



Chapter 4. SDR Platform Example: The USRP

1

Chapter 4. USRP - Outline

- General Aspects, Applications
- USRP Architecture
- RF Daughterboard Example: WBX
- Software Environments

General Aspects

- ▶ **USRP – Universal Software Radio Peripheral**
- ▶ SDR family designed and sold by ***Ettus Research*** (bought in 2010 by National Instruments)
- ▶ First generation – USRP1 (2004)
 - ▶ Based on an Altera Cyclone FPGA
 - ▶ Host PC connected using an USB2.0 interface

General Aspects (2)

- 2nd generation: USRP2 (2008)
 - Based on a Xilinx Spartan3-2000 FPGA
 - Gigabit Ethernet Interface

- 3rd generation: USRP N200/210, B200/210 (2011)
 - Two different sub-families: networked and bus
 - Based on Xilinx Spartan-3A FPGAs

- 4th generation: USRP X300/310, E310 (2014)
 - Dual 10 Gigabit Ethernet or PCIe Interfaces
 - Based on Xilinx Kintex7 FPGAs.

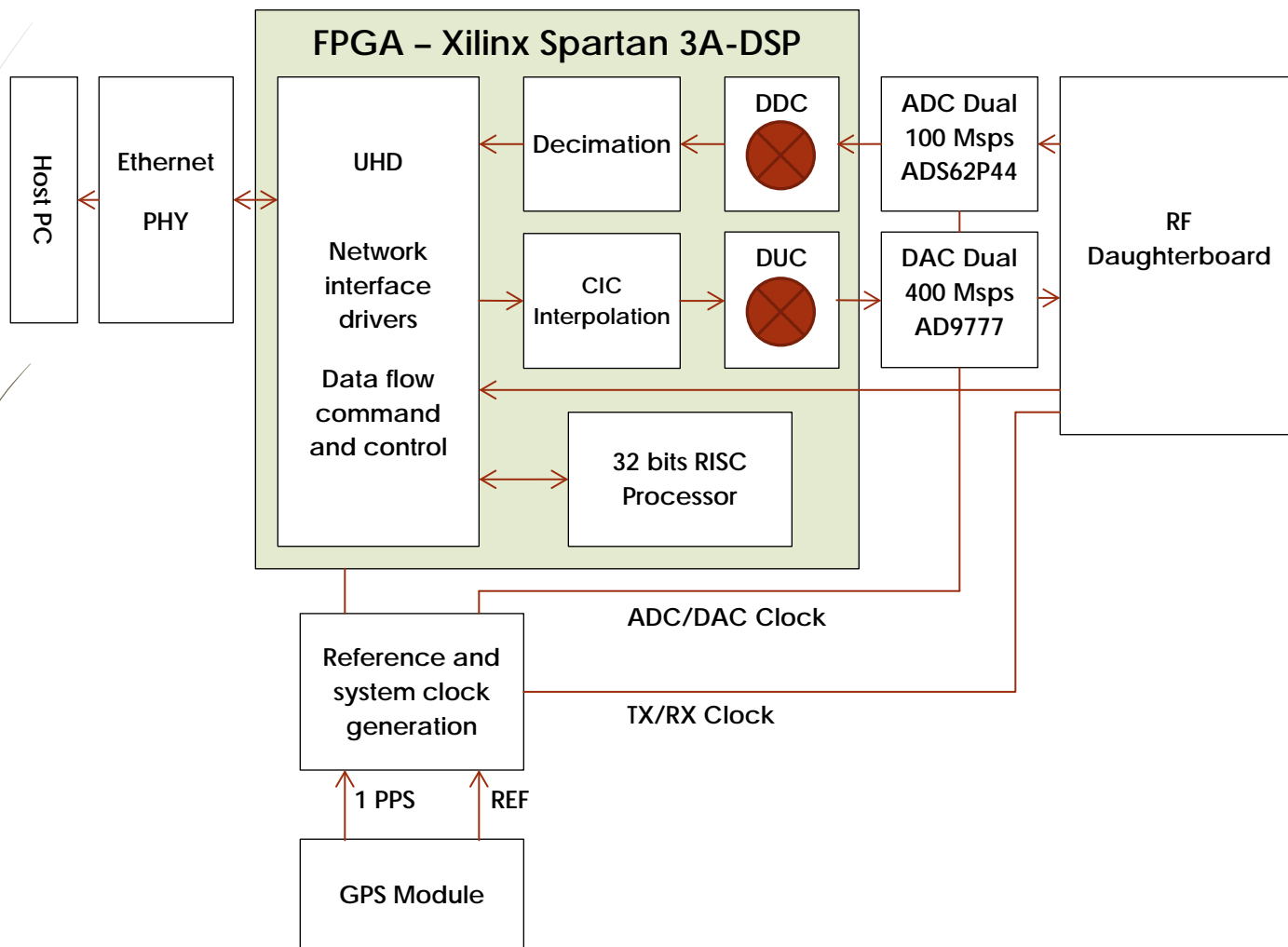
Application examples

- ▶ FM Radio Receiver
 - ▶ <http://www.ettus.com/kb/detail/sdr-for-beginners-building-an-fm-receiver-with-the-usrp-and-gnu-radio>
- ▶ GNSS (GPS, Glonass, Galileo) Receiver
 - ▶ <https://gnuradio.org/redmine/projects/gnuradio/wiki/GlobalPositioningSystem>
- ▶ GSM Base Station (OpenBTS)
 - ▶ <http://openbts.org/>
- ▶ Radar
 - ▶ <http://www.ettus.com/application/detail/usrp-technology-for-multiband-passive-bi-static-radar>
- ▶ ZigBee Transceiver
 - ▶ <http://www.ccs-labs.org/software/gr-ieee802-15-4/>
- ▶ Further examples: <http://cgran.org>

Example: USRP N210

- ▶ Hardware specifications and performance:
 - ▶ Dual 100 MS/s, 14-bit ADC
 - ▶ Dual 400 MS/s, 16-bit DAC
 - ▶ DDC/DUC with 25 MHz Resolution
 - ▶ Up to 50 MS/s Gigabit Ethernet Streaming
 - ▶ 50 MS/s at 8 bit resolution
 - ▶ 25 MS/s at 16 bit resolution
 - ▶ Xilinx Spartan 3A-DSP 3400 FPGA
 - ▶ 1 MB High-Speed SRAM
 - ▶ Frequency Accuracy 2.5ppm

