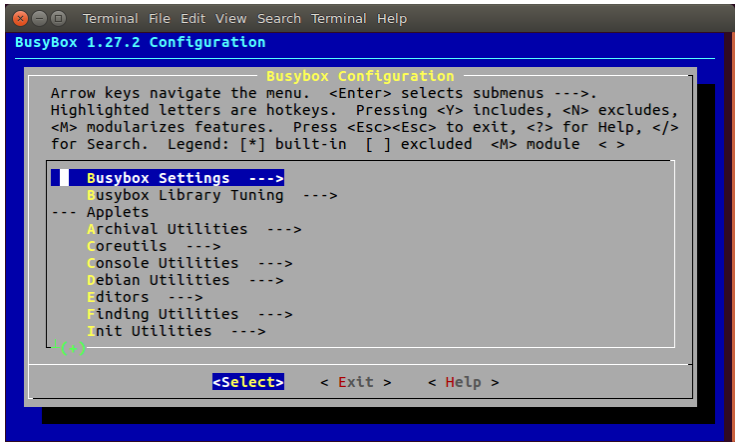
Customizing the PlutoSDR filesystem

- Customize buildroot target packages

```
michael@HAL9000:~/devel/plutosdr-fw/buildroot$ make menuconfig
michael@HAL9000:~/devel/plutosdr-fw/buildroot$ make savedefconfig
michael@HAL9000:~/devel/plutosdr-fw$ make
```



- Customize buildroot busybox tools




```
michael@HAL9000:~/devel/plutosdr-fw/buildroot$ make busybox-menuconfig
michael@HAL9000:~/devel/plutosdr-fw/buildroot$ cp output/build/busybox-*/.config board/pluto/busybox-*.config
michael@HAL9000:~/devel/plutosdr-fw$ make
```

Customizing the PlutoSDR filesystem
Adding files

- For temporary modifications
  - Modify the target filesystem directly and then rebuild the image

```
michael@HAL9000:~/devel/plutosdr-fw$ cp ~/foobar.sh buildroot/output/target/sbin/
michael@HAL9000:~/devel/plutosdr-fw$ make
```

