



*University of Trento, Dept. of Information Engineering and Computer Science (DISI)*

Course of:  
**Design of Networks and  
Communication Systems**

Part 1: Physical Layer design and dimensioning

*Lecturer:*

**Claudio Sacchi**, *(Ph.D., assistant professor)*

# Outline

- *Fundamentals of digital communications;*
- *Block diagram description of a digital communication system;*
- *Source and channel coding;*
- *Digital modulation formats;*
- *Link performance parameters;*
- *Software Defined Radio (SDR) concept;*
- *Introduction to the support software: Mathworks MATLAB/SIMULINK;*
- *Introduction to support software: National Instruments LabView Communications;*
- *Software Defined Radio platforms: National Instrument USRP transmission boards.*

# Fundamentals of digital communications

- Definition of digital communication system
  - A communication system is substantially a way of transferring information from one source to another;
  - The components of a communications system serve a common purpose, are technically compatible, use common procedures, respond to controls, and operate in union;
  - In this course, we shall only consider digital communication systems operating over wireless channels;
  - A digital communication system transfers information in digital format to destination.

# Fundamentals of digital communications

- Digital transmission

- Digital communication is based on the transmission of information encoded in an alphabet of finite length. The elements of the alphabet  $S$  are called **symbols**:

$$S = \{S_1, S_2, \dots, S_N\}$$

- A **message** is a set of transmitted symbols belonging to  $S$ :

$$M = \{S_i, S_j, \dots, S_k\}$$

- The symbols are arrays of binary elements (0,1):

$$S_i = \{1, 0, 1, 1, \dots\} \quad \text{length}(S_i) = \log_2(N)$$

# Fundamentals of digital communications

- Digital modulation

- Every information symbol is transmitted by a specific time-varying signal called **waveform**:

$$S_i \rightarrow S_i(t) \quad t \in [0, T)$$

- $T$  is called **symbol duration**. It is always finite. The inverse of  $T$  ( $1/T$ ) is called **symbol emission rate** or **baud-rate** (measured in baud/sec).
- The function that associates an information symbol to a waveform is called **digital modulation**.

# Fundamentals of digital communications

- Radio-frequency (RF) transmission

- The digital radio transmission is always performed in **radio frequency (RF)**;
- Actually, the waveform modulates the amplitude or the frequency, or the phase a sinusoidal signal called **carrier**. The frequency of the carrier, denoted by  $f_c$ , is called **radio frequency**. The modulated carrier is the signal transmitted by the antenna:

$$S_i \rightarrow S_i(t) \ t \in [0, T) \Rightarrow X_i(t) = F[S_i(t), A_c \cos(2\pi f_c t + \theta)]$$



*Modulated carrier*



*Modulating function*

# Fundamentals of digital communications

- Modulating function

- Amplitude modulation:

$$F[S_i(t), A_c \cos(2\pi f_c t + \theta)] = A_c [\alpha S_i(t)] \cos(2\pi f_c t + \theta)$$

- Frequency modulation:

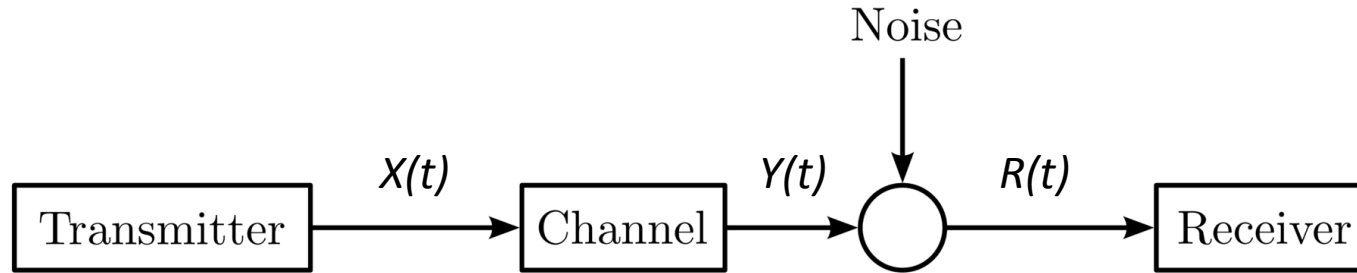
$$F[S_i(t), A_c \cos(2\pi f_c t + \theta)] = A_c \cos\left(2\pi \left\{f_c [\beta S_i(t)]\right\} t + \theta\right)$$

- Phase modulation:

$$F[S_i(t), A_c \cos(2\pi f_c t + \theta)] = A_c \cos\left(2\pi f_c t + \theta + [\gamma S_i(t)]\right)$$

# Block diagram description

- First level block diagram:



**Transmitter** contains everything is needed to generate **waveforms** bearing information bits

**Channel** is the signal propagation mean, **with all the signal degradation sources**

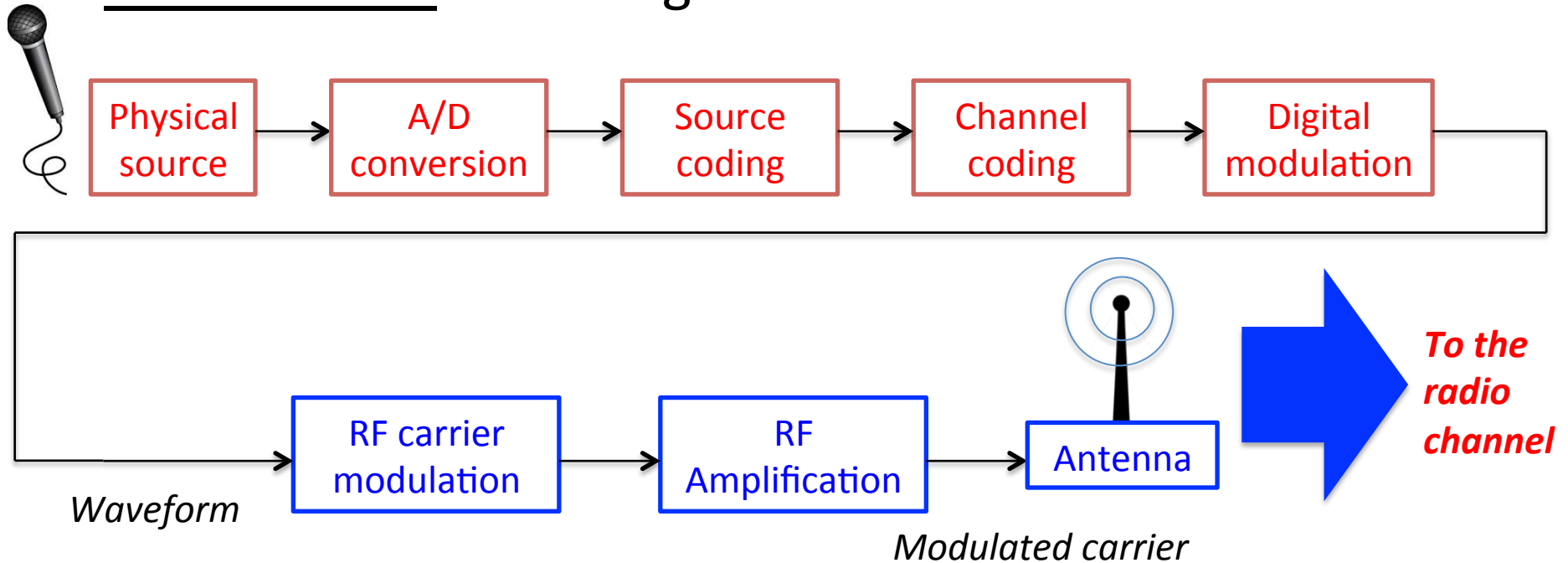
**Noise** represent the contribution **of all unwanted signals not related to** the transmitted information