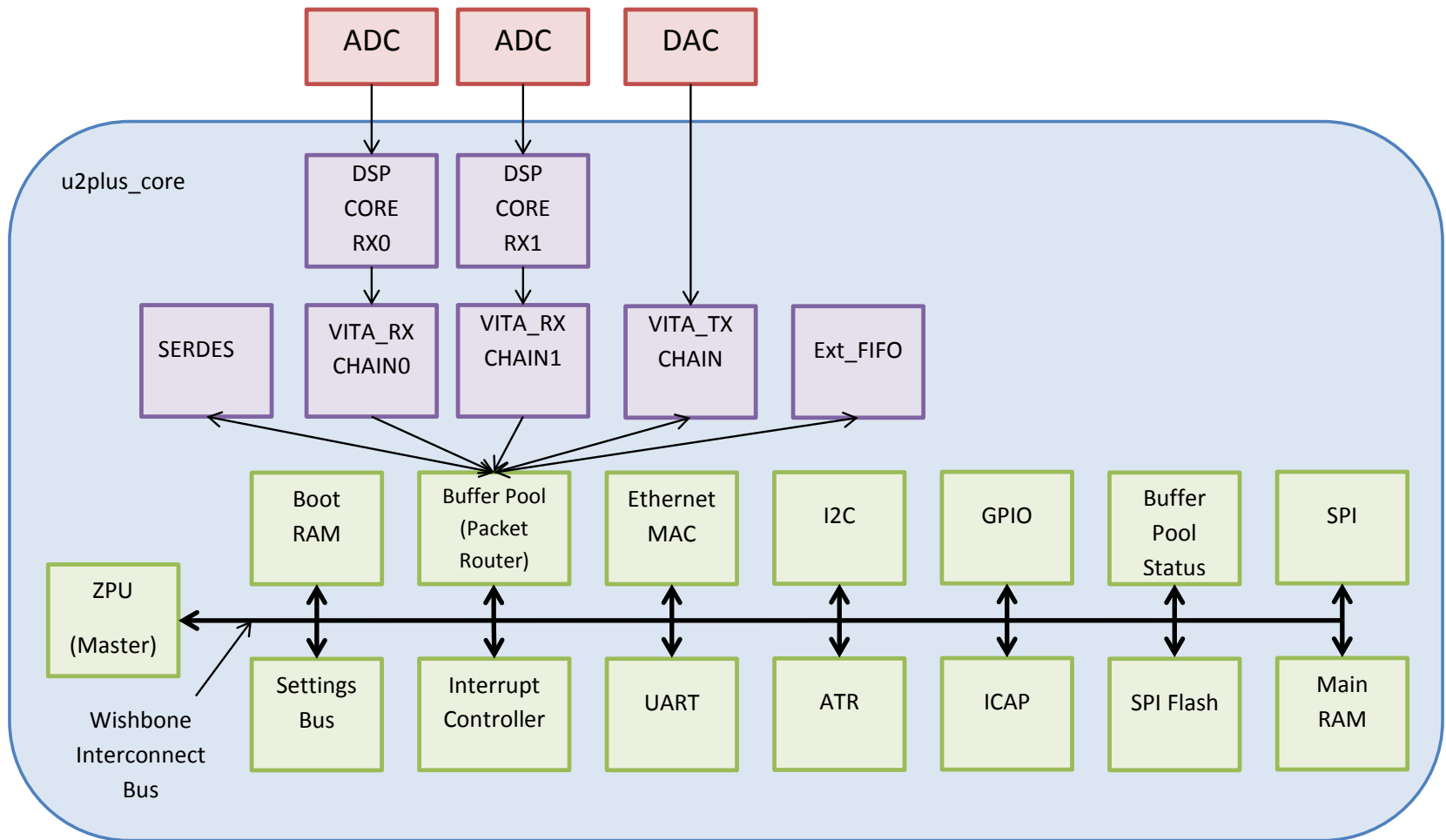
USRP N210 - Block Diagram



USRP N210 - Motherboard

- ▶ Contains the A/D and D/A converters
- ▶ Most of the signal processing corresponding to the steps described in the DFE chapter are implemented in the Xilinx Spartan 3A-DSP 3400 FPGA