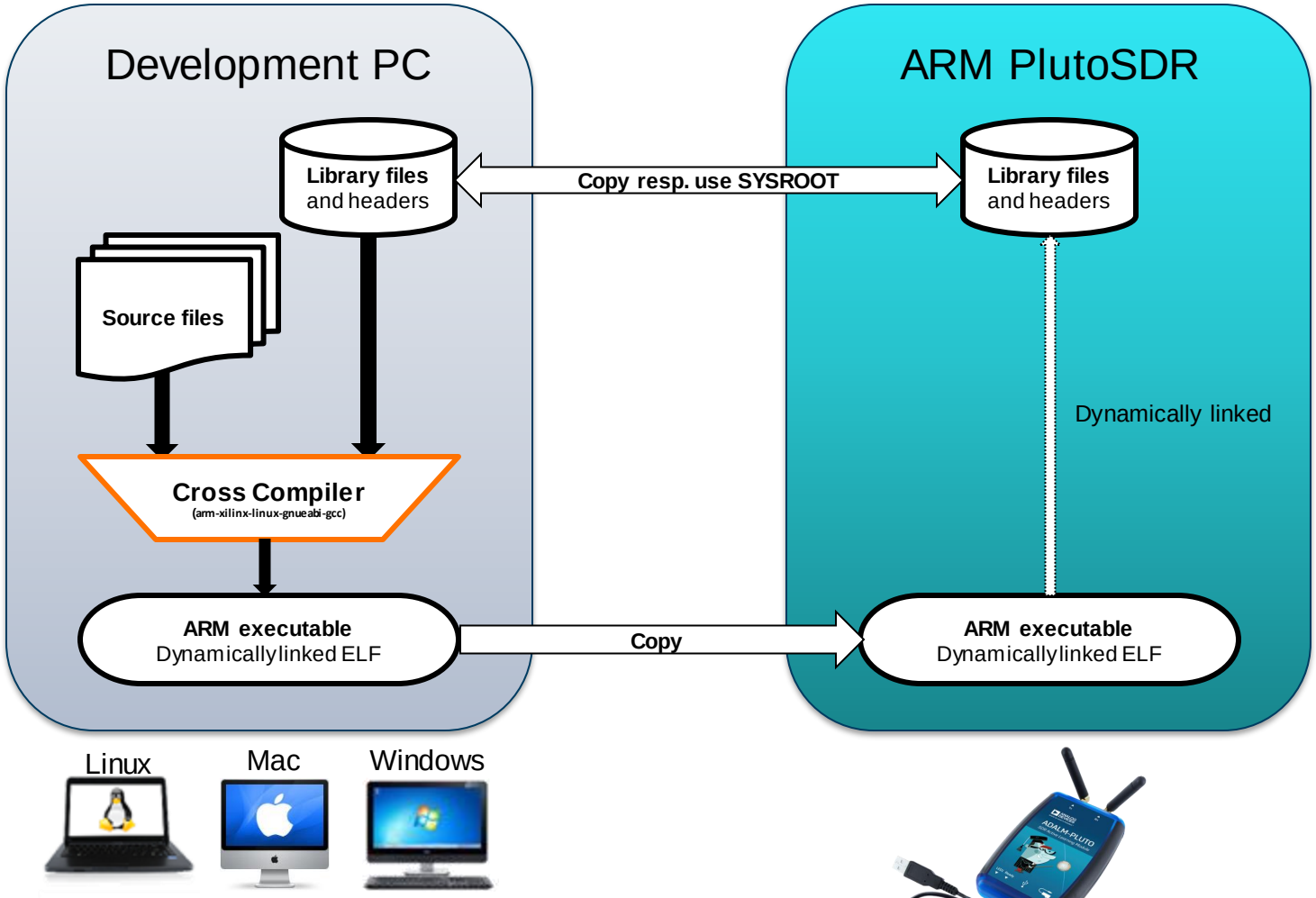
- For permanent additions
  - Post-build scripts
    - Are shell scripts called after Buildroot builds all the selected software, but before the rootfs images are assembled.

```
michael@HAL9000:~/devel/plutosdr-fw$ cat buildroot/board/pluto/post-build.sh
[ – snip –]
${INSTALL} -D -m 0644 ${BOARD_DIR}/input-event-daemon.conf ${TARGET_DIR}/etc/
[– snip --]
```

  - Filesystem overlays
    - A tree of files that is copied directly over the target filesystem after it has been built.
    - https://buildroot.org/downloads/manual/manual.html

Cross-compiling external applications using SYSROOT

- Along with each PlutoSDR firmware release vX.XX we also provide the buildroot generated SYSROOT.
  - sysroot-vX.XX.tar.gz
- This allows you to later compile dynamically linked applications that can be executed on the PlutoSDR.