**Receiver** is the part of the system devoted at recovering the transmitted and **corrupted** information



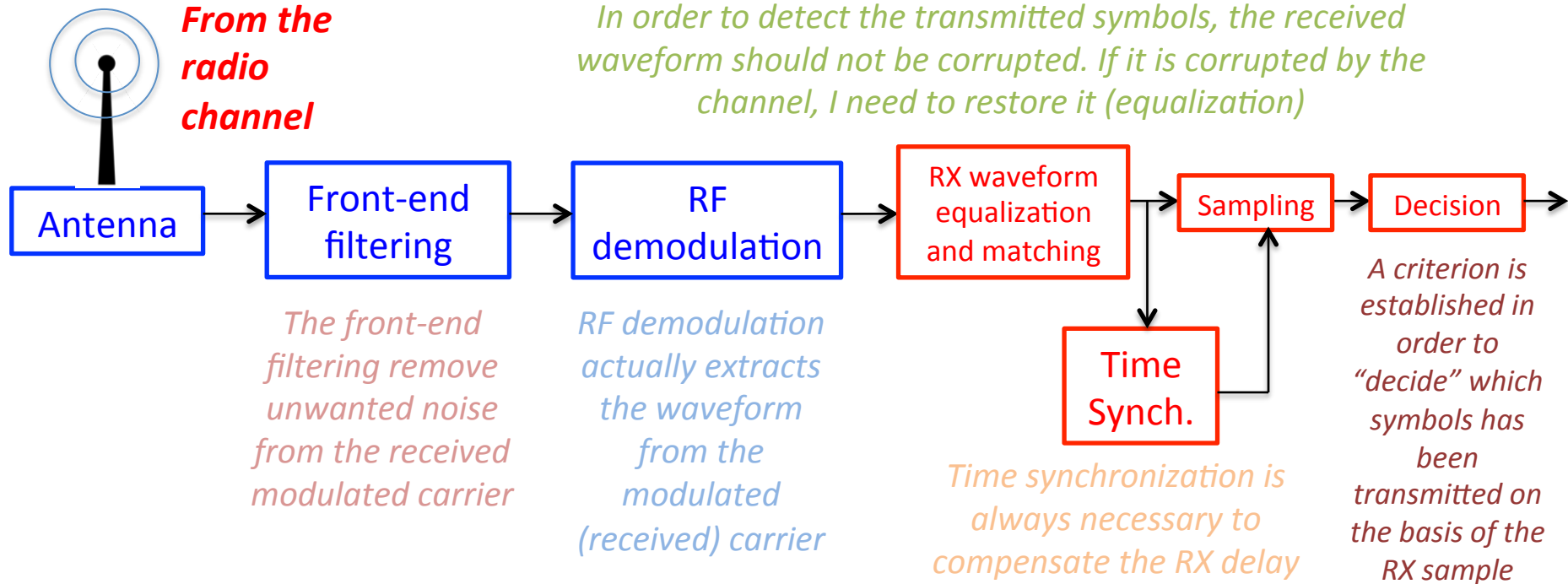
# Block diagram description

- Second level block diagram: **transmitter**



# Block diagram description

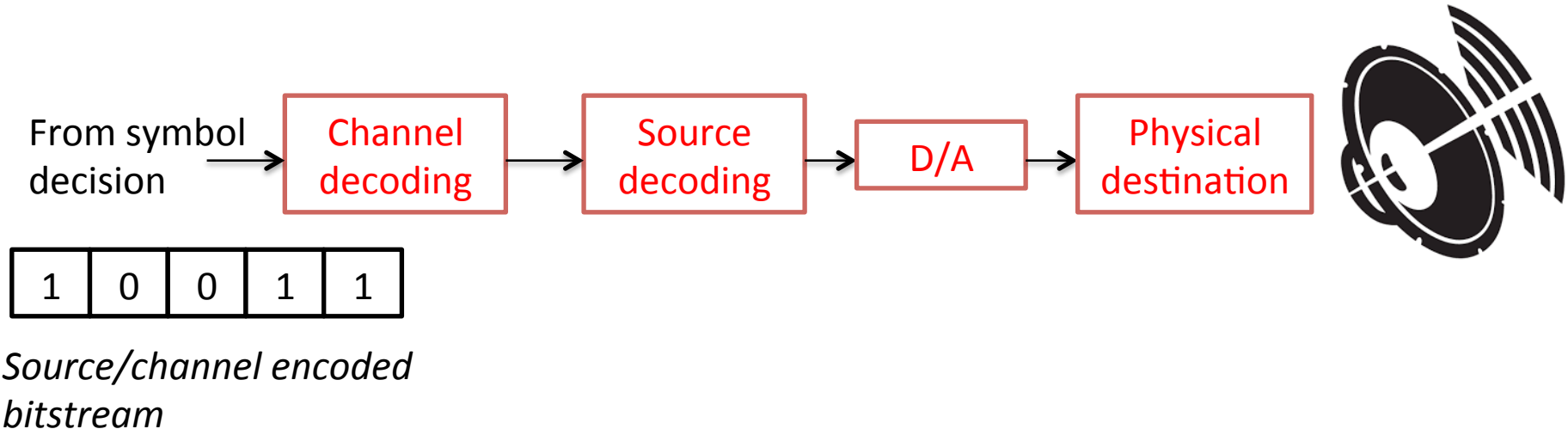
- Second level block diagram: receiver



# Block diagram description

- Decoding

- After symbol decision, we have the **decoding operations**:



# Source coding

- Source coding is explicitly required when multimedia information is transmitted. It compresses the information and reduces the bitrate;
- Indeed, the bitrate of uncompressed PCM audio and video sources is intolerable for transmission over real channels;
- The price to be paid is a loss of quality after decoding (“lossy” coding), but, often, such a loss is not perceivable. Source-coded signals are generally very prone to channel errors. Bit-error-rate should be kept conveniently low;
- The best known source-coded audio and video formats are:
  - **AMR, AMR-WB G.722, Speex** for speech coding (telephony, audio-conferencing, Skype);
  - **MP3** for high quality stereo music listening;
  - **MPEG2, MP4, H.264** for high-quality video broadcasting.