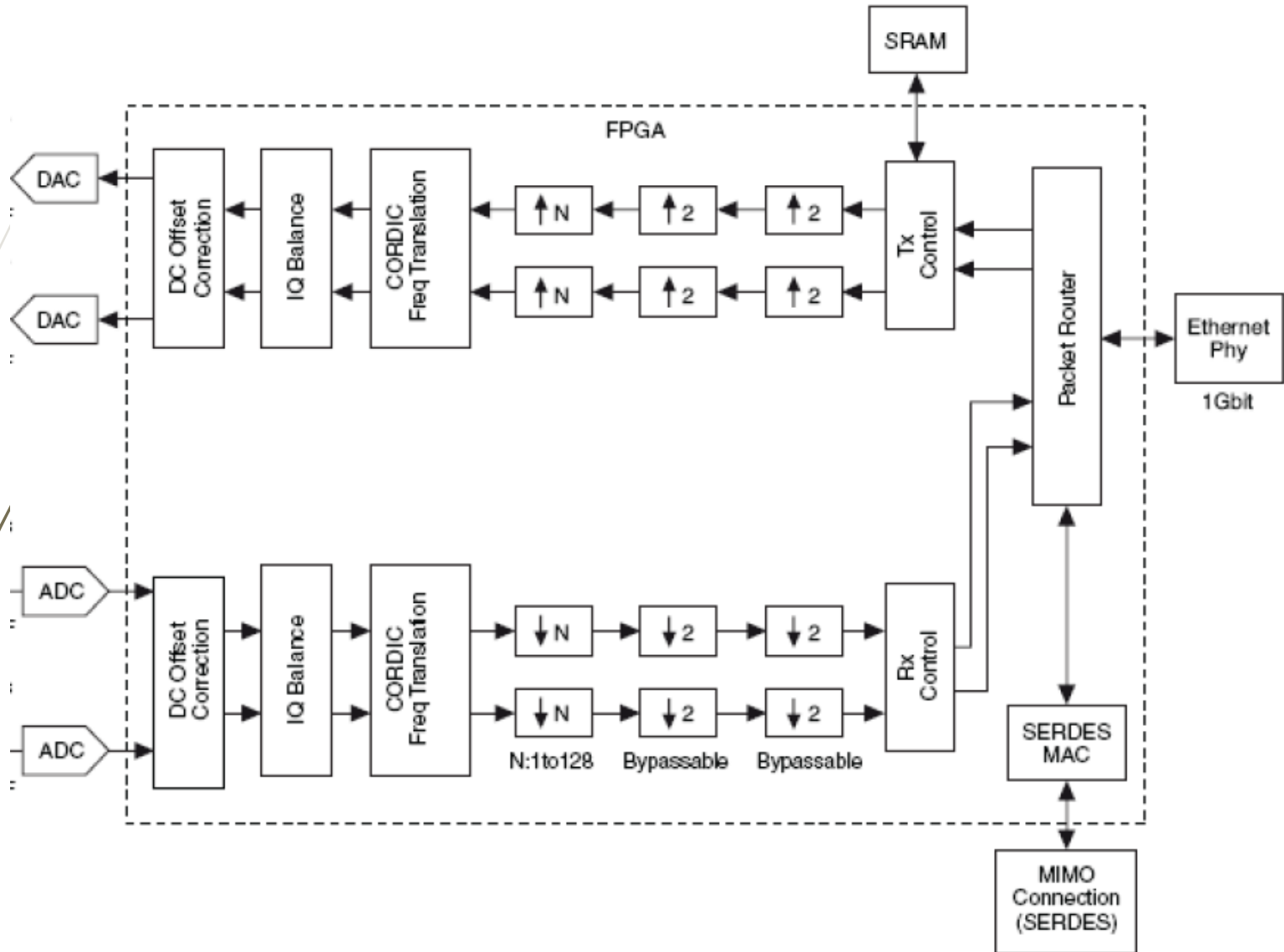
USRP N210 - FPGA



USRP N210 – FPGA (2)

- ZPU – 32bit CPU core
(<http://opencores.org/project,zpu>)
- ICAP - Internal Configuration Access Port
- ATR - Automatic Transmit Receive
- SERDES – Serializer/Deserializer
- GPIO – General Purpose Input/Output
- I2C – Inter-Integrated Circuit bus
- SPI – Serial Peripheral Interface bus
- VITA - VMEbus International Trade Association bus

USRP N210 – FPGA (3)



USRP N210 - Limitations

