

Efficient implementation of a spectrum scanner on a software-defined radio platform



François Quitin, Riccardo Pace
Université libre de Bruxelles (ULB), Belgium

Context and objectives

Regulators need to detect abusive usage of RF spectrum

- Lots of technicians driving around on all kind of missions
⇒ Use (reasonably) cheap hardware to do opportunistic scanning



- How to use SDRs to do (pseudo)-realtime spectrum scanning?
 - USRP-N210
 - Single-board host computer (e.g. Raspberry Pi form factor)
 - No user intervention

Context and objectives

What's wrong with existing codes ?

- <http://www.ni.com/white-paper/13882/en/>
- « Spectrum analyzer with USRP, GNU Radio and MATLAB »
<http://www.av.it.pt/conftele2009/papers/114.pdf>
- https://github.com/Edgarware/USRP_Spec_Analyzer
- ...

Problems with these implementations?

- Scanning bandwidth may be larger than USRP bandwidth
- Fully software-based (Python, Labview, C++), so hard to do real-time on low-weight host

⇒ Solution: implement the spectrum scanner on the USRP FPGA !

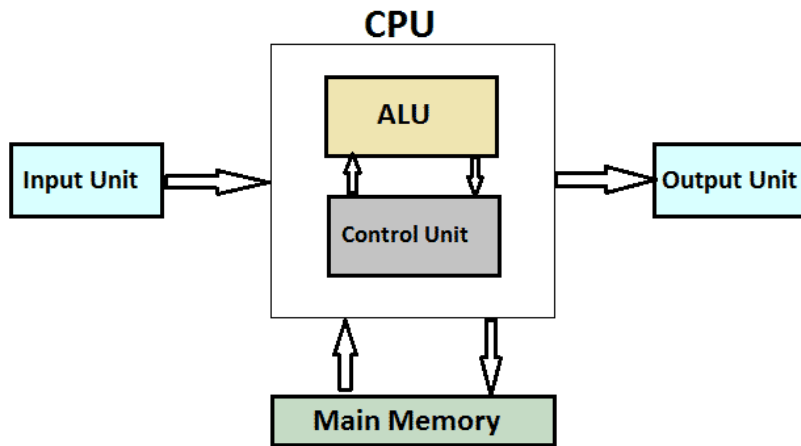
- Use software for scan coordination and data recording

Outline

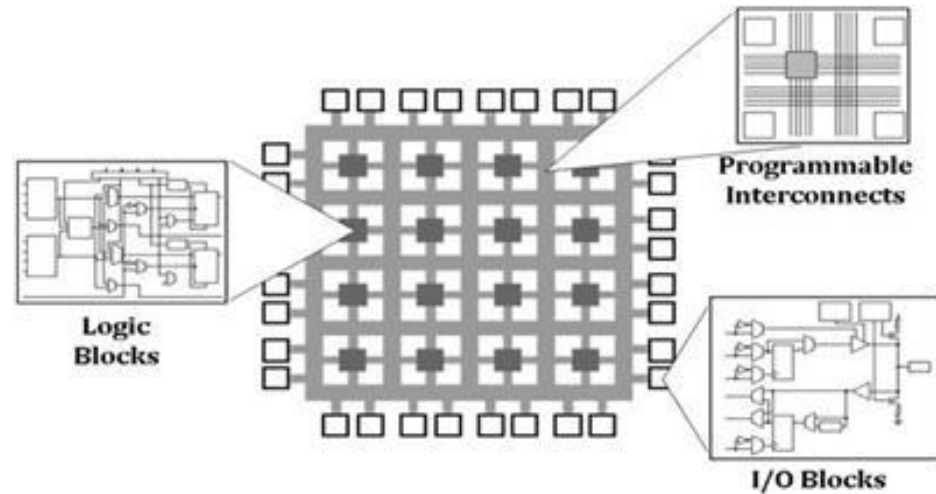
- Overall system design
 - Intro to FPGA: difference with μC
 - Spectrum scanner design
- FPGA design
 - FFT module
 - Square magnitude module
 - Energy detection module
 - Data synchronizer module
- Software design
 - Usefull low-level UHD commands
 - Retuning and streaming
 - GUI with gnuplot-iostream
- Some results
- Demo time!