# Channel coding

- Typically noise and propagation impairments introduce errors in the received bit-stream;
- The bit-error-rate (BER) is the quality measure of communication systems. High BERs are not tolerable, in particular for source-coded signals;
- In order to correct errors generated by noise and other impairments present in the communication link, we should resort to channel coding, known also as **forward error correction** (FEC) coding;
- Channel coding is based on 2<sup>nd</sup> Shannon's theorem: adding redundancy to the message, we can **theoretically** remove all bit errors. In practice, some arrangements are required to effectively implement channel coding over real channels.

# Digital modulation formats

- The best-known and widest-use digital modulation formats are:
  - Amplitude-Shift Keying (ASK), known also as Pulse-Amplitude Modulation (PAM);
  - Phase-Shift Keying (PSK);
  - Amplitude-Phase Keying (APK), known also as Multi-level QAM (M-QAM).
- Let's introduce first the most generic analytical form for a digital modulation:

$$X(t) = \sum_{k=-\infty}^{+\infty} \alpha_k f(t - kT) A_c \cos(2\pi f_c t + \theta + \phi_k) = \operatorname{Re} \left\{ \left[ \sum_{k=-\infty}^{+\infty} \alpha_k e^{j\phi_k} f(t - kT) \right] A_c e^{j(2\pi f_0 t + \theta)} \right\}$$

$$S_k(t) = \alpha_k e^{j\phi_k} f(t - kT) \in \mathbb{C} \quad \text{Waveform associated to the } k^{\text{th}} \text{ transmitted symbol}$$

$$f(t) \quad \text{Pulse shaping function}$$

# Digital modulation formats

- Low-pass equivalent signal

- We can express the generic modulation format as follows:

$$\begin{aligned} X(t) &= A_c \sum_{k=-\infty}^{+\infty} \alpha_k \cos(\phi_k) f(t - kT) \cos(2\pi f_c t + \theta) - A_c \sum_{k=-\infty}^{+\infty} \alpha_k \sin(\phi_k) f(t - kT) \sin(2\pi f_c t + \theta) = \\ &= A_c I(t) \cos(2\pi f_0 t + \theta) - A_c Q(t) \sin(2\pi f_0 t + \theta) = \operatorname{Re} \left\{ [I(t) + jQ(t)] A_c e^{j(2\pi f_0 t + \theta)} \right\} \end{aligned}$$

$$s_{lp}(t) \hat{=} [I(t) + jQ(t)] \in \mathbb{C} \quad \text{Low-pass equivalent (complex) signal}$$

Phase component

Quadrature component

$$\text{PLEASE, NOTICE THAT: } X(t) = \operatorname{Re} \left\{ s_{lp}(t) A_c e^{j2\pi f_0 t} \right\}$$

# Digital modulation formats

## ● Modulation formats in analytical details:

### ● ASK (PAM):

$$\alpha_k = A_k, \phi_k \equiv 0 \quad k = 1, 2, \dots, N$$

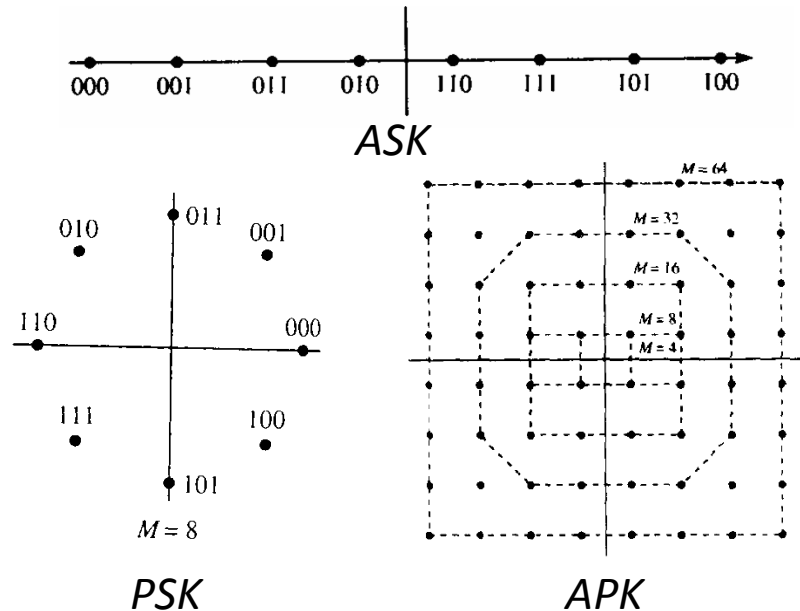
### ● PSK:

$$\alpha_k \equiv 1, \phi_k \equiv \frac{2\pi(k-1)}{N} \quad k = 1, 2, \dots, N$$

### ● APK (M-QAM)

$$\alpha_k \equiv A_k, \phi_k \equiv \frac{2\pi(k-1)}{N} \quad k = 1, 2, \dots, N$$

*Symbol representation in the complex plan (constellations)*





# Digital modulation formats

- Pulse shaping function

- The pulse shaping function is critical in determining the performance of the digital modulation system;
- In order to avoid inter-symbol interference (ISI), pulse shaping function **must** satisfy the Nyquist's conditions:

$$f(0) = 1 \quad f(kT) = 0 \quad \forall k \in \mathbb{Z}, k \neq 0$$

- The basic pulse shaping function is the rectangular pulse, i.e.:

$$f(t) = \Pi\left(\frac{t}{T}\right) = \begin{cases} 1 & -T/2 \leq t \leq T/2 \\ 0 & \textit{otherwise} \end{cases}$$

# Digital modulation formats

- General format for the received modulated signal

- Starting from the previous assumptions, we can derive a general form of the received signal in a communication system:

$$r(t) = LA_c \sum_{k=-\infty}^{+\infty} \alpha_k g(t - \tau - kT) \cos(2\pi f_r t + \phi_k + \beta) + n(t)$$

- $L$  is the **amplitude gain** (in general  $L < 1 \Rightarrow$  attenuation);
- $g(t)$  is the **corrupted pulse shaping function** (distortion);
- $\tau$  is the **propagation delay**;
- $f_r$  is the **received carrier frequency** (different from  $f_c$ , please notice!)
- $\beta$  is the **carrier phase jitter**, encompassing also the propagation delay;
- $n(t)$  is the **overall noise**.