Options to copy files to the PlutoSDR

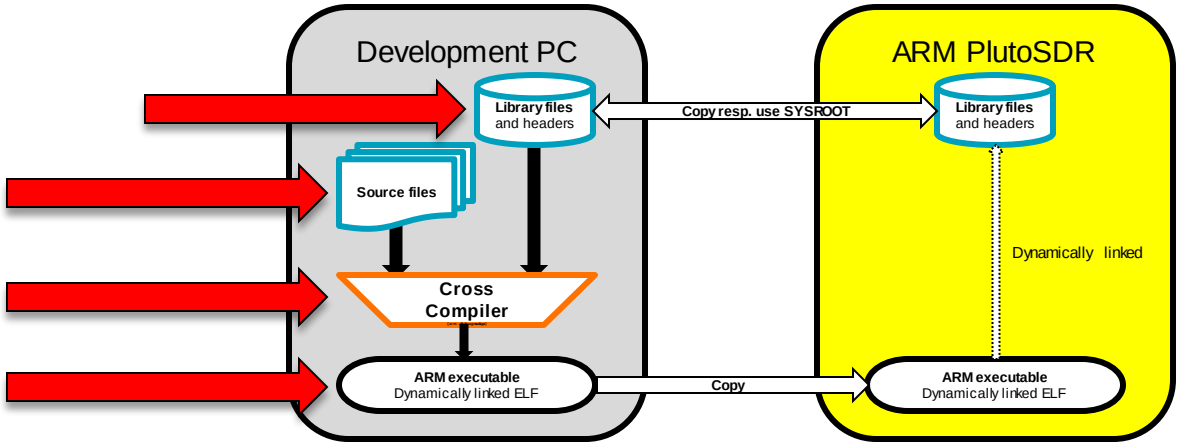
- **Customizing the PlutoSDR filesystem**
- **Scp - Transferring files over SSH**
  - # scp SomeFile root@192.168.2.1:/SomePath
    - Password: *analog*
    - **# sshpass -p analog** scp SomeFile root@192.168.2.1:/SomePath
    - If you host PC supports Avahi/Zeroconf try using hostname: root@pluto
  - SSH key on the PlutoSDR changes every boot. Avoid storing the key using this ssh_config:
    - https://github.com/analogdevicesinc/plutosdr_scripts/blob/master/ssh_config
- USB OTG Host Mode - **Mass Storage Drive Support**
  - Supports FAT/FAT32 filesystems
  - Automount and safe unmount support
  - LED1 mount indicator
  - **Auto Run Support**
    - **runme[XX][.sh]**

```
#!/bin/sh

/media/sda/SomeEXEC
```

