



**GENERALITAT
VALENCIANA**
ConSELLERIA de EducacióN,
Universidades y Empleo

Summary of 2023 Activities



Ciemat

Centro de Investigaciones
Energéticas, Medioambientales
y Tecnológicas

AITANA

Abraham Menéndez

IFIC - Universitat de València

01/02/2024

Outline

- *Introduction*
- *Block diagram of a LLRF system*
- *Block Description + Updates*
- *Engineering tasks performed*
- *Conclusions*
- *Next Steps*

Introduction

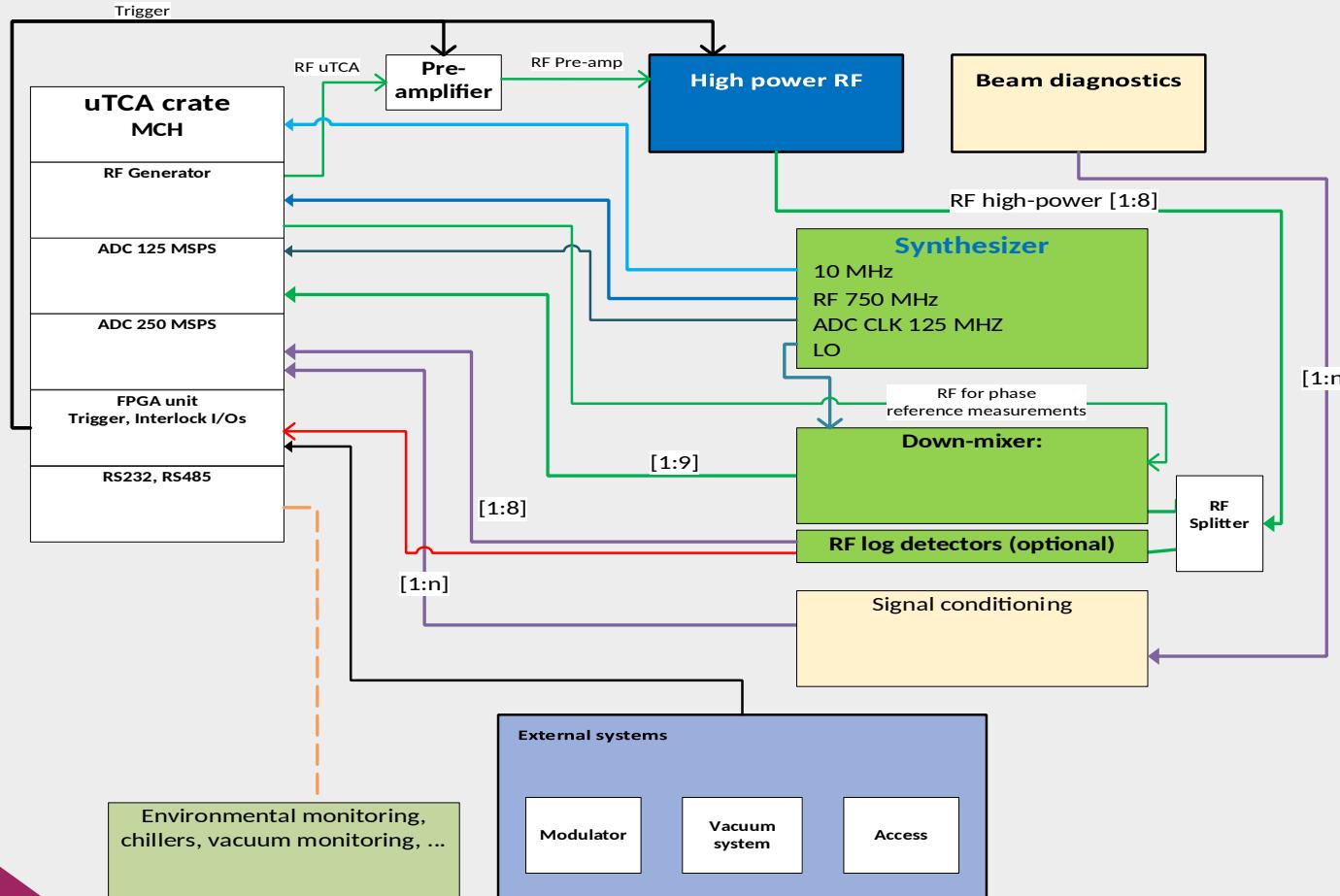
The main work that We have been doing in the last year has been:

Continue developing and updating a **LLRF Control & Acquisition system based on uTCA**, for testing of High-Gradient Acceleration Cavities.

This main work was divided into:

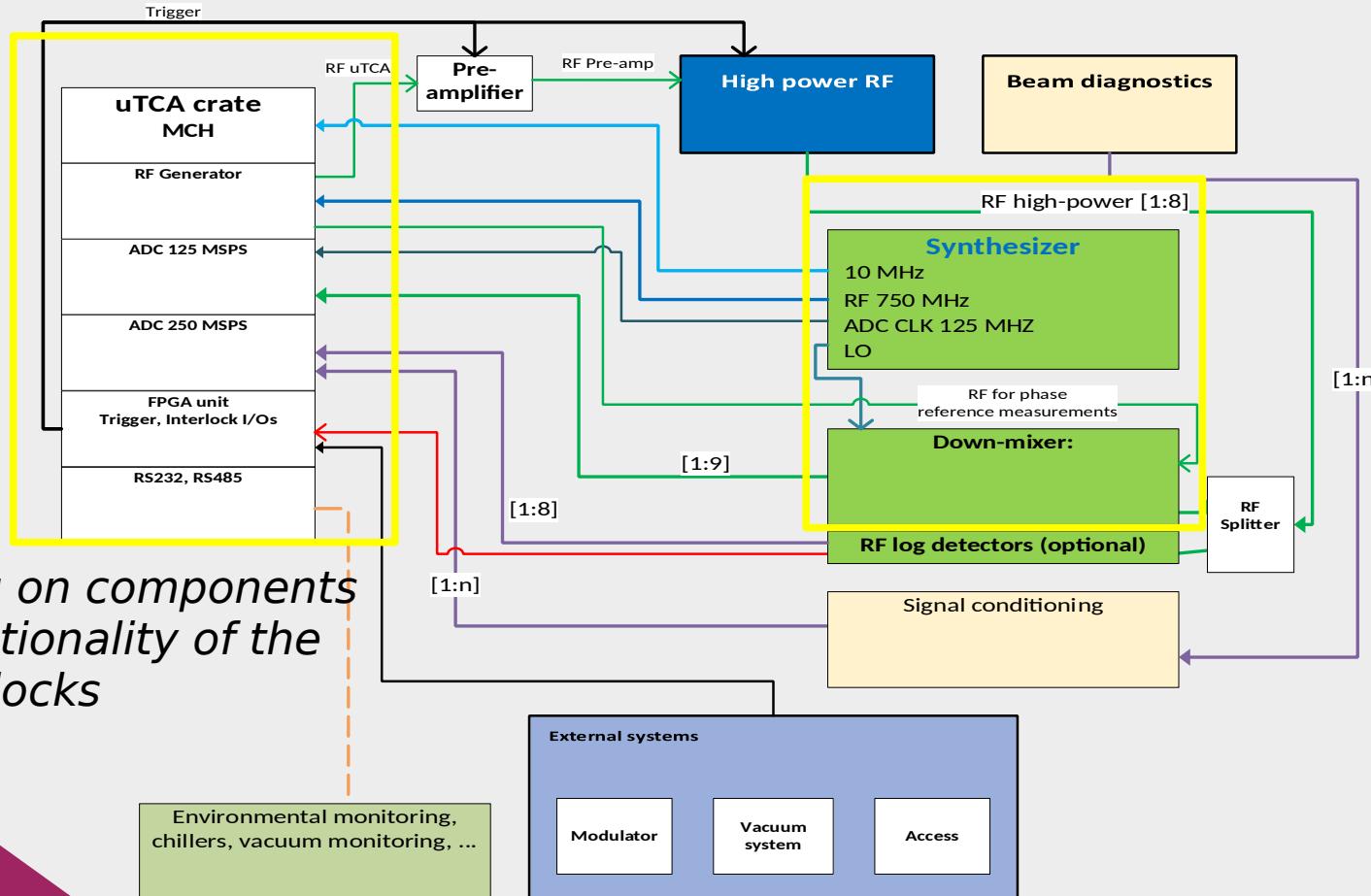
- 1) the acquisition, assembly and commissioning of several electronic modules to expand the current system
- 2) and also, the development of several Software and Hardware engineering tasks to control the system.

Block diagram of a LLRF system



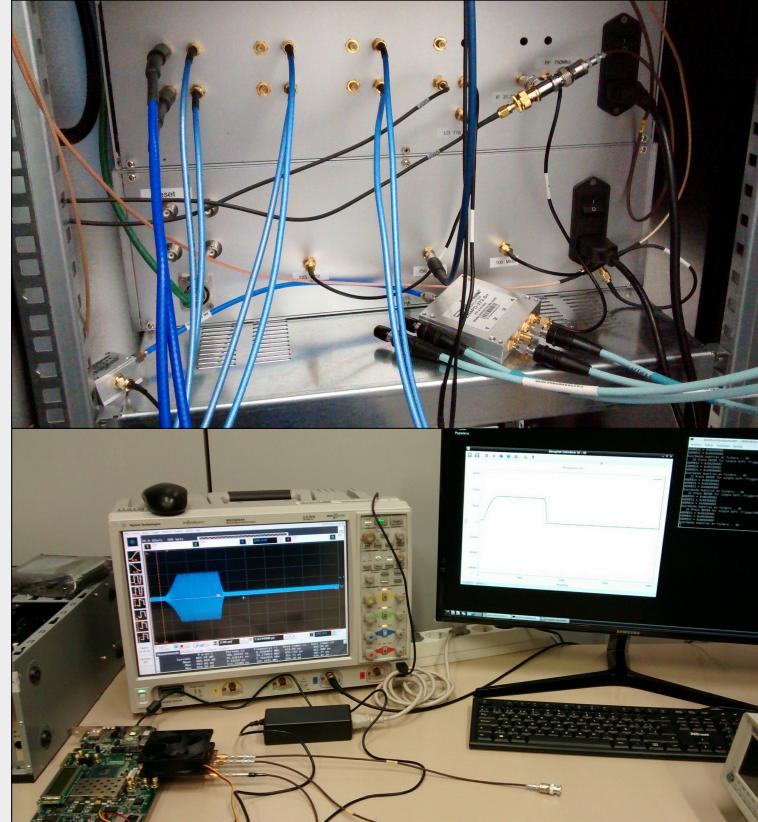
Block diagram of a LLRF system

Since last year, We have been working to upgrade our current system:



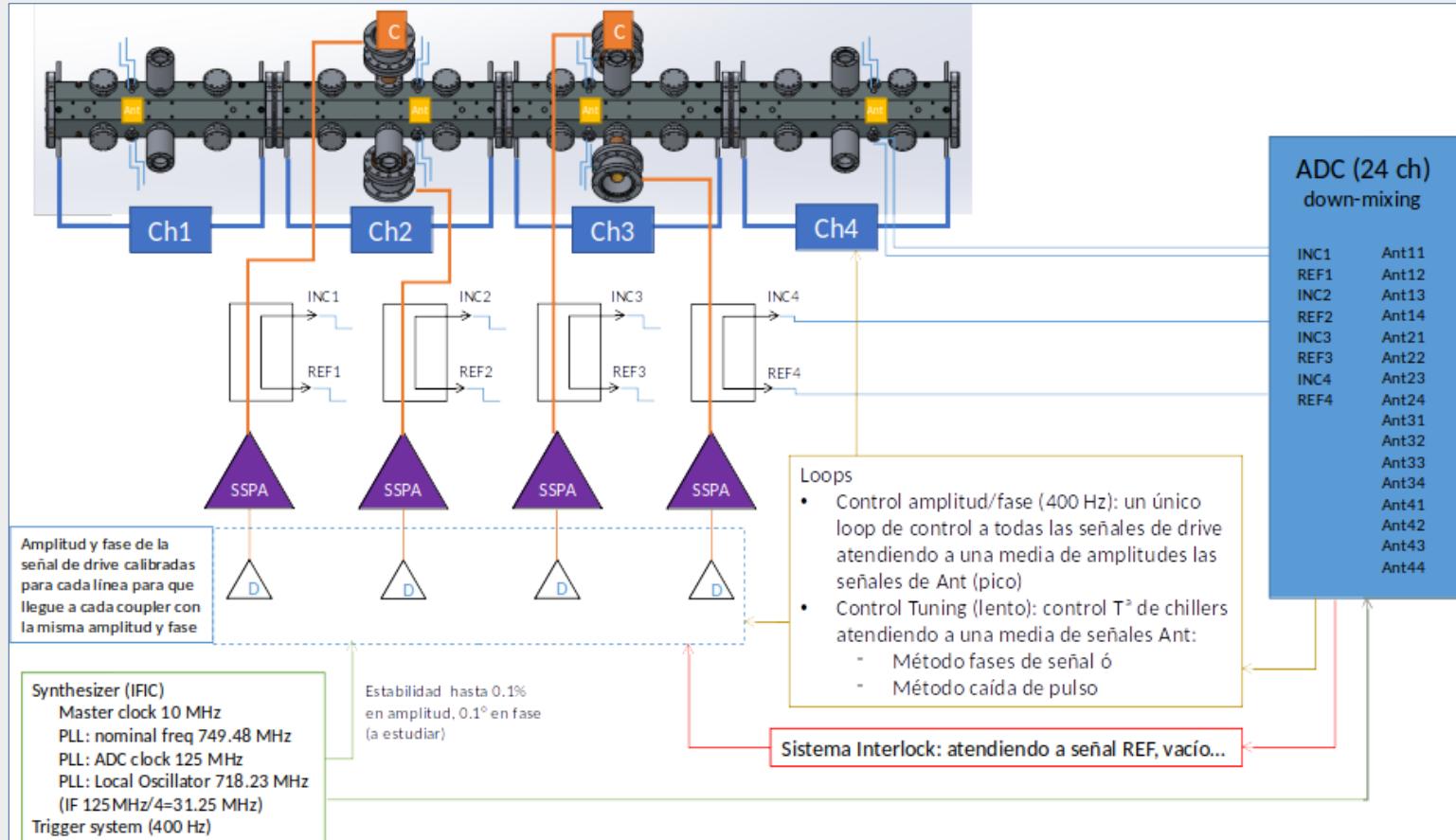
Block diagram of a LLRF system

Some pictures of our initial hardware deployment
(still in progress), located in the RF Lab at IFIC.



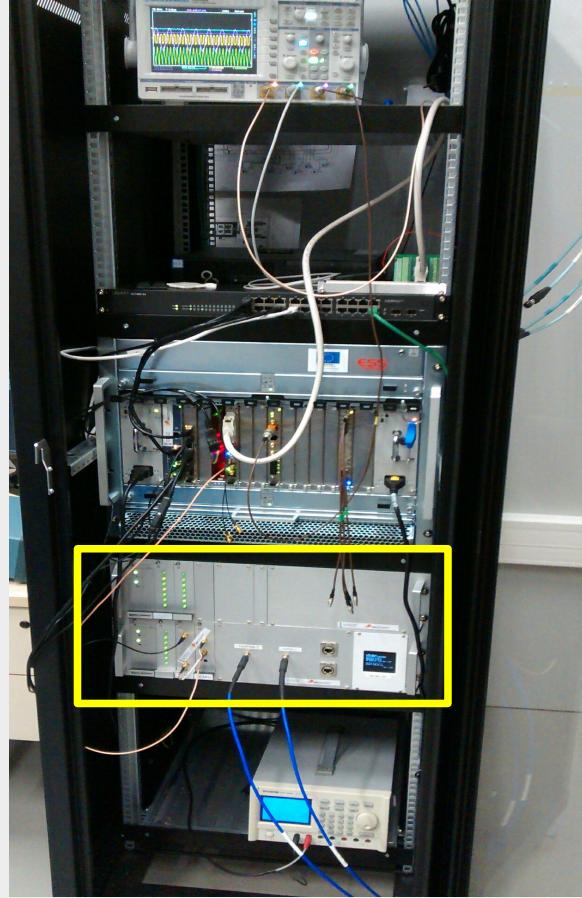
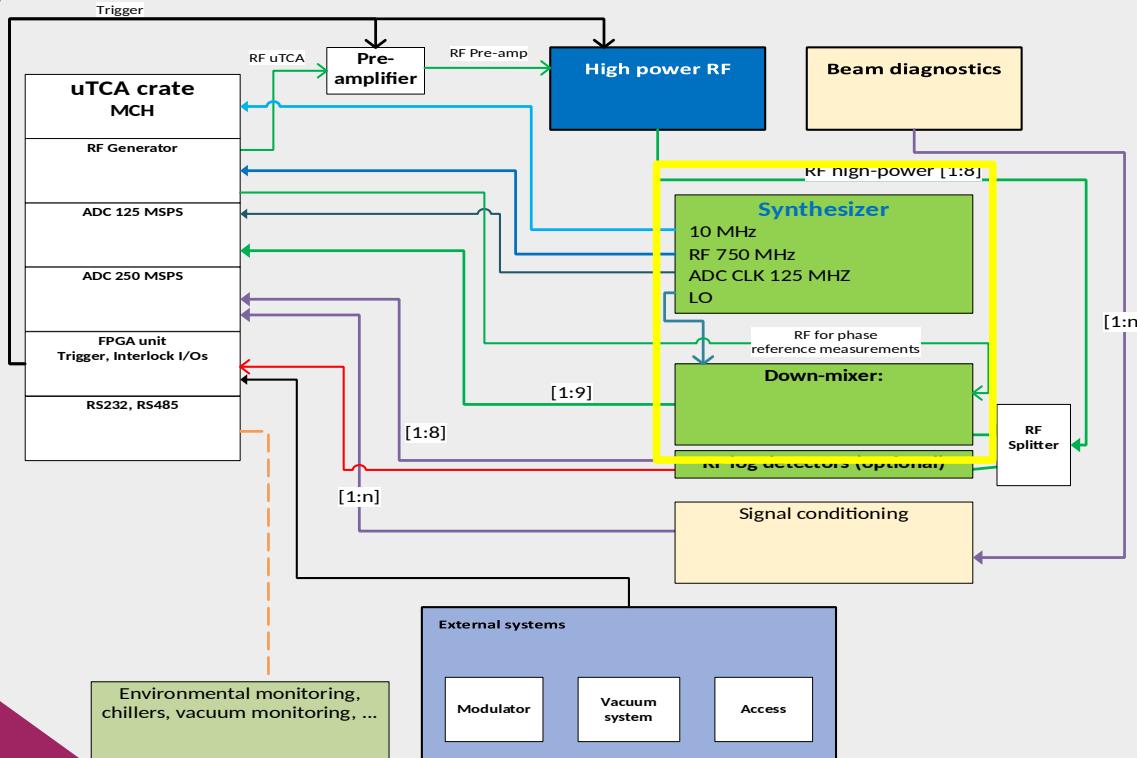
Block diagram of a LLRF system