**NOTICE:** the corrupted pulse shaping function may not respect the Nyquist's conditions. Therefore, we may have ISI.

# Link performance parameters

- PHY-layer quality parameters

- Ideally, the physical layer should be error-free, i.e.: the symbol sequence “decided” by the receiver should be exactly equal to the transmitted symbol sequence;
- Unfortunately, the decision is made on a received signal sample **distorted and contaminated by noise**. Therefore, the probability of making a wrong decision does not equal to zero;
- Now, we are considering the quality parameters observed by simulations and on-field testing (empirical evaluation, not analytical). They are:
  - **SYMBOL-ERROR-RATE (SER)**, defined as the ratio between the number of wrong symbol decisions and the total number of transmitted symbols;
  - **BIT-ERROR-RATE (BER)**, defined as the ratio between the number of erroneously decoded bits (0 instead of 1 or 1 instead of 0) and the total number of transmitted bits.
- Both these parameters require the a-priori knowledge of the transmitted messages that is possible only in controlled environment (simulations and/or in-lab emulations).

# Link performance parameters

- Link between SER and BER

- In PSK modulation, adopting the Gray coding for bit to symbol allocation (i.e. 1-bit distance between adjacent symbols in the complex plane), a very good approximation of BER is computed as follows:

$$BER \approx \frac{SER}{\log_2(N)}$$

- In case of ASK and APK modulation formats, the aforesaid approximation is still acceptable, but not so good.

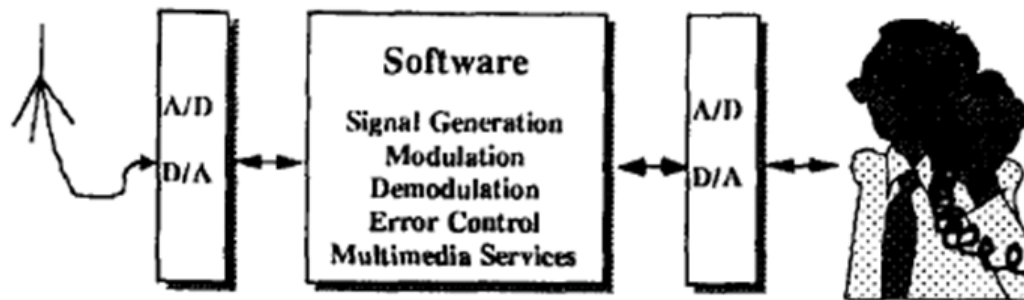
# Link performance parameters

- Packet error (loss) rate
  - This is not truly a parameter directly linked to the physical layer (packets are processed by higher layers);
  - However, if the received information is accessible not in terms of bits, just in terms of packets, it is possible to measure the **packet-error-rate** (PER) that in some applications coincides with the **packet-loss-rate** (PLR);
  - PER can be approximately linked to BER, assuming that the majority of packets is wrong because 1 bit of them is wrong, i.e.:

$$PER \approx BER \cdot packet\_length$$

# The Software Defined Radio (SDR) concept

- Mitola's paradigmatic concept of "Software Radio"
  - J. Mitola enunciated his revolutionary idea in a first paper dated 1993;
  - The SDR paradigmatic approach of Mitola can be summarized by this picture:



*Big question: is it feasible? Because, if it would be feasible, we might be able to design our PHY-layer implementing software code and by not assembling hardware components!!*

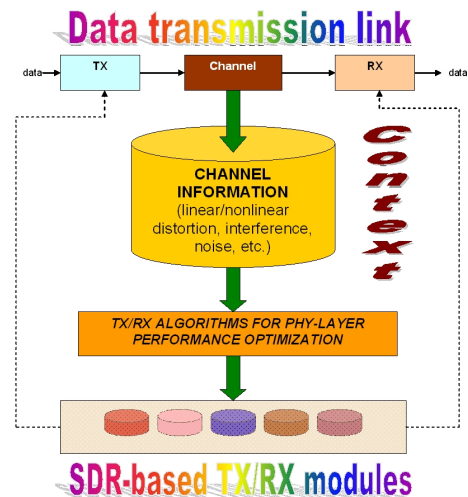
# The Software Defined Radio (SDR) concept

- Answer to the previous “big question”

- The current state-of-the-art of general-purpose embedded signal processing architectures allows the SDR implementation only for baseband signal processing TX/RX tasks.
- Indeed, the A/D conversion of 2 GHz modulated RF carriers would require sampling rates of the order of 40 GHz (not affordable);
- But, the sampling rate of baseband waveforms is limited to some tenths of MHz e.g. for LTE signals. This is fully affordable;
- Anyway, all the crucial tasks of digital transmission are performed in the baseband digital domain, namely:

- Source coding;
- Channel coding;
- Symbol generation;
- Waveform synthesis;
- Time synchronization;
- Equalization and waveform matching;
- Decision;
- Channel decoding;
- Source decoding.