What is an FPGA ...

... and why can it be faster than a microcontroller?



- Executes 1 instruction / cycle
- Serial processing
- Compilation: converts HLL to micro-instructions
- Limited by duration of execution
- HLL: C++, Python, ...

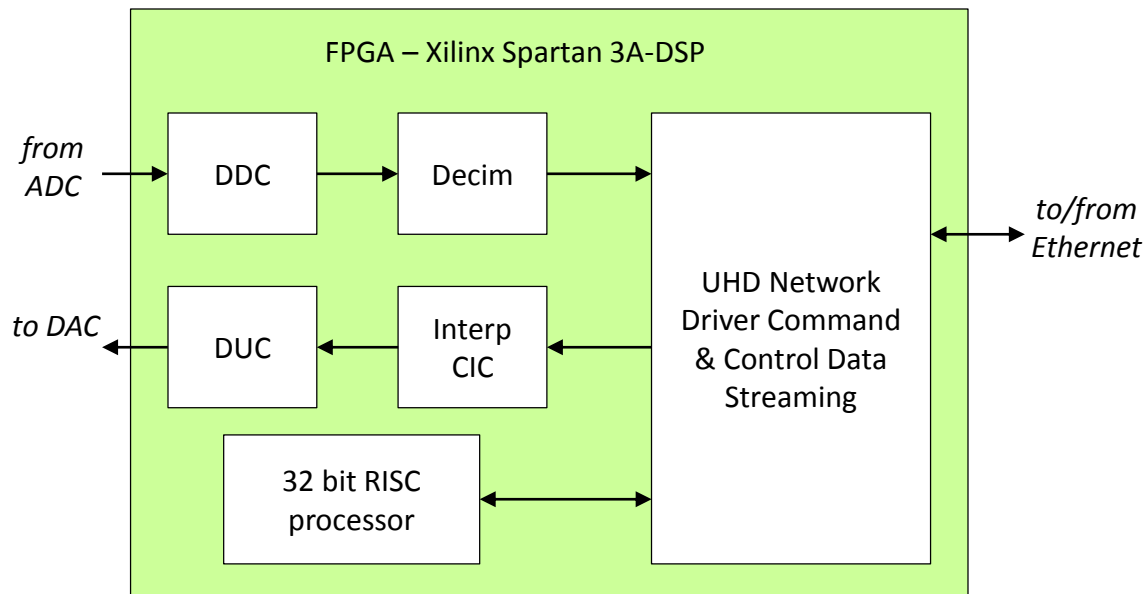


- Processes N instructions / cycle
- Parallel processing
- Synthesis and P&R: converts « HLL » to inter-connection diagram
- Limited by size of circuitry
- « HLL »: VHDL or Verilog

The FPGA in the USRP

Does some basic stuff, but still some space left !

- Digital up- and down conversion
- Decimation/interpolation
- Formating of samples for UHD drivers