Desired target configuration:



Block Description:

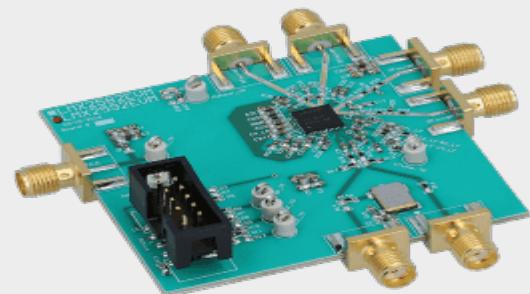
Signal generation (Synthesizer, DownMixer)



Block Description:

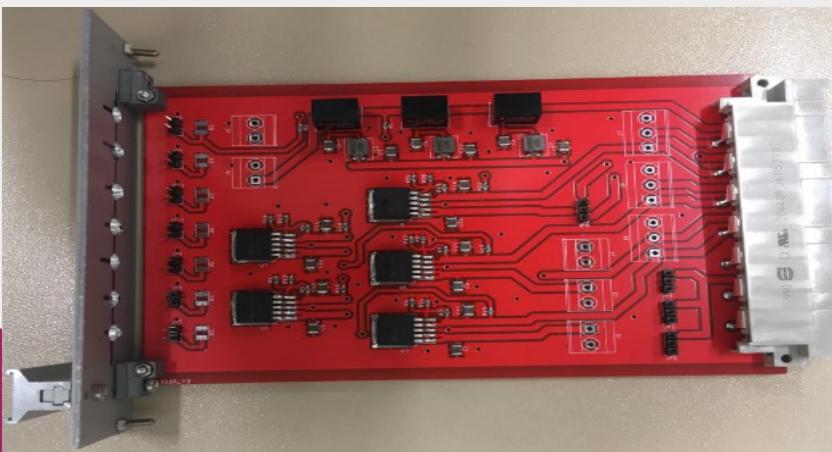
Signal generation (Synthesizer Updates)

- The Synthesizer developed and updated allow us:
 - to generate all the high-frequency signals for a correct system operation, using for it an Ultra Low Jitter oscillator with some programmable Phase-Locked Loop hardware (PLL).
 - The most important feature is the low noise generated in the main clock signal.
 - Every internal registers are configurable, to achieve the desired frequencies mentioned above, with the new software updates.



Block Description:

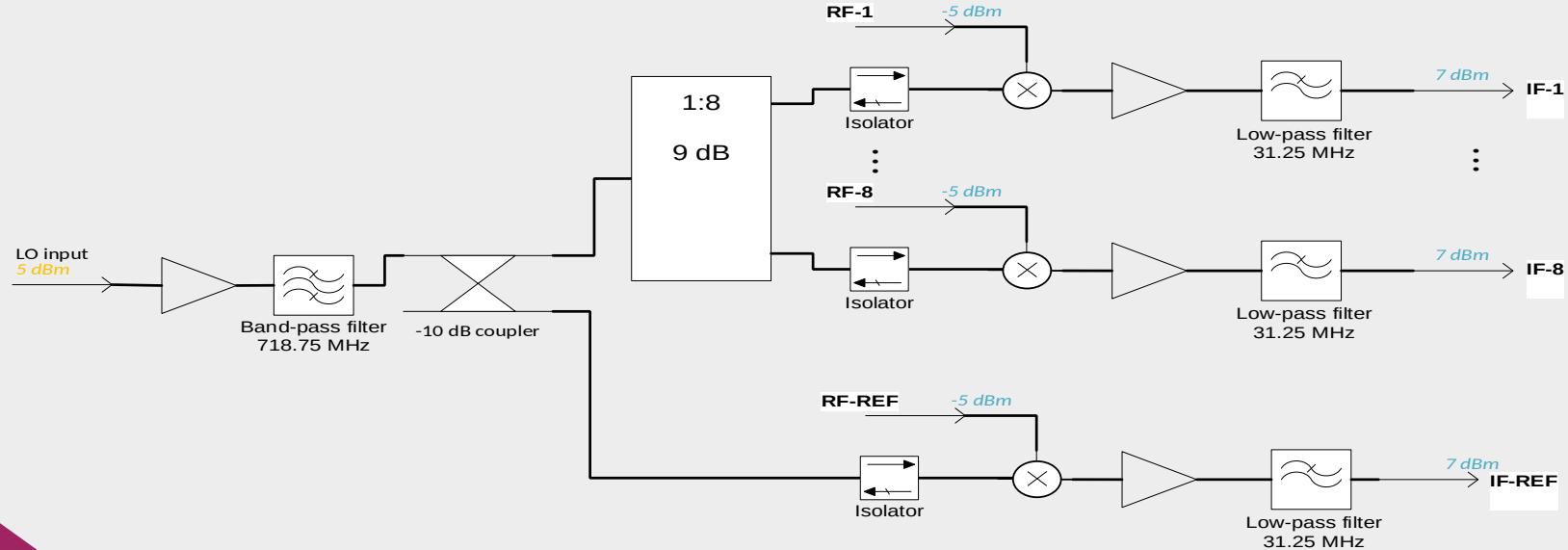
Signal generation (Synthesizer Updates)



Block Description:

Signal generation (DownMixer Updates)

- With the DownMixer built:
 - We can mix, amplify and filter RF input signals to setup the system (Up to 8 channels).



Block Description:

Signal generation (DownMixer Updates)

- Now We are in construction of 2 new DownMixer equipments.
- These will allow us to support for 16 new input channels (RF Signals).