SDR could be a great deal for future mobile communications



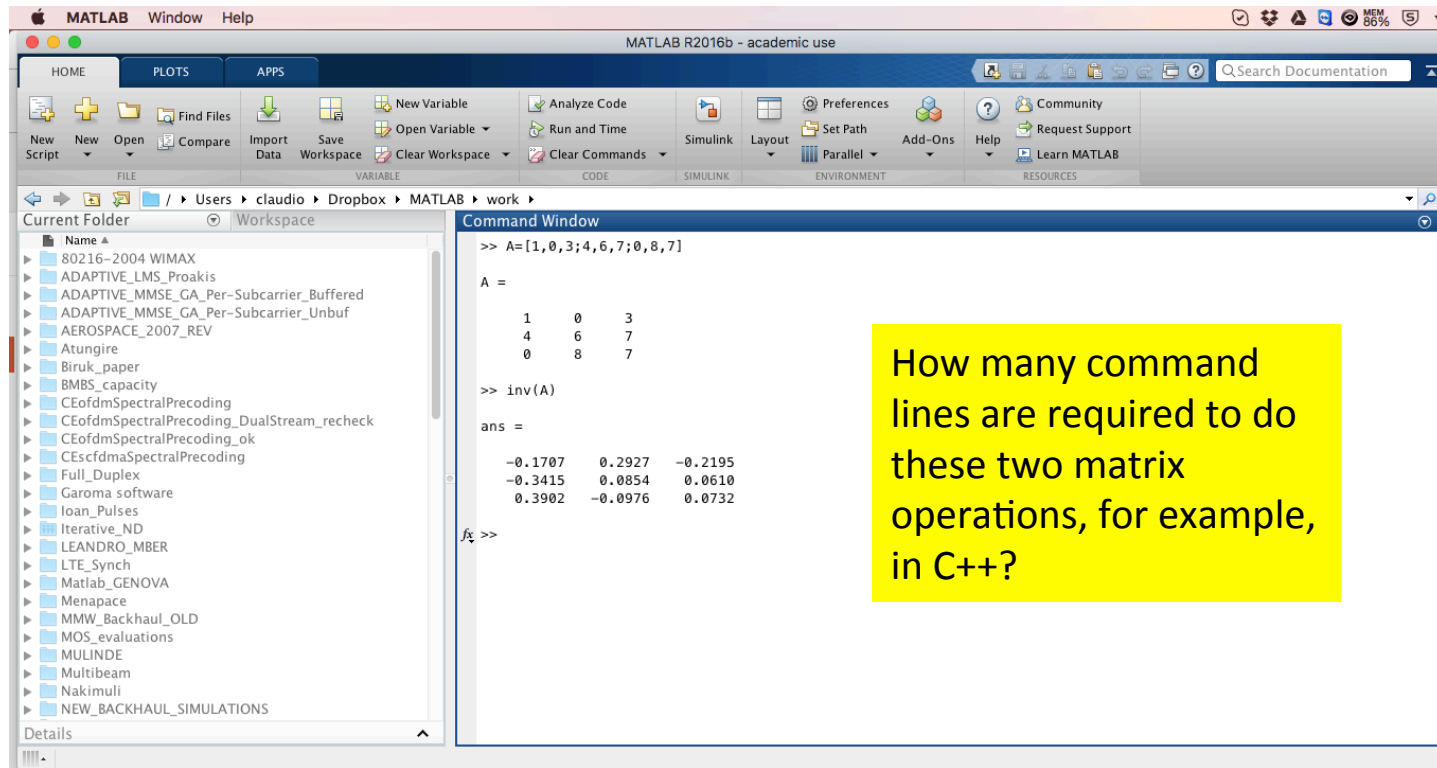
# Introduction to the software: MATLAB/SIMULINK

- The typical academic simulation environment
  - MATLAB is the result of a project of US Mathworks, whose first release was issued in 1984;
  - The fundamental object of MATLAB is the **ARRAY**, whose declaration and dimensioning is not explicit, like in C and C++. Actually, the array (or the matrix) is defined by the programmer simply by inserting the row-column numerical values;
  - The array processing is overall managed by library functions and by functions defined by the programmer. The strong characteristics of MATLAB are the ease of use, the linearity in programming. The weaknesses are related to the low execution speed (abstraction is paid in terms of CPU computational burden and memory occupation!)



# Introduction to the software: MATLAB/SIMULINK

## ● MATLAB interface (trivial example)



The screenshot displays the MATLAB R2016b interface. The top menu bar includes 'MATLAB', 'Window', and 'Help'. Below the menu is a toolbar with icons for 'HOME', 'PLOTS', and 'APPS'. The main workspace area shows a file browser on the left with a tree view of folders, including '80216-2004 WIMAX', 'ADAPTIVE\_LMS\_Proakis', and 'NEW\_BACKHAUL\_SIMULATIONS'. The right pane is the 'Command Window', which contains the following text:

```
>> A=[1,0,3;4,6,7;0,8,7]
A =
     1     0     3
     4     6     7
     0     8     7
>> inv(A)
ans =
    -0.1707    0.2927   -0.2195
   -0.3415    0.0854    0.0610
    0.3902   -0.0976    0.0732
```

A yellow text box on the right side of the screenshot contains the following text:

How many command lines are required to do these two matrix operations, for example, in C++?

# Introduction to the software: MATLAB/SIMULINK

## ● MATLAB toolboxes

- MATLAB is formed by a core of basic functions for array processing, but it provides also some specialist libraries called TOOLBOXES that allows to implement more complex functionalities inherent to telecommunications, signal processing, image processing, control, optimization, etc.
- At the right hand, you can see a list of installed toolboxes of MATLAB 2016b version.