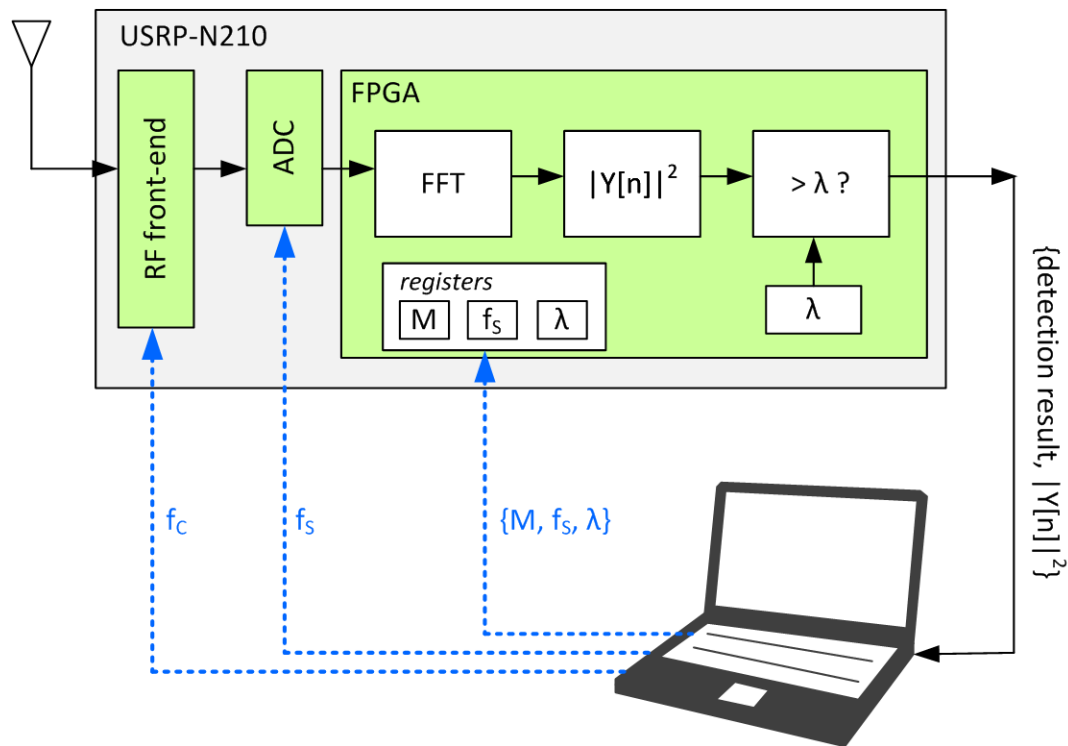
Hardware elements available in the FPGA

	Default FPGA image
Flip Flops	42%
4-input LUT	65%
Slices	82%
DSP48A	24%
RAM16BWER	32%

Spectrum scanner design

Mixed FPGA-software architecture

- FPGA to perform CPU-intensive task
- Software to coordinate retuning of carrier frequency and log data



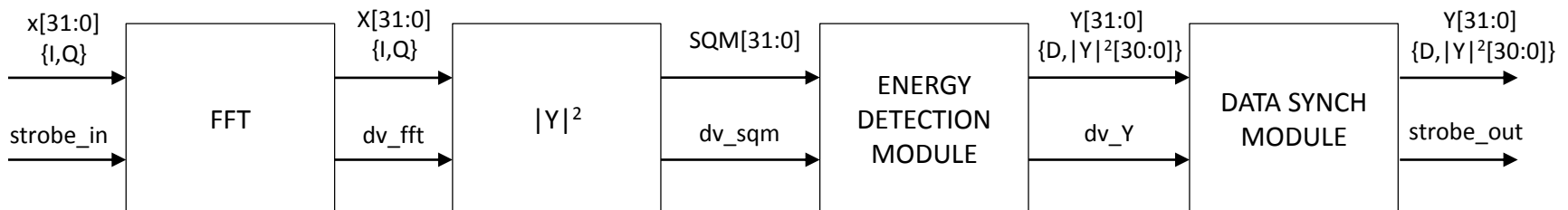
Outline

- Overall system design
 - Intro to FPGA: difference with μC
 - Spectrum scanner design
- **FPGA design**
 - **FFT module**
 - **Square magnitude module**
 - **Energy detection module**
 - **Data synchronizer module**
- Software design
 - Usefull low-level UHD commands
 - Retuning and streaming
 - GUI with gnuplot-iostream
- Some results
- Demo time!