- Maximum necessary transfer rate over the host PC to USRP interface, in case of Rx:

$$2 \times 100 \text{ Msps} \times 14b = 2.8 \text{ Gsps}$$

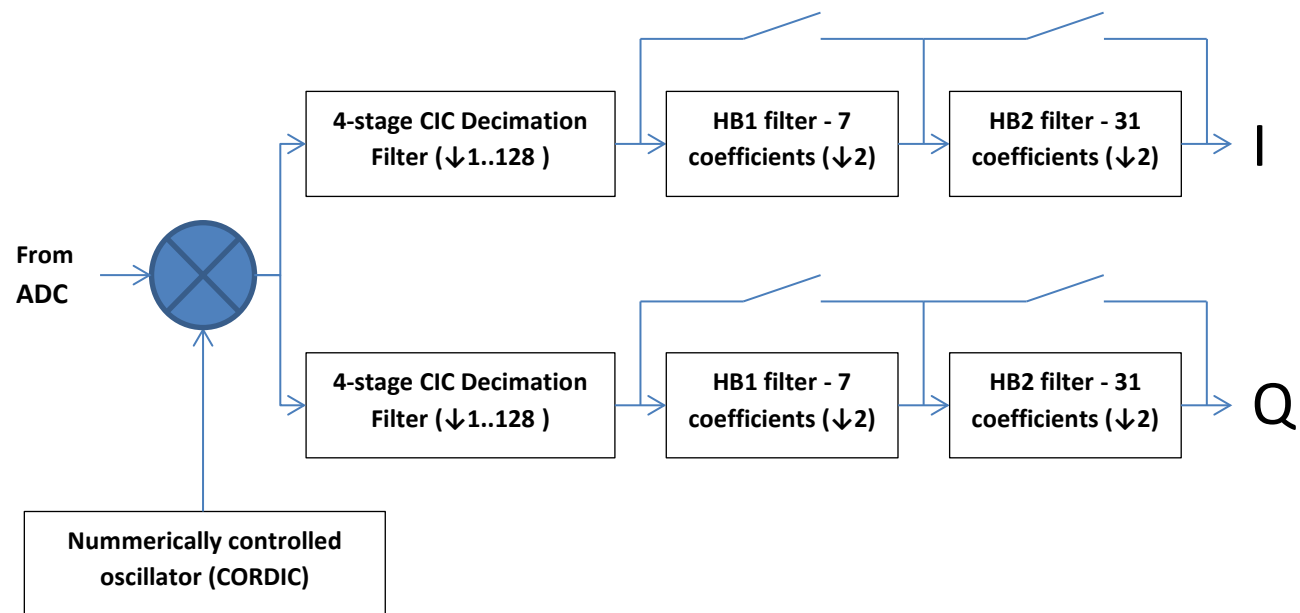
- Considering that the interface host PC-USRP is a 1GB Ethernet one, the maximum available transfer rate in case of Rx is:

$$1/4 \times 2 \times 100 \text{ Msps} \times 14b = 700 \text{ Msps}$$

- A minimum decimation rate of 4 is necessary in order to accommodate the limited available transfer rate