runme01.sh

Cross-compiling external applications using sysroot – Example ADS-B dump1090

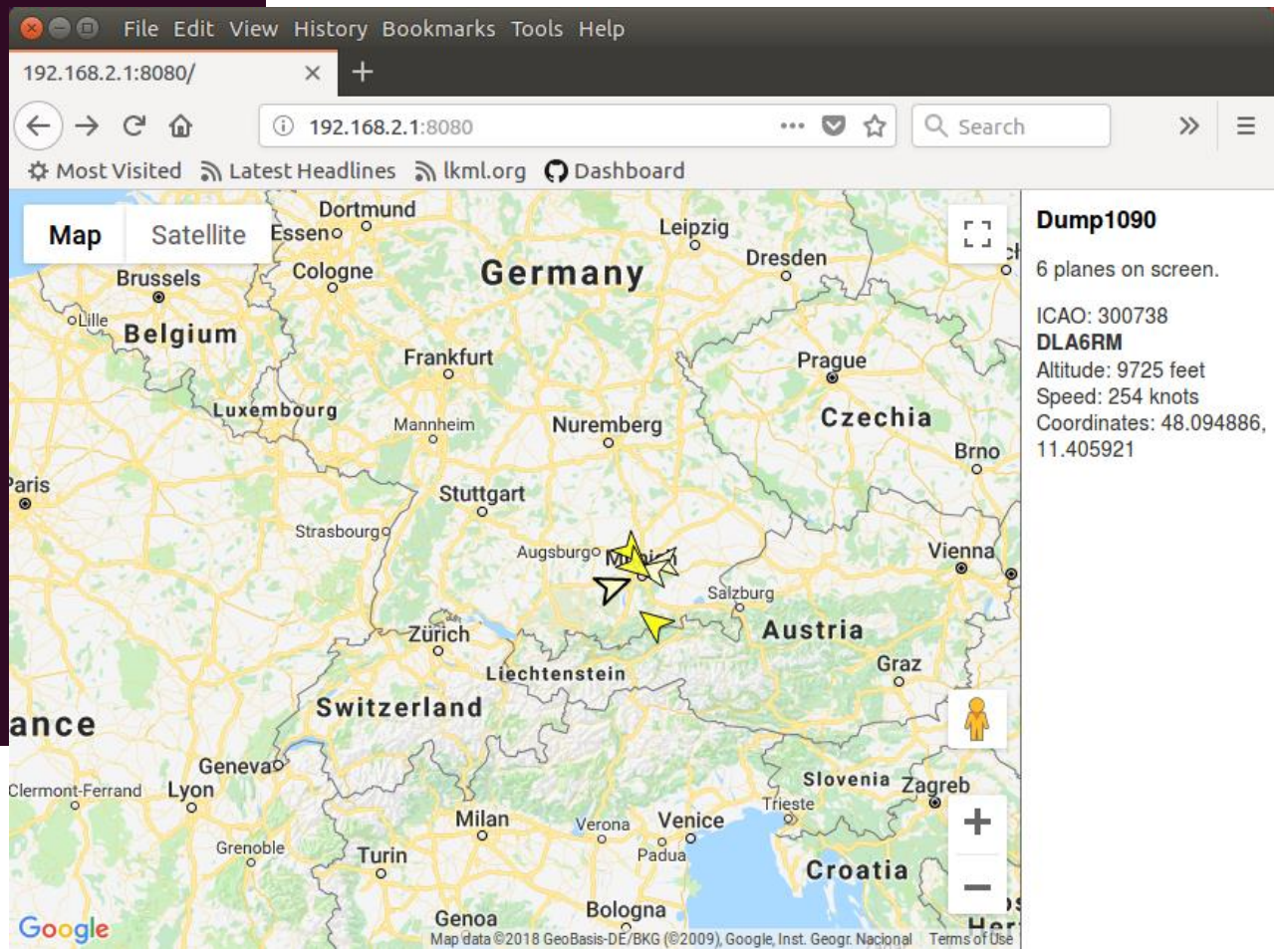
```
Terminal File Edit View Search Terminal Help

~/devel$ wget -q https://github.com/analogdevicesinc/plutosdr-fw/releases/download/v0.29/sysroot-v0.29.tar.gz
michael@HAL9000:~/devel$ tar xzf sysroot-v0.29.tar.gz
~/devel$ git clone -q https://github.com/PlutoSDR/dump1090.git
michael@HAL9000:~/devel$ cd dump1090
michael@HAL9000:~/devel/dump1090$ export PATH=$PATH:/opt/Xilinx/SDK/2017.2/gnu/arm/lin/bin
~/devel/dump1090$ CC=arm-xilinx-linux-gnueabi-gcc CFLAGS=--sysroot=../staging LDFLAGS=--sysroot=../staging make
arm-xilinx-linux-gnueabi-gcc --sysroot=../staging -c dump1090.c
arm-xilinx-linux-gnueabi-gcc --sysroot=../staging -c anet.c
arm-xilinx-linux-gnueabi-gcc -g -o dump1090 dump1090.o anet.o --sysroot=../staging -liio -lpthread -lm -lad9361
michael@HAL9000:~/devel/dump1090$ file dump1090
dump1090: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.3, for GNU/Li
nux 2.6.32, not stripped
~/devel/dump1090$ scp dump1090 root@192.168.2.1:/sbin/
root@192.168.2.1's password:
dump1090                                                    100%   73KB  72.9KB/s   00:00
michael@HAL9000:~/devel/dump1090$
```



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Example Running – ADS-B dump1090