FPGA design

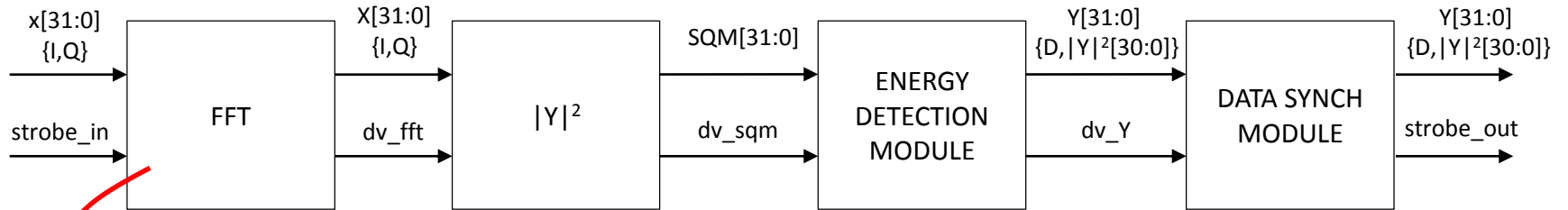
Detailed FPGA architecture

- Modules that can be cascaded
- 2 inputs and outputs for each module
 - One for the actual data
 - One that indicates if data is valid
- Two versions of the Energy Detection Module
 - Fixed threshold (set manually from software)
 - Automatic threshold (mostly automatic, partial manual setting possible)



FPGA design

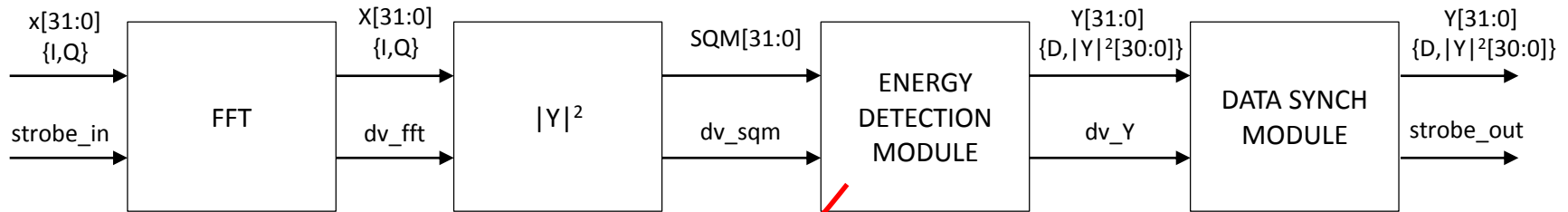
Details of the different modules



- FFT module
 - 1024-point FFT (not configurable)

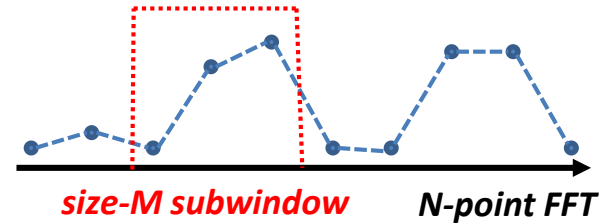
FPGA design

Details of the different modules



- Energy detection module

$$\sum_{i=1}^M |Y|^2 > \lambda ?$$



- **Fixed threshold:** set manually by user from software
- **Automatic threshold:** $\lambda = \lambda^* + \alpha D_{window}$