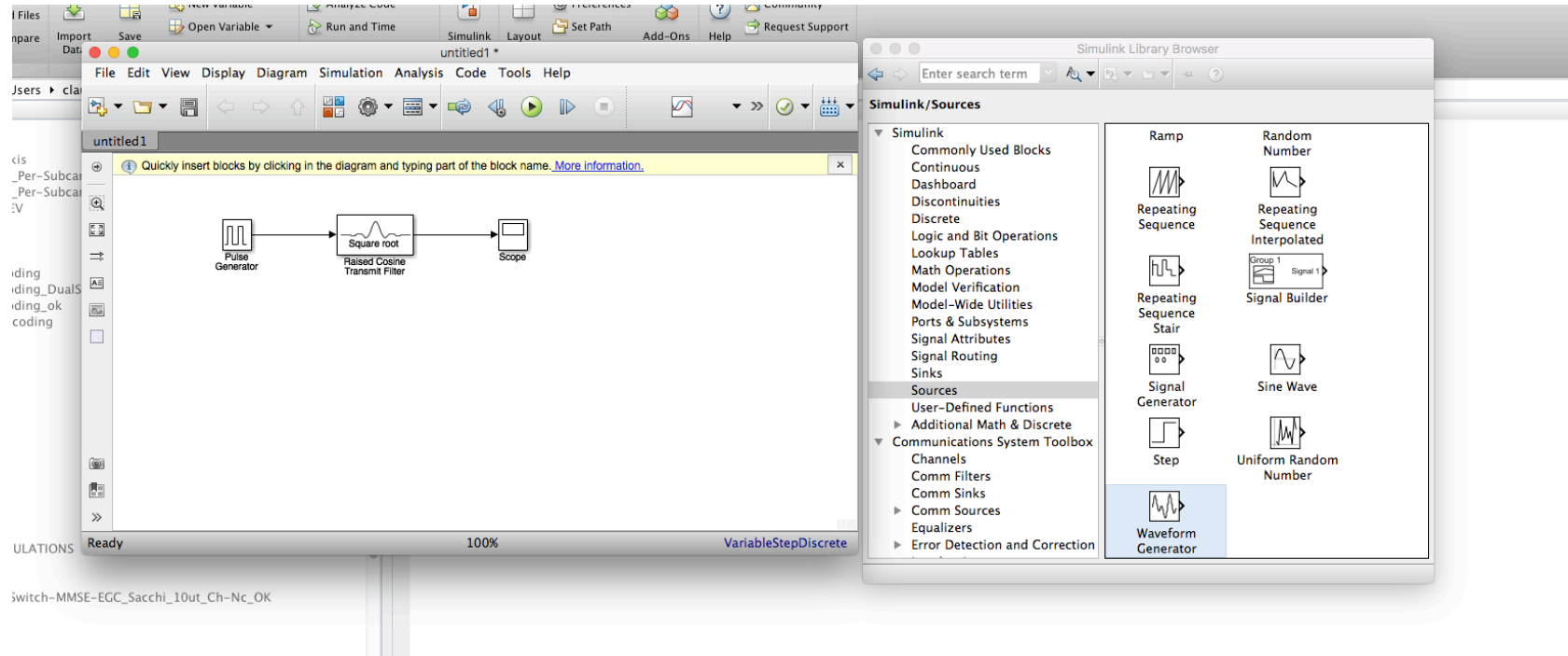
|   |               |          |
|---|---------------|----------|
| Bioinformatics Toolbox                  | Version 4.7   | (R2016b) |
| Communications System Toolbox           | Version 6.3   | (R2016b) |
| Computer Vision System Toolbox          | Version 7.2   | (R2016b) |
| Control System Toolbox                  | Version 10.1  | (R2016b) |
| Curve Fitting Toolbox                   | Version 3.5.4 | (R2016b) |
| DSP System Toolbox                      | Version 9.3   | (R2016b) |
| Database Toolbox                        | Version 7.0   | (R2016b) |
| Datafeed Toolbox                        | Version 5.4   | (R2016b) |
| Econometrics Toolbox                    | Version 3.5   | (R2016b) |
| Embedded Coder                          | Version 6.11  | (R2016b) |
| Financial Instruments Toolbox           | Version 2.4   | (R2016b) |
| Financial Toolbox                       | Version 5.8   | (R2016b) |
| Fixed-Point Designer                    | Version 5.3   | (R2016b) |
| Global Optimization Toolbox             | Version 3.4.1 | (R2016b) |
| Image Acquisition Toolbox               | Version 5.1   | (R2016b) |
| Image Processing Toolbox                | Version 9.5   | (R2016b) |
| Instrument Control Toolbox              | Version 3.10  | (R2016b) |
| MATLAB Coder                            | Version 3.2   | (R2016b) |
| MATLAB Compiler                         | Version 6.3   | (R2016b) |
| MATLAB Compiler SDK                     | Version 6.3   | (R2016b) |
| Mapping Toolbox                         | Version 4.4   | (R2016b) |
| Model Predictive Control Toolbox        | Version 5.2.1 | (R2016b) |
| Neural Network Toolbox                  | Version 9.1   | (R2016b) |
| Optimization Toolbox                    | Version 7.5   | (R2016b) |
| Parallel Computing Toolbox              | Version 6.9   | (R2016b) |
| Partial Differential Equation Toolbox   | Version 2.3   | (R2016b) |
| RF Toolbox                              | Version 3.1   | (R2016b) |
| Robust Control Toolbox                  | Version 6.2   | (R2016b) |
| Signal Processing Toolbox               | Version 7.3   | (R2016b) |
| SimBiology                              | Version 5.5   | (R2016b) |
| SimEvents                               | Version 5.1   | (R2016b) |
| Simscape                                | Version 4.1   | (R2016b) |
| Simscape Electronics                    | Version 2.10  | (R2016b) |
| Simscape Fluids                         | Version 2.1   | (R2016b) |
| Simscape Multibody                      | Version 4.9   | (R2016b) |
| Simscape Power Systems                  | Version 6.6   | (R2016b) |
| Simulink 3D Animation                   | Version 7.6   | (R2016b) |
| Simulink Coder                          | Version 8.11  | (R2016b) |
| Simulink Control Design                 | Version 4.4   | (R2016b) |
| Simulink Desktop Real-Time              | Version 5.3   | (R2016b) |
| Stateflow                               | Version 8.8   | (R2016b) |
| Statistics and Machine Learning Toolbox | Version 11.0  | (R2016b) |
| Symbolic Math Toolbox                   | Version 7.1   | (R2016b) |
| System Identification Toolbox           | Version 9.5   | (R2016b) |
| Trading Toolbox                         | Version 3.1   | (R2016b) |
| Wavelet Toolbox                         | Version 4.17  | (R2016b) |

# Introduction to the software: MATLAB/SIMULINK

- SIMULINK: the friendly GUI environment
  - SIMULINK (Simulation and Link) is an extension of MATLAB that offers modeling, simulation, and analysis of dynamical systems under a graphical user interface (GUI) environment;
  - SIMULINK is based on block diagrams of Dynamic Systems;
  - The construction of a model is simplified with click-and-drag mouse operations. SIMULINK includes a comprehensive block library of toolboxes (called **blocksets**) for both linear and nonlinear analyses;
  - The programmer can add new SIMULINK blocks to the existing ones, writing them in MATLAB by means of the so-called S-functions. Remember that SIMULINK is a part of MATLAB.