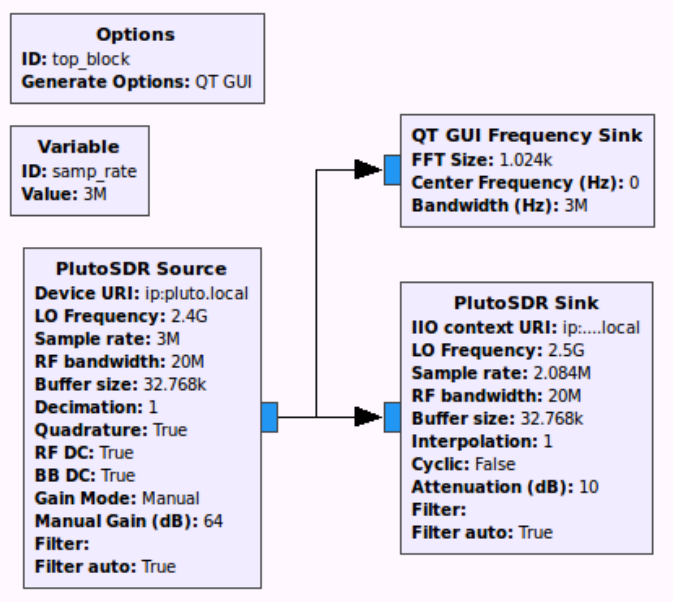
```
Terminal File Edit View Search Terminal Help
pluto login: root
Password:
Welcome to:

 _____ _       _        _________________
|  __ \ |     | |      /  ___|  _  \ ___ \
| |_/ / |_   _| |_ ___ \ `--.| | | | |_/ /
|  __/| | | | | __/ _ \ `--. \ | | |    /
| |   | | |_| | || (_) /\__/ / |/ /| |\ \
\_|   |_|\__,_|\__\___/\____/|___/ \_| \_|

v0.29
http://wiki.analog.com/university/tools/pluto
# dump1090 --net
* Acquiring IIO context
Found 5 device(s):
* Acquiring AD9361 streaming devices
* Acquiring AD9361 phy channel 0
* Initializing AD9361 IIO streaming channels
* Enabling IIO streaming channels
* Creating non-cyclic IIO buffers
```



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

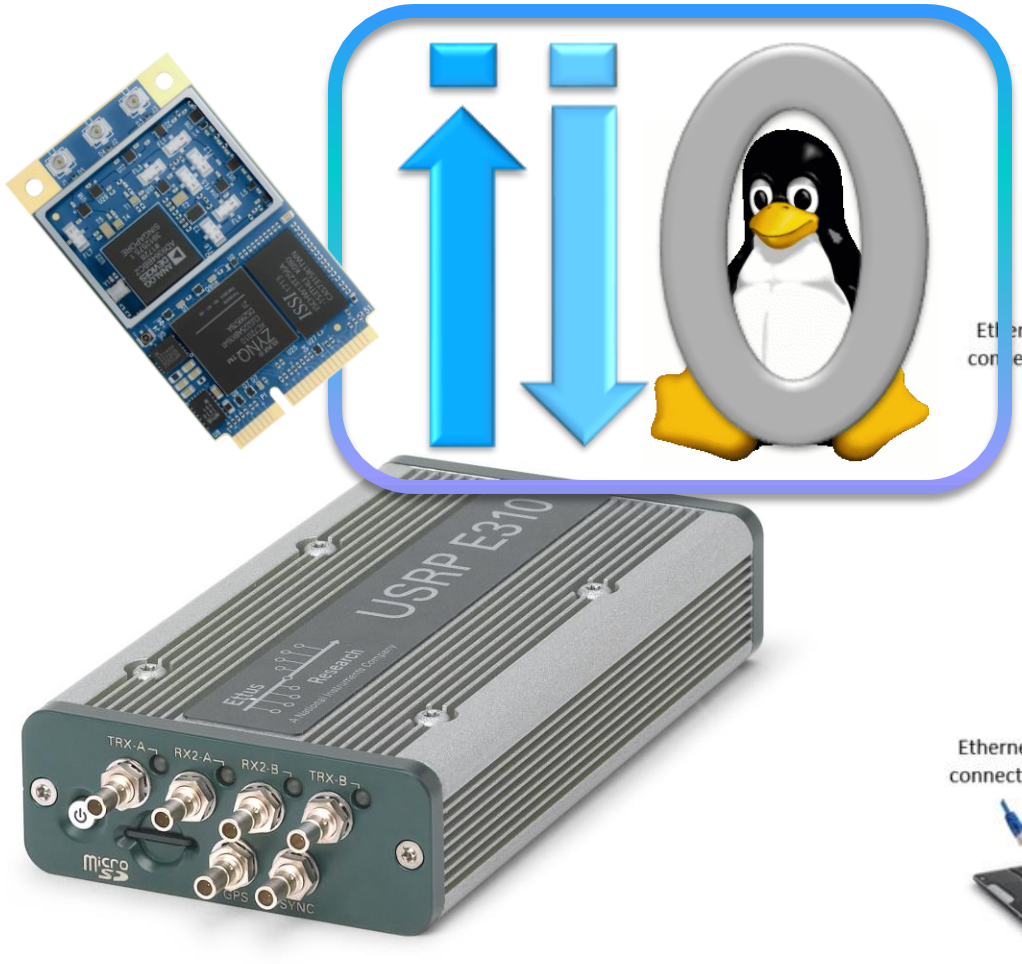
GNU Radio *on* the PlutoSDR: Proof of Concept