FPGA - DDC

► Digital Down Converter (DDC)



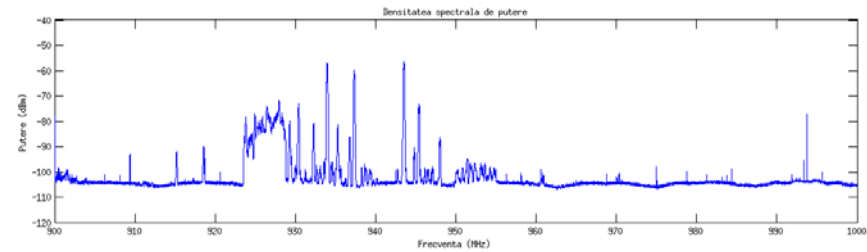
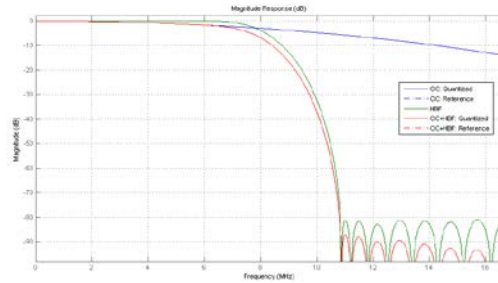
FPGA – DDC (2)

- ▶ DDC structure for different decimation rates

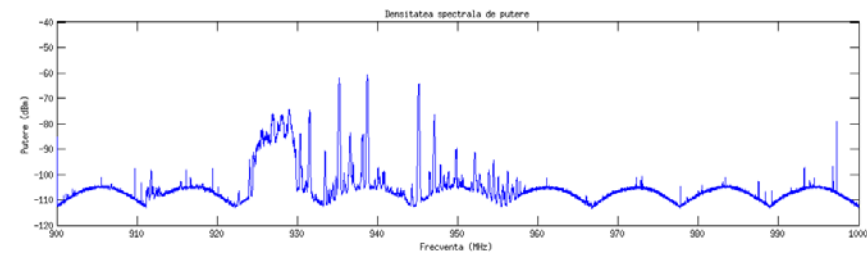
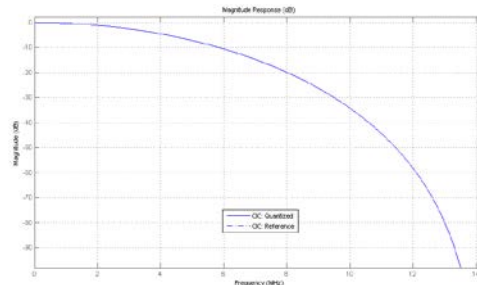
Decimation rate	Obtained sampling frequency	Structure of the DDC
4	25,00 MHz	CIC + HBF1 + HBF2
5	20,00 MHz	CIC
6	16,66 MHz	CIC + HBF1
7	14,28 MHz	CIC
8	12,50 MHz	CIC + HBF1 + HBF2
9	11,11 MHz	CIC
10	10,00 MHz	CIC + HBF1

FPGA – DDC (3)

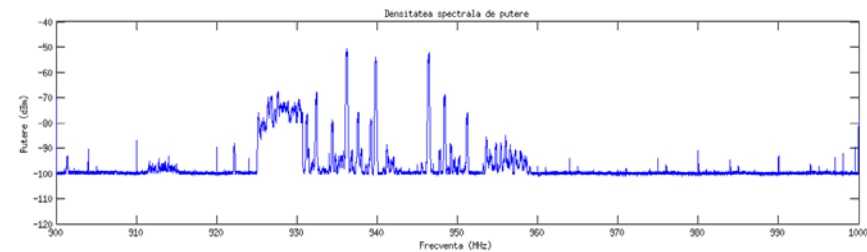
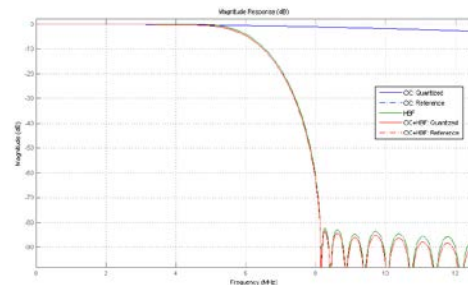
Decimation
rate 6