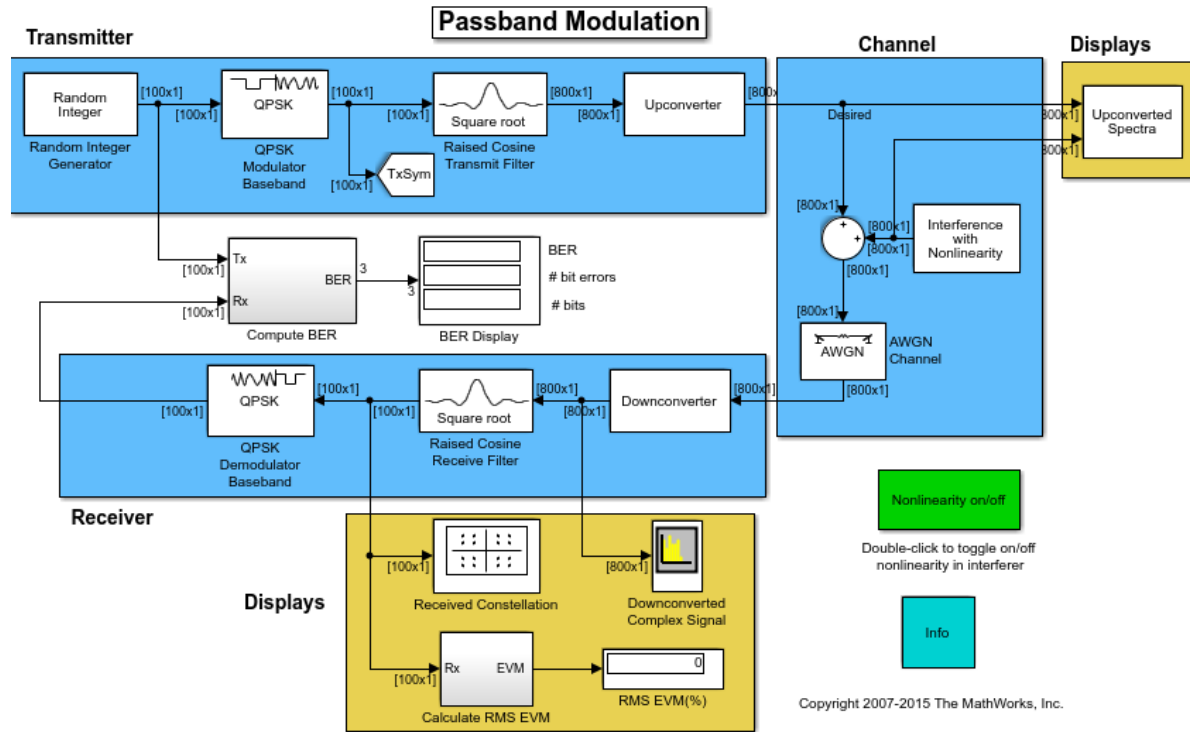
# Introduction to the software: MATLAB/SIMULINK

## ● SIMULINK: the GUI interface



# Introduction to the software: MATLAB/SIMULINK

- SIMULINK: example of implemented communication systems

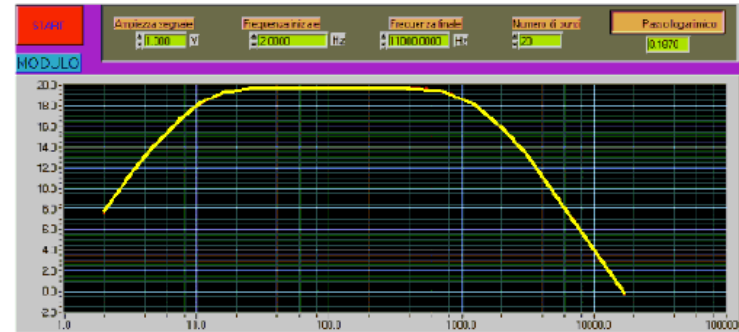
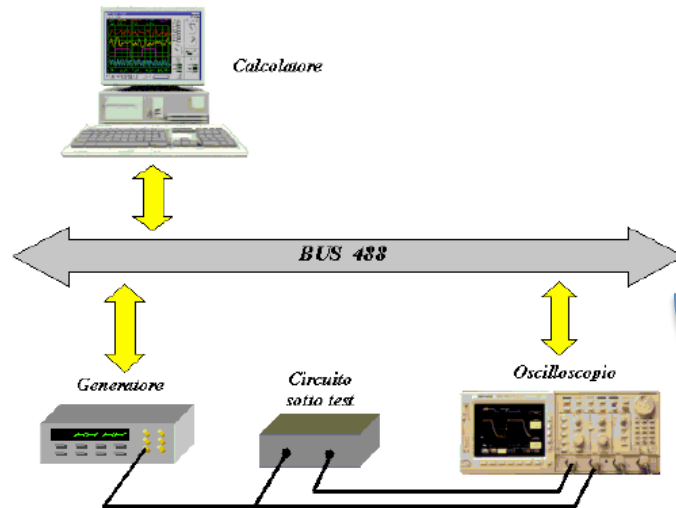


# Introduction to the software: LabVIEW

- LabVIEW: the benchmark for testing real devices
  - LabVIEW (*Laboratory Virtual Instrument Engineering Workbench*) is a software licensed by National Instruments;
  - LabVIEW allows to:
    - Acquire data from physical devices connected to the PC;
    - Directly manage the physical devices;
    - Analyze and process signals in real-time.
  - LabVIEW is characterized by a GUI very similar to that of SIMULINK, allowing to implement complex systems, simply by connecting pre-defined processing blocks.

# Introduction to the software: LabVIEW

- LabVIEW: a graphical example



LabVIEW  
window

To know a bit more: let's see the YOUTUBE tutorial:  
[https://www.youtube.com/watch?v=Em5R\\_RM8E08](https://www.youtube.com/watch?v=Em5R_RM8E08)

# National Instrument USRP transmission boards

- The PHYSICAL devices allowing in-lab SDR implementations
  - USRP is an acronym standing for **Universal Software Radio Peripheral**: they are the available low-cost PHYSICAL devices allowing in-lab testing of SDR technologies;
  - They are, actually, RF transceiver boards, receiving as input and providing as output digital waveforms synthesized via software by a commodity PC;
  - USRP boards are based on Field Programmable Gate Array (FPGA) signal processing devices. They implement the following functionalities:
    - **TX**: D/A conversion, RF up-conversion, physical transmission (with antennas);
    - **RX**: physical detection, RF down-conversion, A/D conversion



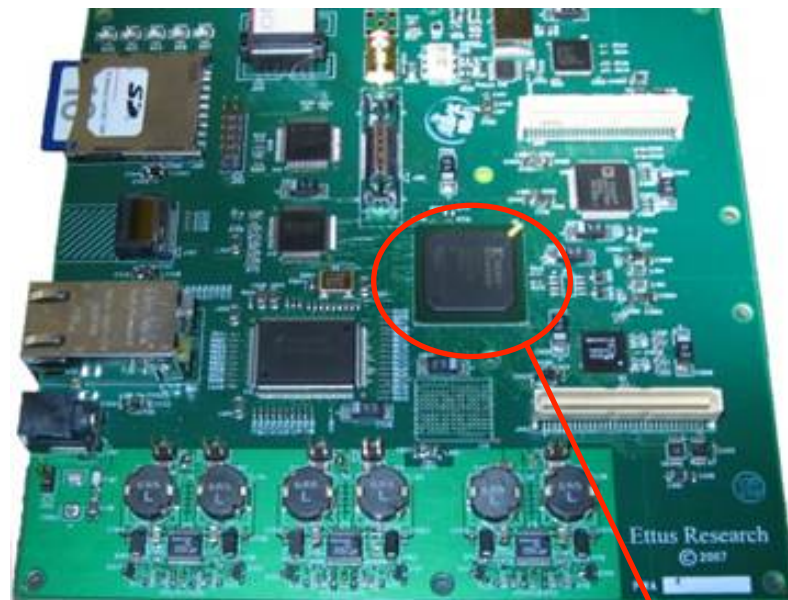
# National Instrument USRP transmission boards

- How is USRP made?