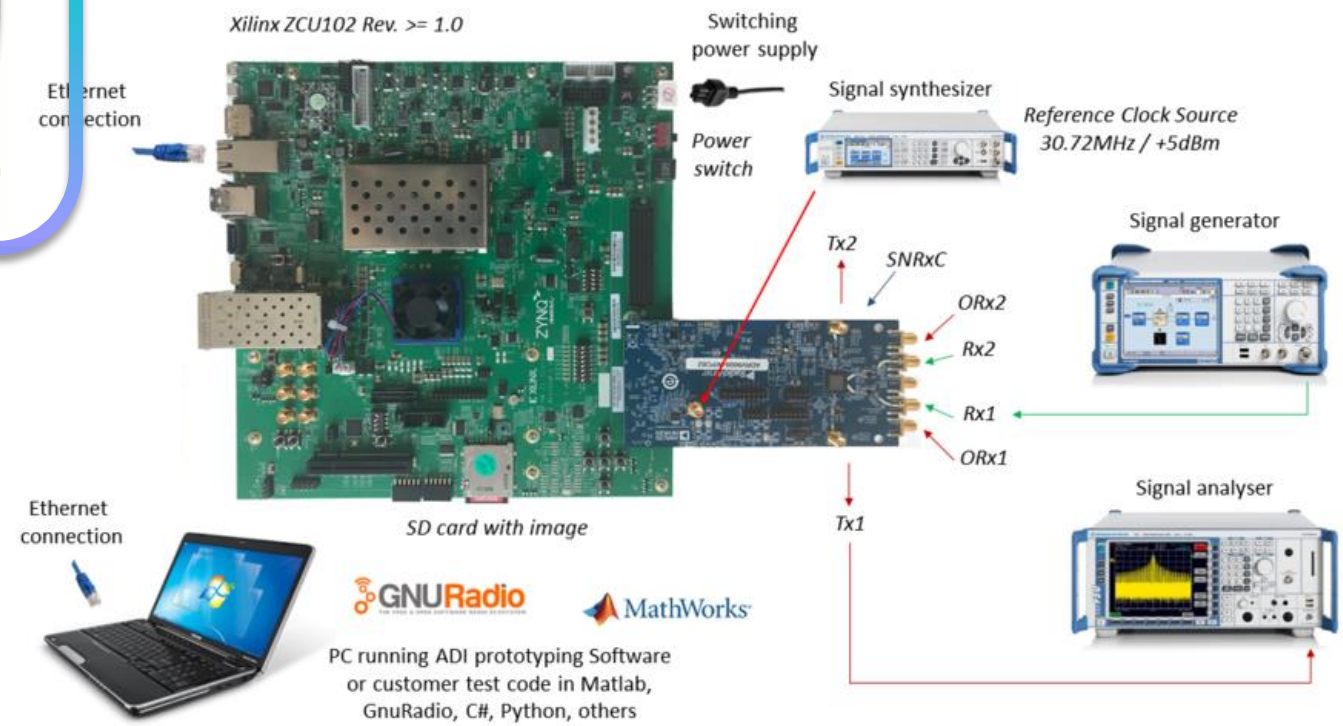
GNU Radio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM



- **Basic concept:**
  - Copy some Ubuntu armhf userland to a USB Flash Drive
  - Enable ext4 filesystem support in the kernel
  - Mount FlashDrive
  - Switching from the PlutoSDR buildroot to the Ubuntu root filesystem
    - Using busybox **switch_root** command
      - chroot into a new filesystem and exec a new init process out of the new filesystem
  - Launch GNU Radio or use apt-get to install it
- **Please see here:**
  - https://ez.analog.com/university-program/f/discussions/98761/gnu-radio-on-the-plutosdr-proof-of-concept