Assembly sample:



Block Description:

Signal generation (DownMixer Updates)

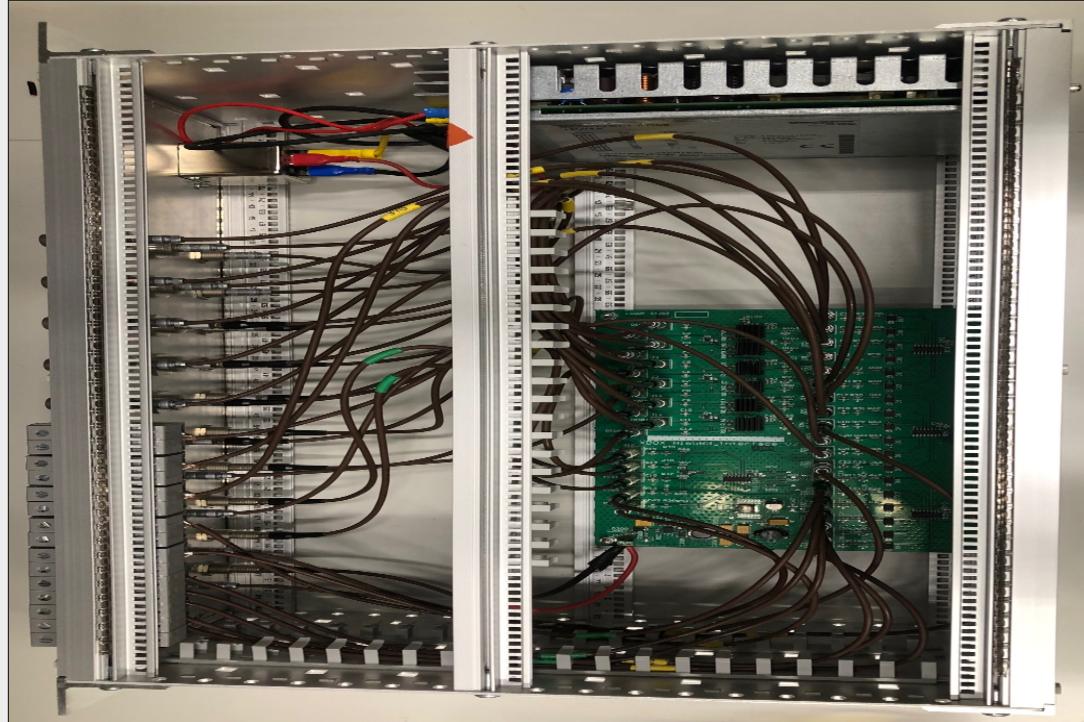
- All these components will be included in a new DownMixer crates:



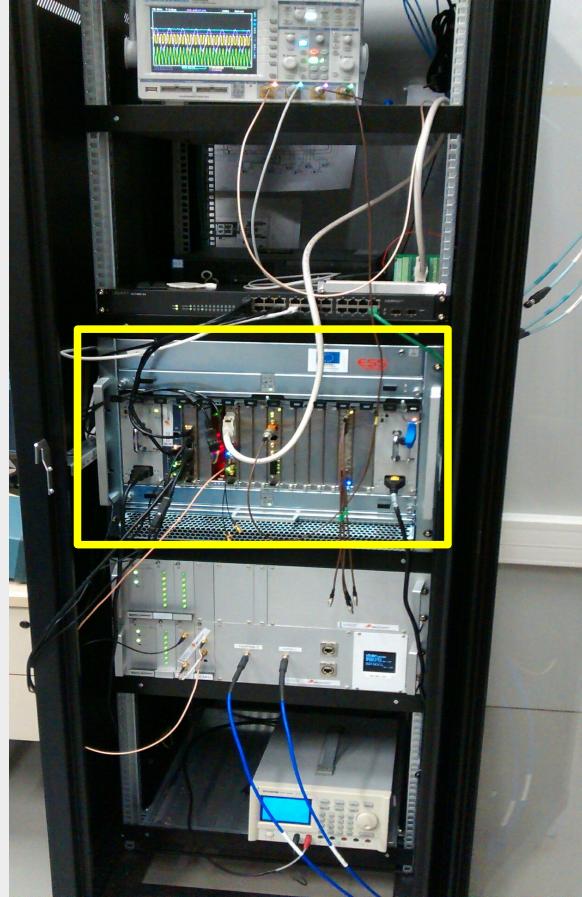
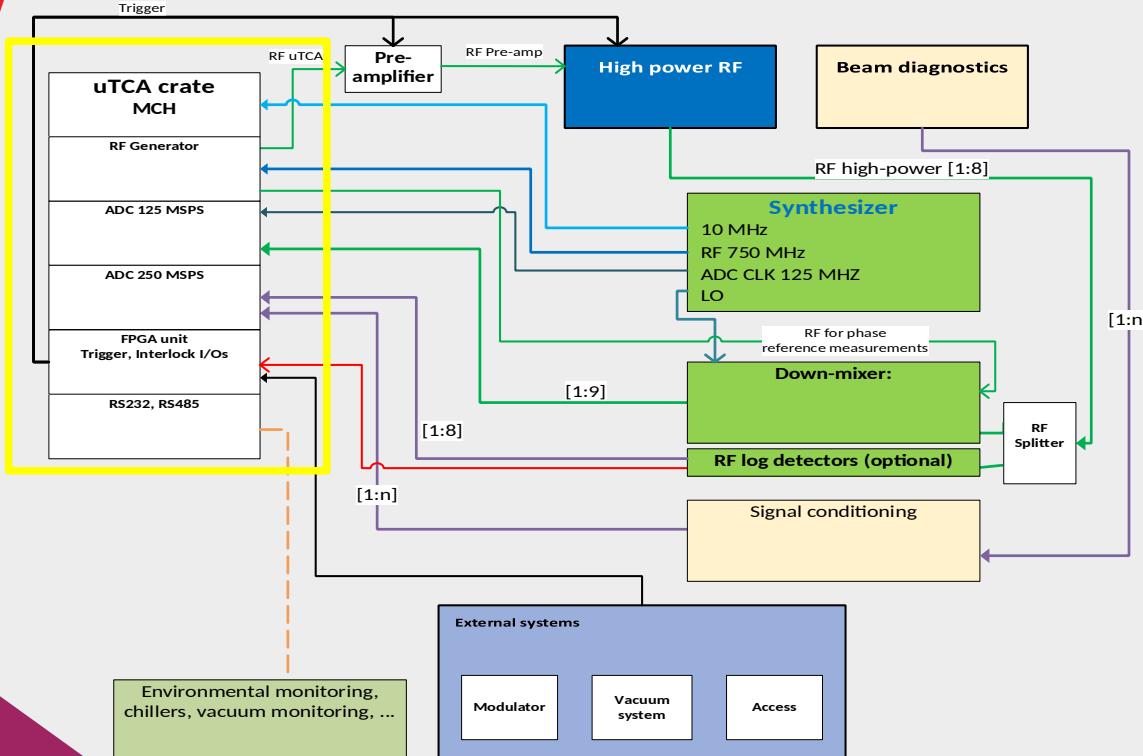
Block Description:

Signal generation (Signal conditioning Upgrade)

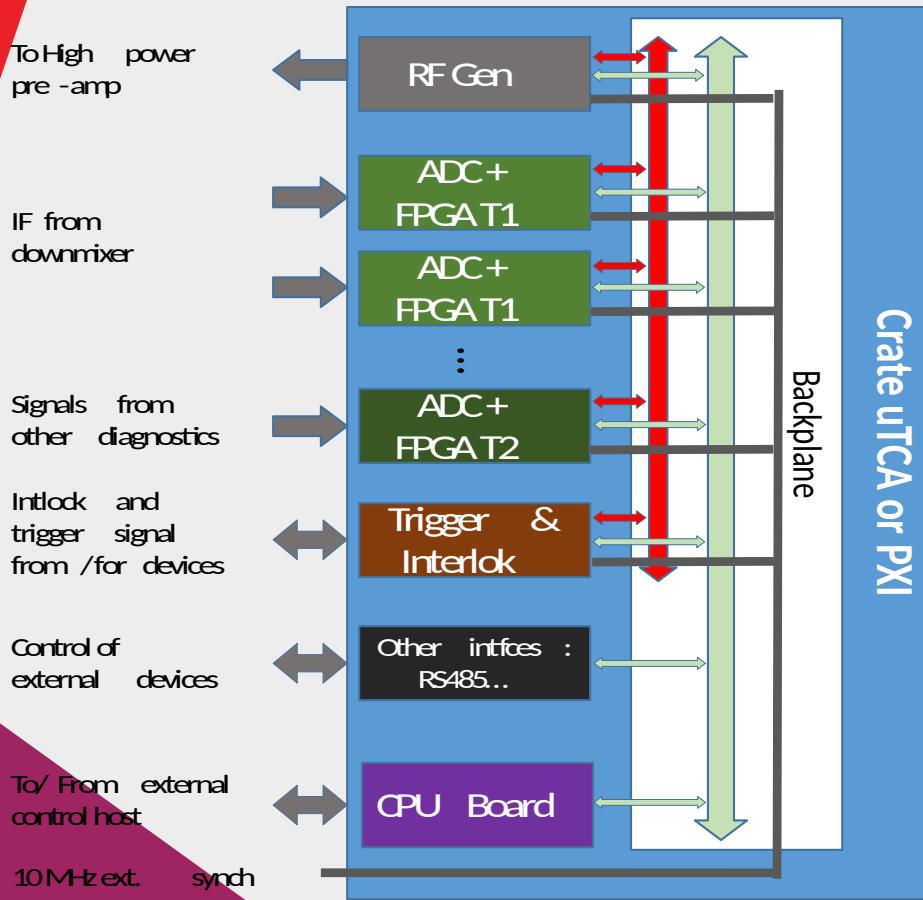
- We have also integrated a new block for conditioning external signals (Power Level Shifters):



Block Description: uTCA system



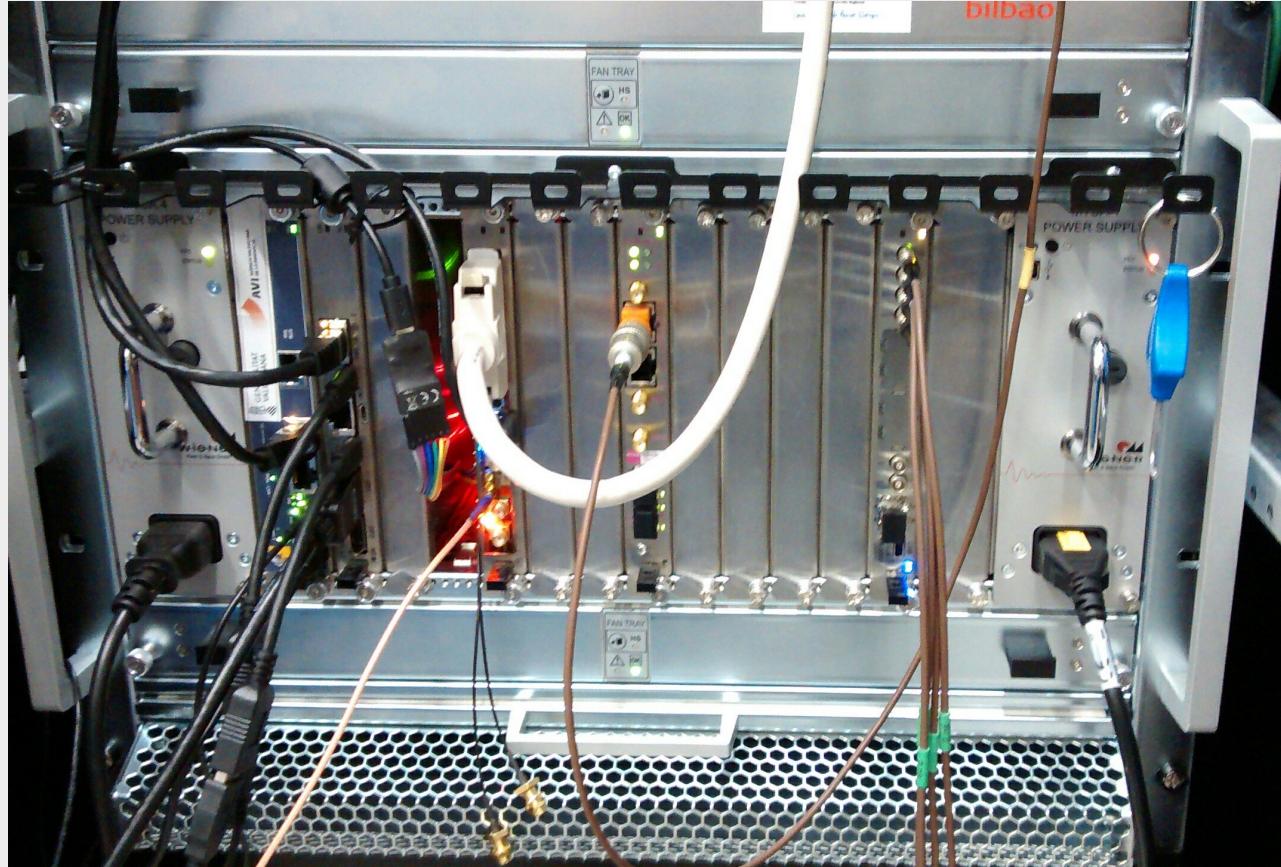
Block Description: uTCA system



- uTCA means:
(Micro Telecommunications Computing Architecture).
- It is a modular system, composed by hardware cards (AMC) connected to a backplane to share data and signals.
- Each card has a specific functionality.