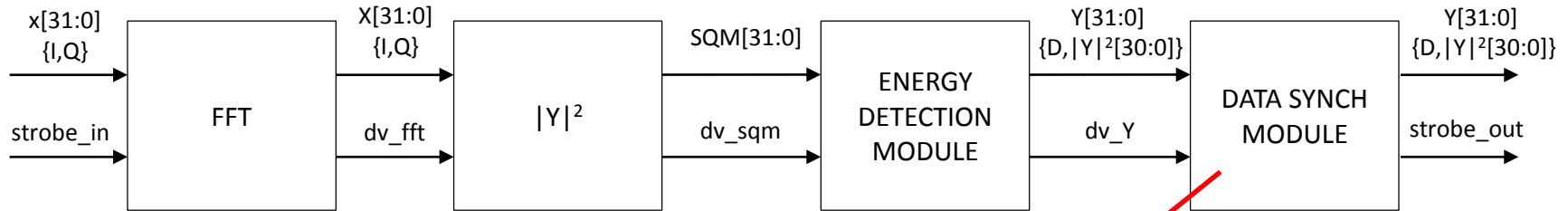
with $D_{window} = \frac{\sum_{k=n}^{n+M} |Y[k]|^2}{\sum_{k=1}^N |Y[k]|^2 / N}$

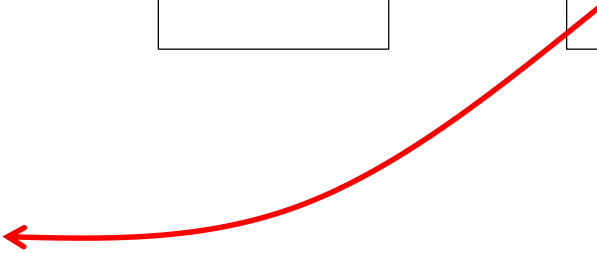
Energy over current subwindow

Average energy over 10MHz-window

FPGA design

Details of the different modules



- Data synchronizer module 
 - Readapt the rate of samples to USRP sample rate
 - Ratio between USRP sample rate and f_{clock} (=100 MHz) has to be an integer value
 - Design fully compatible with host UHD drivers

FPGA design

Resource utilization of our additional blocks

- Our design is very cheap in resource utilization !

	Default FPGA image	Fixed threshold	Automatic threshold
Flip Flops	42%	+3%	+4%
4-input LUT	65%	+3%	+4%
Slices	82%	+2%	+4%
DSP48A	24%	+7%	+9%
RAM16BWER	32%	+12%	+14%

Outline

- Overall system design
 - Intro to FPGA: difference with μC
 - Spectrum scanner design
- FPGA design
 - FFT module
 - Square magnitude module
 - Energy detection module
 - Data synchronizer module
- Software design
 - Usefull low-level UHD commands
 - Retuning and streaming
 - GUI with gnuplot-iostream
- Some results
- Demo time!

Software design

Some useful low-level UHD commands

- Set FPGA register from host side (threshold, subwindow size, ...)

```
// set threshold of the energy detector module  
usrp->set_user_register(TH_ADDRESS,threshold,0);
```

- Specify time of command (retune of carrier frequency)


```
// set the command in time  
usrp->set_command_time(cmpd_time[i]);  
t_result[i] = usrp->set_rx_freq(tune_request[i]);
```

Software design: re-tuning and streaming

Send future retune commands while streaming

```
...
for k=0 to k=7 do
    set command time @ cmd_time[k]
    tune_request[k]
    rx stream command @ cmd_time[k]+delta
end for
while (1) do
    receive samples
    k++
    set command time @ cmd_time[k]
    tune_request[k]
    rx stream command @ cmd_time[k]+delta
end while
```