IIO on ARM enabled COTS SDR Transceivers and FPGA/FMC



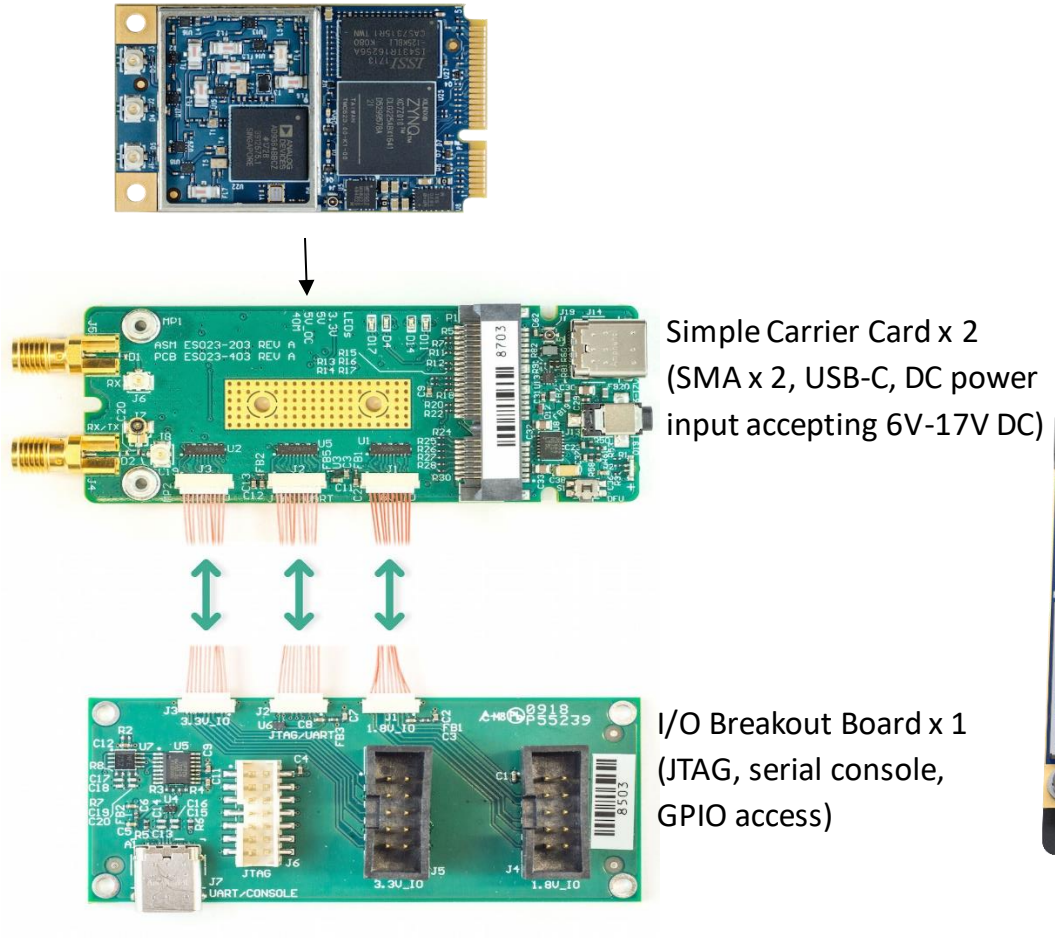


ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Sidekiq Z2 Evaluation Kit

- PlutoSDR Firmware can be build for EPIQ Sidekiq Z2
  - Mini PCIe card form factor
  - AD9364
  - LNA, RF filtering
  - High-precision reference clock

- Follow the PlutoSDR firmware build instructions with the exception that the TARGET variable must be set.
  - **TARGET=sidekiqz2**




```
michael@HAL9000:~/devel/plutosdr-fw$ TARGET=sidekiqz2 make
```