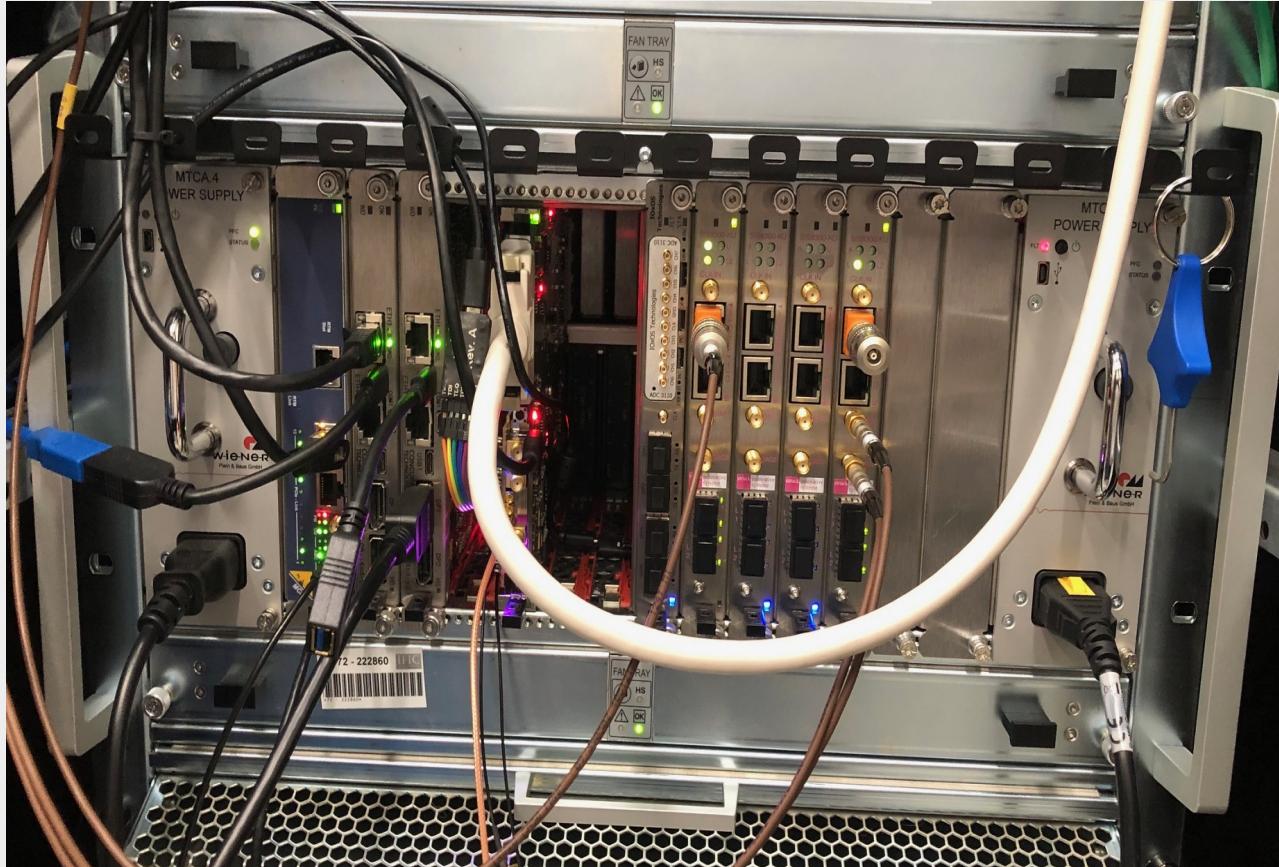
Block Description:

uTCA system. Initial Crate (2021-2022)



Block Description:

uTCA system. Current Crate (2023)



Block Description:

uTCA system. (Updated AMC cards)

- We have added and configured to our uTCA system:
 - 3 new AMC cards (with RF Signal capture + Analog to Digital converters + FPGA for signal processing).
 - 3 new RTM boards (for RF pulse generation).

This set allow us to have 26 more input channels (RF Signals).

2 new Vector Modulators for more pulse generation.

- 1 Zynq UltraScale+ FMC Carrier AMC
For critical supervisory tasks of the system (Interlock, Triggers, I/O...)

Engineering tasks

Development of IP Cores for FPGAs

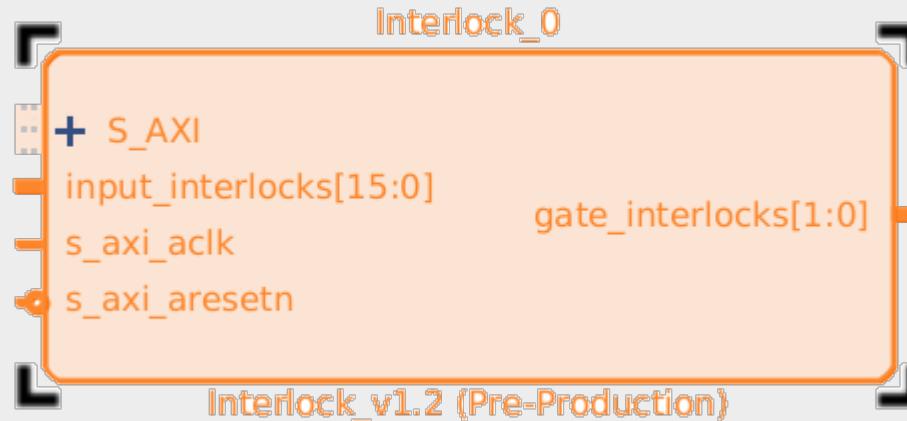
- For an optimal LLRF system response, it has been necessary to develop hardware components (IP Cores) that allow us to control various important functionalities.
- They have been developed for the FMC Carrier AMC designed system (into FPGA), with Verilog HDL language.
- Some critical hardware RT components are:
 - Configurable Trigger generators



Engineering tasks

Development of IP Cores for FPGAs

- Monitor and control of the configurable Interlocks system.