Retune lock
time,
typically 1 ms

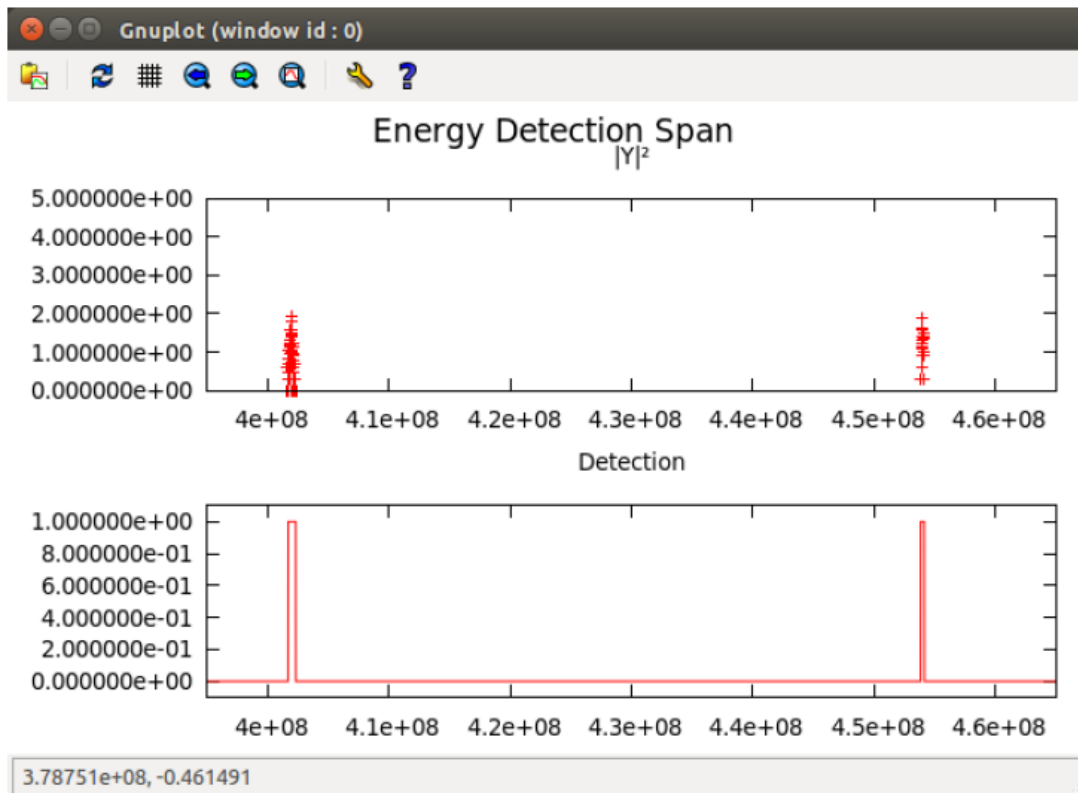


- With this laptop: 250 ms for scanning 1 GHz band w/o overflows

Software design: light-weight GUI

Using gnuplot-iostream

- Low refresh rate to avoid hogging CPU
- Data is also saved to a log file



SQM.dat



detection.dat

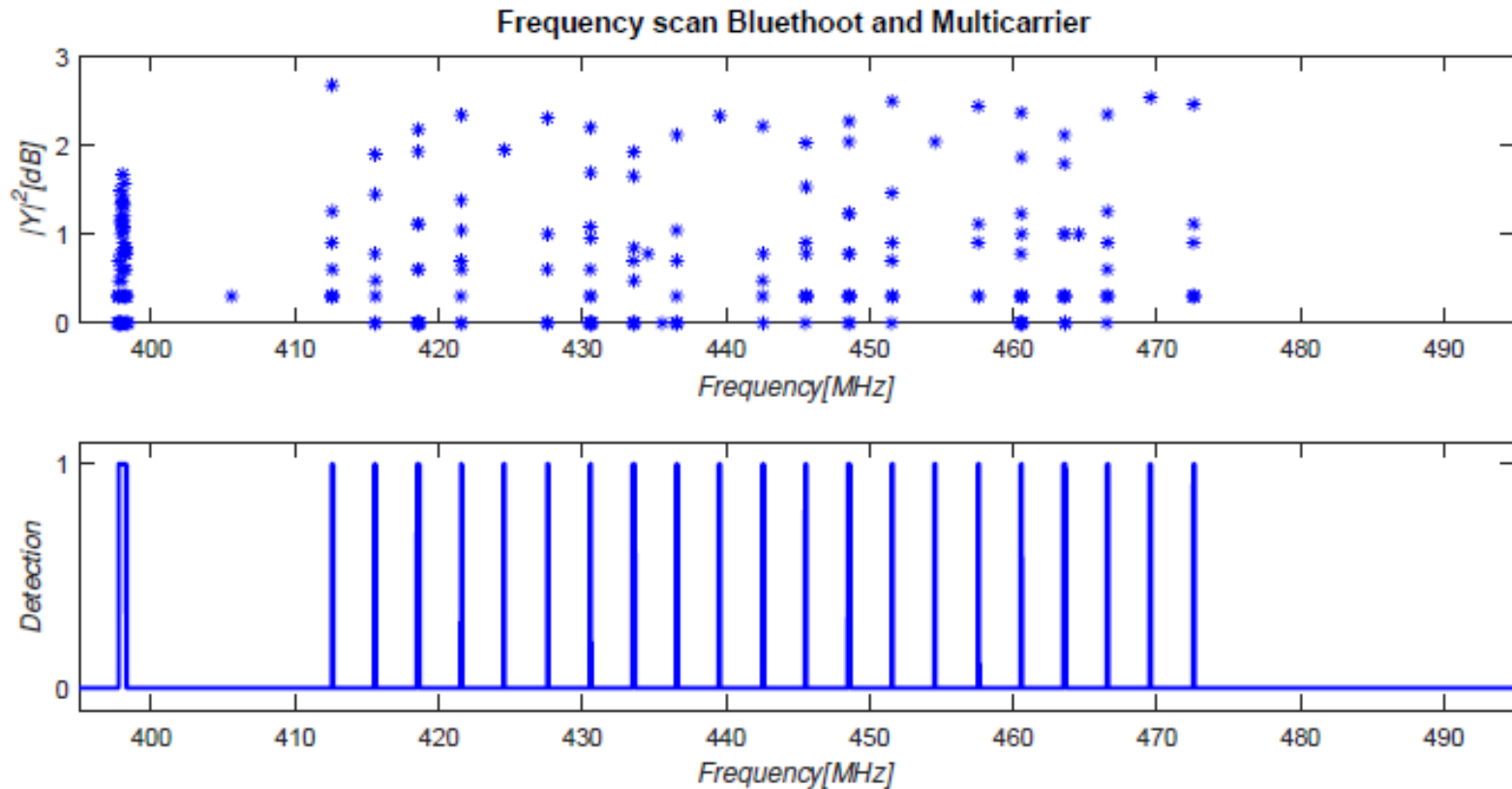
Outline

- Overall system design
 - Intro to FPGA: difference with μC
 - Spectrum scanner design
- FPGA design
 - FFT module
 - Square magnitude module
 - Energy detection module
 - Data synchronizer module
- Software design
 - Usefull low-level UHD commands
 - Retuning and streaming
 - GUI with gnuplot-iostream
- Some results
- Demo time!