Decimation
rate 7

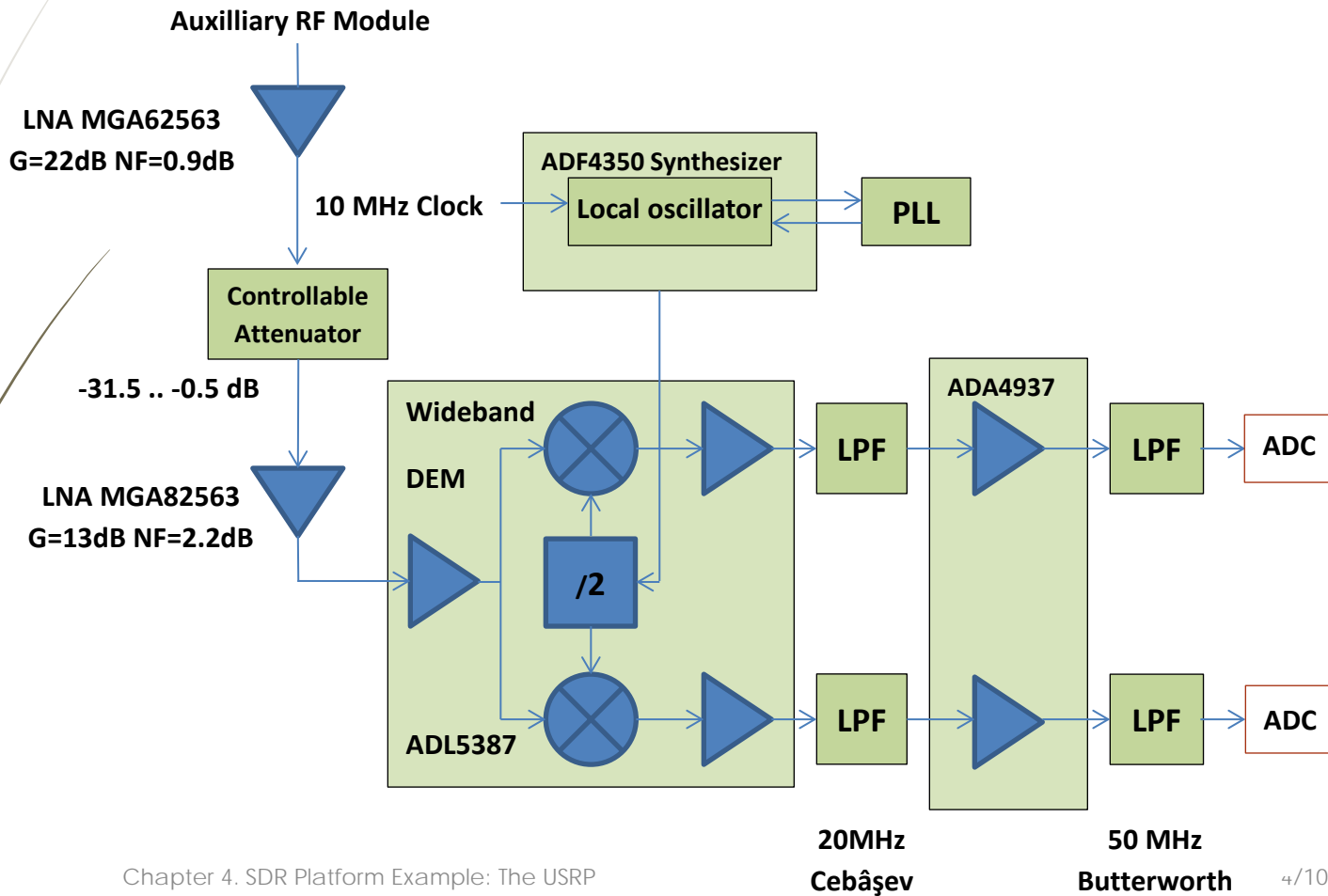


Decimation
rate 8



RF Daughterboard: WBX

Block Diagram – Rx

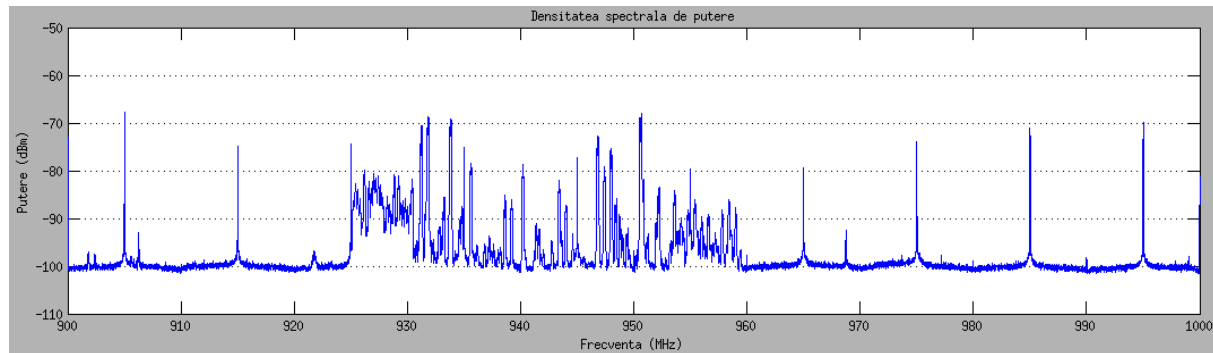


RF Daughterboard: WBX (2)