**External view (N210 model)**

*Gigabit  
Ethernet  
interface for PC  
connection*



**Internal view**

*Xilinx  
Spartan 3A  
3400 FPGA*

# National Instrument USRP transmission boards

- Technical features of USRP N210
  - Four 100 MS/s 14-bit Analog Digital Converters;
  - Four 400 MS/s 16-bit Digital Analog Converters;
  - Gigabit Ethernet interface;
  - Host sample rate (8bit/16 bit): 50/25 MS/s;
  - RF front-end: 2.4 GHz and 5 GHz frequencies (ISM band), 15dBm TX EIRP;
  - Capability of MIMO support.

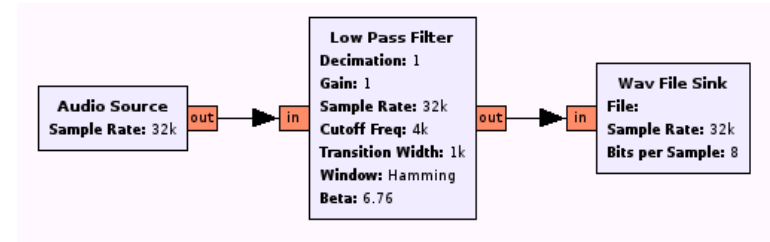
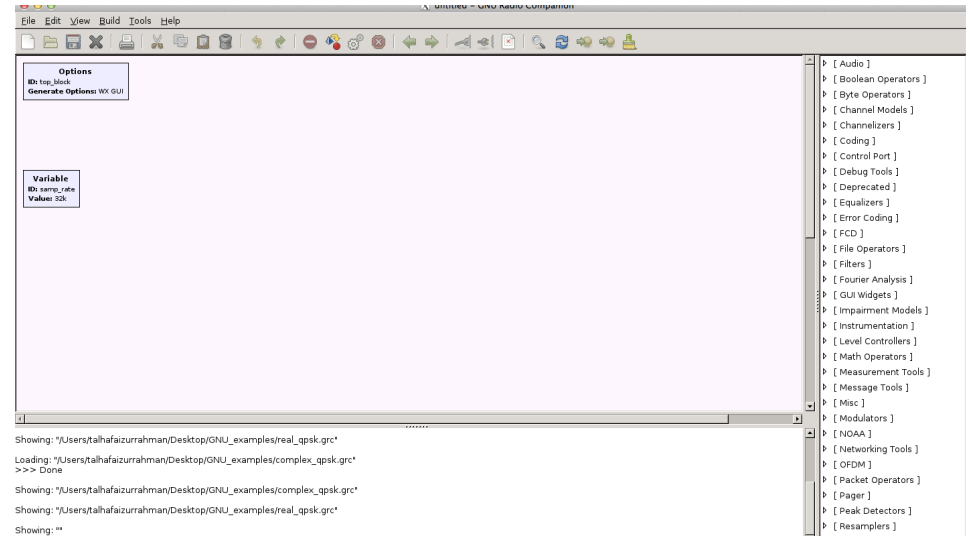
# National Instrument USRP transmission boards

- How can we program them? First solution: GNU-radio
  - For some years, the only software environment used in combination with USRP boards was **GNU-radio**;
  - So far, it is still the main software environment used for academic and in-lab SDR experimentations;
  - GNU-radio is a fully open-source SDR platform originated by a voluntary project. The GNU-radio is a big networked community with a web site, a blog and a lot of social tools to interact (see: <https://www.gnuradio.org>);
  - GNU-radio signal processing procedures are written partially in C++ and, partially, in Python. The Python code is necessary to link the C++ compiled blocks into tree-structured software transceiver architectures. GNU-radio efficiently runs only under **UBUNTU** and **MAC-OS**.



# National Instrument USRP transmission boards

- How can we program them? Second solution: GRC
  - GRC stands for **GNU-Radio Companion**: it is the GUI of GNU-radio (exactly as SIMULINK is the GUI of MATLAB);
  - GRC allows to build graphical flowgraphs of transmitter and receiver (with GNU-radio, we should have built the graph by means of Python: it is less user-friendly);
  - Actually, the components of the system are connected as blocks in a diagram (similarly to SIMULINK and LabVIEW, see figures aside):



# National Instrument USRP transmission boards

- How can we program them? Third solution: MATLAB/SIMULINK
  - Recently, it has made possible to program USRP board directly with MATLAB and SIMULINK also under Windows OS environment;
  - The Communications System Toolbox includes support packages that enable to process wireless signals from/to USRP boards using MATLAB and SIMULINK;
  - Some examples are available (see e.g. <http://it.mathworks.com/help/supportpkg/usrpradio/examples/qpsk-transmitter-with-usrp-r-hardware.html>)

# National Instrument USRP transmission boards

- How can we program them? BEST solution: LabVIEW
  - From one hand, National Instrument is now owner of Ettus and, therefore, of USRP technology;
  - On the other hand, National Instrument licenses LabVIEW software;
  - Therefore, the most efficient way to program the USRP boards is to use LabVIEW software, which is predisposed for the use with USRP;
  - LabVIEW allows to process and visualize in real-time USRP signals with augmented efficiency as compared to GRC and SIMULINK. Indeed, LabVIEW is conceived to manage devices and to test them;
  - Very interesting: similarly to GNU-radio, also NI created its own SDR developer community working with USRP and LabVIEW: <https://forums.ni.com/t5/Software-Defined-Radio/tkb-p/3017>.

# National Instrument USRP transmission boards

- LabVIEW SDR forum (home page screenshot)