Some results in the lab

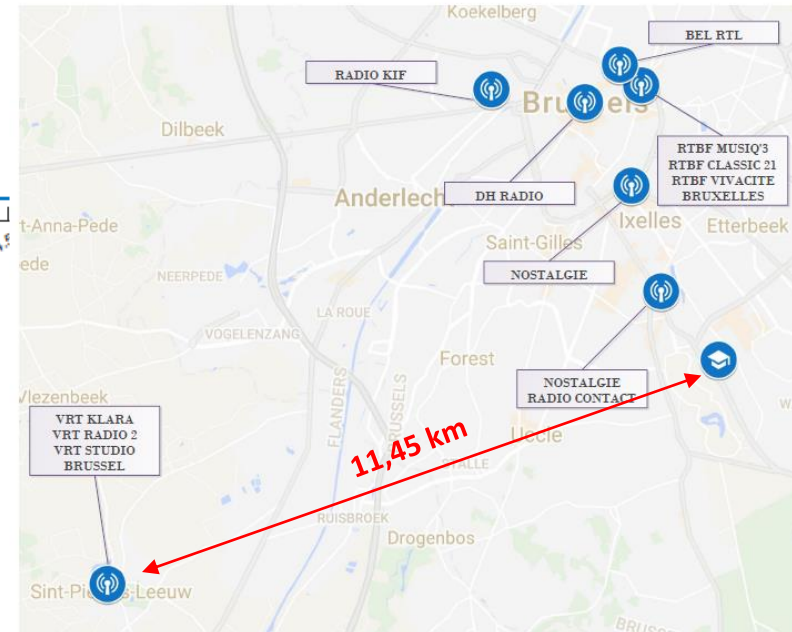
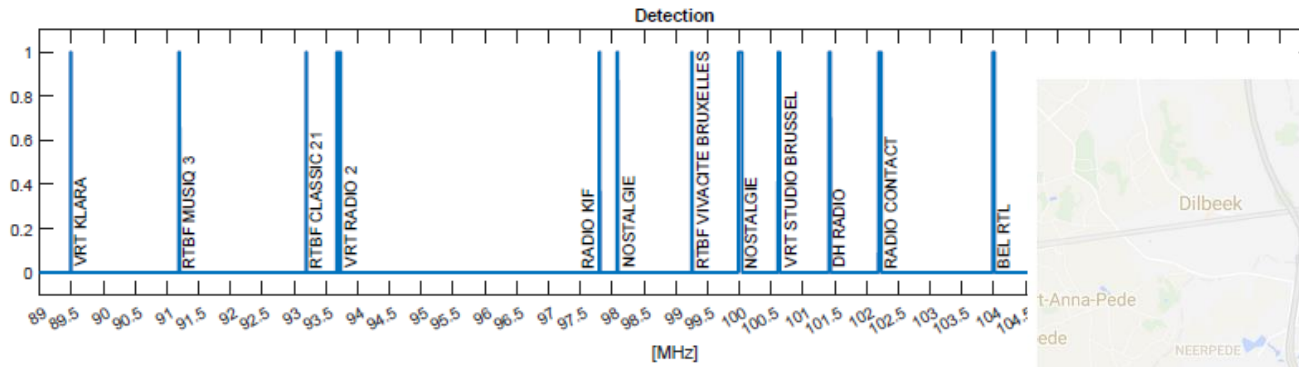
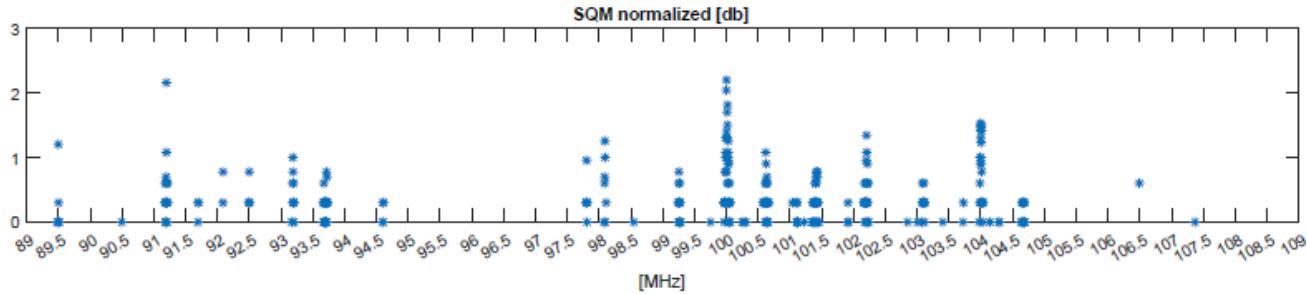
when connecting signal generator to USRP

- Bluetooth + Multicarrier signal



Some results outside the lab

Scanning for FM stations



Some results right here

Demo time !

- Demo of GSM and 3G spectrum scan
- Code available on Github
https://github.com/fquitin/energy_detection_system
- What's in the code?
 - FPGA source code
 - FPGA images, flashable on the USRP
 - Host C++ source code and CMake files
 - Some Matlab scripts with testbenches and postprocessing scripts