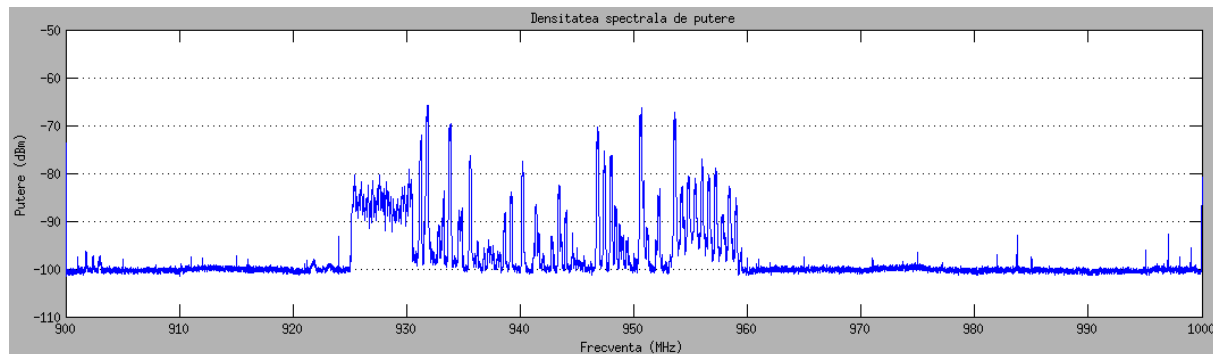
- **Zero IF Architecture** that can be converted in a **Very Low IF Architecture**
 - 50 MHz – 2.2 GHz frequency band
 - Noise Figure: 5-6 dB
 - Sensitivity: -130 dBm
 - IIP2: 40-55 dBm
 - IIP3: 5-10 dBm
 - AGC range: 70 dB

ZIF vs VLIF Architecture

- **ZIF** – parasite components around DC



- **VLIF** – parasite components translated outside the band of interest



Software environments

- ▶ Support for several operating systems (Linux, Windows, Mac OSX, NetBSD, FreeBSD) via the **USRP Hardware Driver (UHD)**
- ▶ Several development environments can be used in conjunction with the USRP hardware:
 - ▶ GNU Radio
 - ▶ Open-source development toolkit that provides signal processing blocks for implementing SDR
 - ▶ Widely used in hobbyist, academic and commercial environments
 - ▶ Matlab (Simulink)
 - ▶ Labview

GNU Radio

- Signal processing blocks implemented in C++
- Library of existing blocks available
 - New blocks can be created and integrated in the library
- Connections between the blocks are typically written in a high-level language (Python)
- A graphical user interface (GNU Radio Companion) , similar to Simulink, is also available

GNU Radio Companion

➤ Example application: FM Radio Receiver

The screenshot displays the GNU Radio Companion interface for an FM Radio Receiver. The main window shows a flow graph with the following components and connections:

- UHD: USRP Source** (Device Address: add...68.10.2, Samp Rate (Sps): 400k, Ch0: Center Freq (Hz): 93.3M, Ch0: Gain (dB): 10) is connected to the **Low Pass Filter**.
- Low Pass Filter** (Decimation: 1, Gain: 1, Sample Rate: 400k, Cutoff Freq: 115k, Transition Width: 30k, Window: Hann, Beta: 6.76) is connected to the **WBFM Receive** block.
- WBFM Receive** (Quadrature Rate: 400k, Audio Decimation: 10) is connected to the **Variable** block (ID: audio_decim, Value: 10) and the **QT GUI Frequency Sink** (Name: FFT Plot, FFT Size: 512, Center Frequency (Hz): 93.3M, Bandwidth (Hz): 400k).
- The **Variable** block is also connected to the **Multiply Const** block (Constant: 1).
- The **Multiply Const** block is connected to the **Audio Sink** (Sample Rate: 40k).