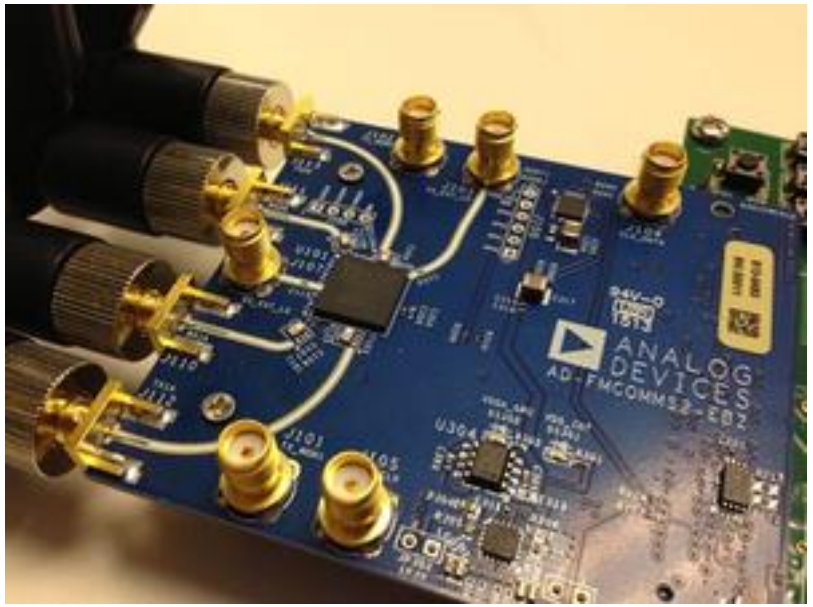
Ettus E310

- Software support
  - RX, TX filter banks
  - USB, Ethernet, RTC, Sensors, LEDs, etc.
- Missing support
  - Half Duplex Antenna switching
  - Software power down
  - Synchronization with PPS time reference

- Building the FPGA boot files
  - Sources
    - https://github.com/analogdevicesinc/hdl/tree/master/projects/usrpe31x
  - Documentation
    - https://wiki.analog.com/resources/fpga/docs/build
    - https://wiki.analog.com/resources/tools-software/linux-software/build-the-zynq-boot-image
- Building the kernel and device tree
  - Sources
    - https://github.com/analogdevicesinc/linux
  - Documentation
    - https://wiki.analog.com/resources/tools-software/linux-build/generic/zynq
      - Kernel config: zynq_e310_defconfig
      - Device tree: zynq-e310.dts

FPGA/FMC

- Analog Devices maintains a number of High Speed Data Acquisition and RF Transceiver reference designs supporting various Intel and Xilinx FPGA carriers:
  - A10 SoC
  - C5 SoC
  - ZCU102
  - KCU105
  - ZC706
  - ZC702
  - Zedboard
  - KC705
  - VC707



- RF Transceivers:
  - ADRV9009
  - ADRV9008-1, ADRV9008-2
  - AD9375
  - AD9371
  - AD9361
  - AD9364
  - AD9363

- https://wiki.analog.com/resources/tools-software/linux-software/zynq_images

Support

EngineerZone™
ANALOG DEVICES SUPPORT COMMUNITY

- https://ez.analog.com/community/university-program
  - ADALM-PLUTO users
- https://ez.analog.com/community/fpga
  - FPGA Developers
- https://ez.analog.com/community/linux-device-drivers/linux-software-drivers
  - libiio users and developers
  - Driver users and developers
- linux-iio@vger.kernel.org
  - IIO mailing list

ANALOG DEVICES

AHEAD OF WHAT'S POSSIBLE™

Q&A

THANKS

VISIT OUR WORKSHOPS

Introduction to the ADALM-PLUTO SDR, Linux's IIO, and Open-Source Toolchains
Tuesday 15:45 - 17:30 & Wednesday 12:45 - 15:15

Systems Programming on the IIO based radios within the IIO Framework
Tuesday 09:30 - 12:00