Parameter table:

Options	Parameter	Parameter	Parameter	Parameter	Parameter
ID: uhd_wbfm_receive Title: UHD WBFM Receive Author: Example Description: WBFM Receive Generate Options: QT GUI	ID: address Label: IP Address Value: addr=192.168.10.2 Type: String Short ID: a	ID: samp_rate Label: Sample Rate Value: 400k Type: Float Short ID: s	ID: freq Label: Default Frequency Value: 93.3M Type: Float Short ID: f	ID: gain Label: Default Gain Value: 0 Type: Float Short ID: g	ID: audio_output Label: Audio Output Device Value: Type: String Short ID: 0

QT GUI Range blocks:

- QT GUI Range** (ID: tun_freq, Label: UHD Freq (MHz), Default Value: 93.3, Start: 87.9, Stop: 108.1, Step: 1)
- QT GUI Range** (ID: tun_gain, Label: UHD Gain, Default Value: 10, Start: 0, Stop: 20, Step: 1)
- QT GUI Range** (ID: fine, Label: Fine Freq (MHz), Default Value: 0, Start: -100m, Stop: 100m, Step: 10m)

Component list on the right:

- [Audio]
- [Boolean Operators]
- [Byte Operators]
- [Channelizers]
- [Channel Models]
- [Coding]
- [Control Port]
- [Debug Tools]
- [Deprecated]
- [Digital Television]
- [Equalizers]
- [Error Coding]
- [FCD]
- [File Operators]
- [Filters]
- [Fourier Analysis]
- [GUI Widgets]
- [Impairment Models]
- [Instrumentation]
- [Level Controllers]
- [Math Operators]
- [Measurement Tools]
- [Message Tools]
- [Misc]
- [Modulators]
- [Networking Tools]
- [NOAA]
- [OFDM]
- [Packet Operators]

Terminal output:

```
/home/alexandru/.grc_gnuradio
Loading: "/home/alexandru/target/share/gnuradio/examples/uhd/uhd_fft.grc"
>>> Done
Showing: "/home/alexandru/target/share/gnuradio/examples/uhd/uhd_fft.grc"
Loading: "/home/alexandru/target/share/gnuradio/examples/uhd/uhd_wbfm_receive.grc"
```

Signal Processing Devices

➤ GPP – General Purpose Processors

- Optimized to handle the widest possible range of applications
- Must excel at fixed and floating-point operations, logical operations and branching
- Are suitable for implementing much of the SDR functionality, like physical layer DSP, protocol and network stacks
- Easiest development environments and highest developer productivity
- Wide range of operating system available
- The easiest platform for SDR development
- Major vendors: **Intel, ARM, AMD, MIPS**

Signal Processing Devices (2)

➤ DSP – Digital Signal Processors

- Microprocessors optimized for number crunching
- Optimized for a much narrower set of target applications than GPPs
- Main advantage over GPP is in power consumption per operation
- Not well suited for control intensive code such as protocol and network stack
- A typical SDR would pair a DSP with a GPP to implement the network stack

Signal Processing Devices (3)

➤ DSP – Digital Signal Processors (2)

- The development environment is more complex than in case of GPPs
- OS support is more limited, many project are not using any OS and are interacting with hardware directly
- Optimal use of DSP requires the developer to be very familiar with the internal architecture
- Niche product that do not offer a compelling advantage over either GPPs or FPGAs
- Major vendors: **Texas Instruments, AMD, Freescale**

Signal Processing Devices (4)

▶ FPGA – Field Programmable Gate Array

- ▶ Microchip designed to be configured by the user after manufacture
- ▶ Contains programmable logic components ("*logic blocks*") and reconfigurable interconnect ("*wires*")
- ▶ The overhead of all the wiring and switches makes the FPGA consume significantly more power (10x) than an equivalent single function design
- ▶ Designs implemented on an FPGA also execute a lot slower (5x) than an equivalent single-function design

Signal Processing Devices (5)

► FPGA – Field Programmable Gate Array (2)

- The ability to implement exactly the functionality required for the radio makes FPGAs dramatically more efficient than GPPs
- FPGAs excel at heavy-duty number crunching in a datapath flow, but are not as well suited for control-intensive processing
- The development flow and environment for an FPGA are very different from those used for DSP or GPP

► Major vendors: **Xilinx, Altera, Lattice Semiconductor**

Signal Processing Devices (6)

► Hybrid Solutions

- Devices that combine two or more of the processors described before
- One of the first commercially available examples is the OMAP architecture from **Texas Instruments**
- An OMAP chip combines a GPP with a DSP
- The ZYNQ family of devices from **Xilinx** combines a GPP with an FPGA
- Hybrid solutions do not introduce any new capabilities, but tend to reduce product cost and size