- Special Latches and Interrupts: Fundamental elements to detect fast input values (digital signals).

Engineering tasks

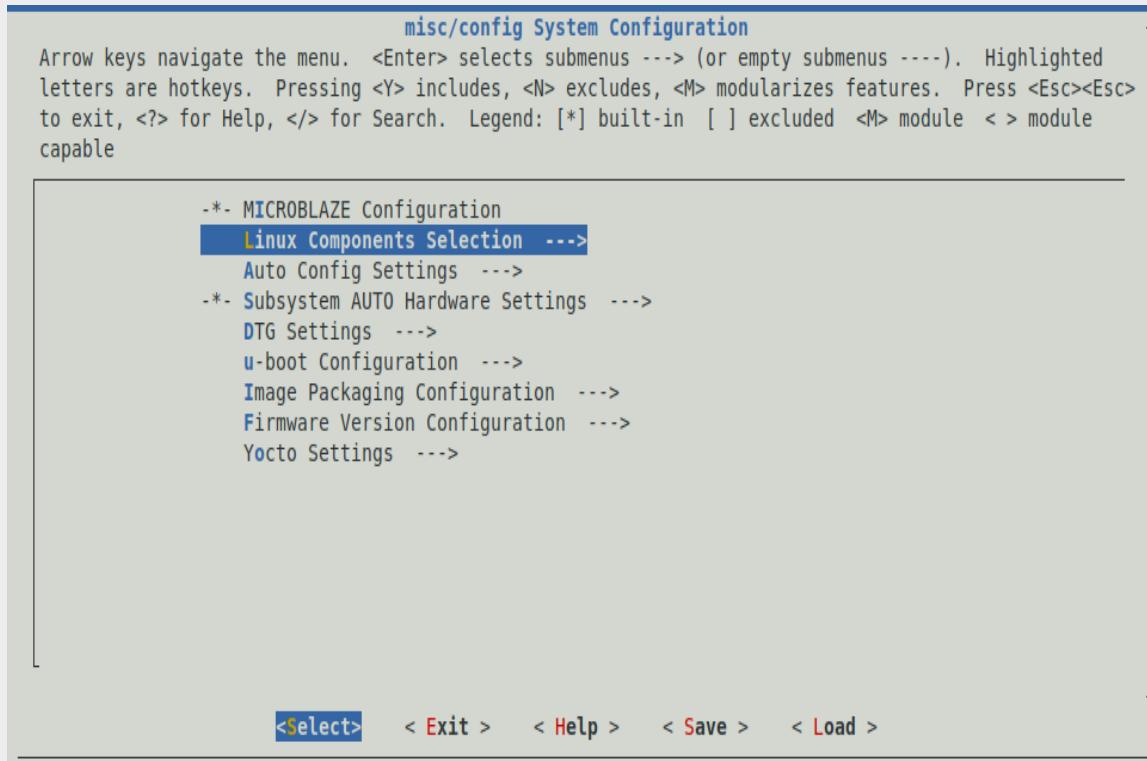
Implementation of Real Time OS

- We have also implemented a Linux-based RTOS, to run critical control tasks from FPGA and MPSOC processing units.
- We have used PetaLinux toolchain for this implementation which has allowed us to:
 - Customize a Kernel and system Drivers.
 - Critical task planner.
 - Interrupt manager.
 - Peripheral memory access.
 - Install python interpreter.
 - Create Tango DeviceServers.
 - Communications over Ethernet.
 - And more...

Engineering tasks

Implementation of Real Time OS

- Some Screenshots: Base System configuration



Engineering tasks

Implementation of Real Time OS

- Some Screenshots: Drivers configuration

```
Linux/microblaze 5.4.0 Kernel Configuration
submenus ---). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features,
ie < > module capable

*** Compiler: microblazeel-xilinx-linux-gcc (GCC) 9.2.0 ***
General setup --->
Endianness selection (Little endian) --->
[ ] SMP support (EXPERIMENTAL)
Platform options --->
Processor type and features --->
Kernel features --->
Bus Options --->
General architecture-dependent options --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
IO Schedulers --->
Executable file formats --->
Memory Management options --->
[*] Networking support --->
[ ] Device Drivers --->
File systems --->
Security options --->
-* Cryptographic API --->
Library routines --->
Kernel hacking --->
```

Engineering tasks

Implementation of Real Time OS

- Some Screenshots: Network configuration (very important)

```
Networking options (submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features.
Module < > module capable

<*> Packet socket
< > Packet: sockets monitoring interface
< > Unix domain sockets
< > UNIX: socket monitoring interface
< > Transport Layer Security support
< > Transformation user configuration interface
< > PF_KEY sockets
[*] TCP/IP networking
  [ ] IP: multicasting
  [ ] IP: advanced router
  [ ] IP: kernel level autoconfiguration
  < > IP: tunneling
  [ ] IP: GRE demultiplexer
  [ ] IP: TCP syncookie support
  < > Virtual (secure) IP: tunneling
  < > IP: Foo (IP protocols) over UDP
  < > IP: AH transformation
  < > IP: ESP transformation
  < > IP: IPComp transformation
  <*> INET: socket monitoring interface
  < > UDP: socket monitoring interface
  < > RAW: socket monitoring interface
  [ ] INET: allow privileged process to administratively close sockets
  [ ] TCP: advanced congestion control ----
  [ ] TCP: MD5 Signature Option support (RFC2385)
  < > The IPv6 protocol ----
  [ ] Security Marking
  [ ] Timestamping in PHY devices
  [ ] Network packet filtering framework (Netfilter) ----
  [ ] BPF based packet filtering framework (BPFILTER) ----
  < > The DCCP Protocol ----
  < > The SCTP Protocol ----
  < > The Reliable Datagram Sockets Protocol
  < > The TIPE Protocol ----
  < > Asynchronous Transfer Mode (ATM)
  < > Layer Two Tunneling Protocol (L2TP) ----
  <*> 802.1d Ethernet Bridging
  [*] IGMP/MLD snooping
  < > Distributed Switch Architecture ----
  [ ] Select < Exit > < Help > < Save > < Load >
```

Engineering tasks

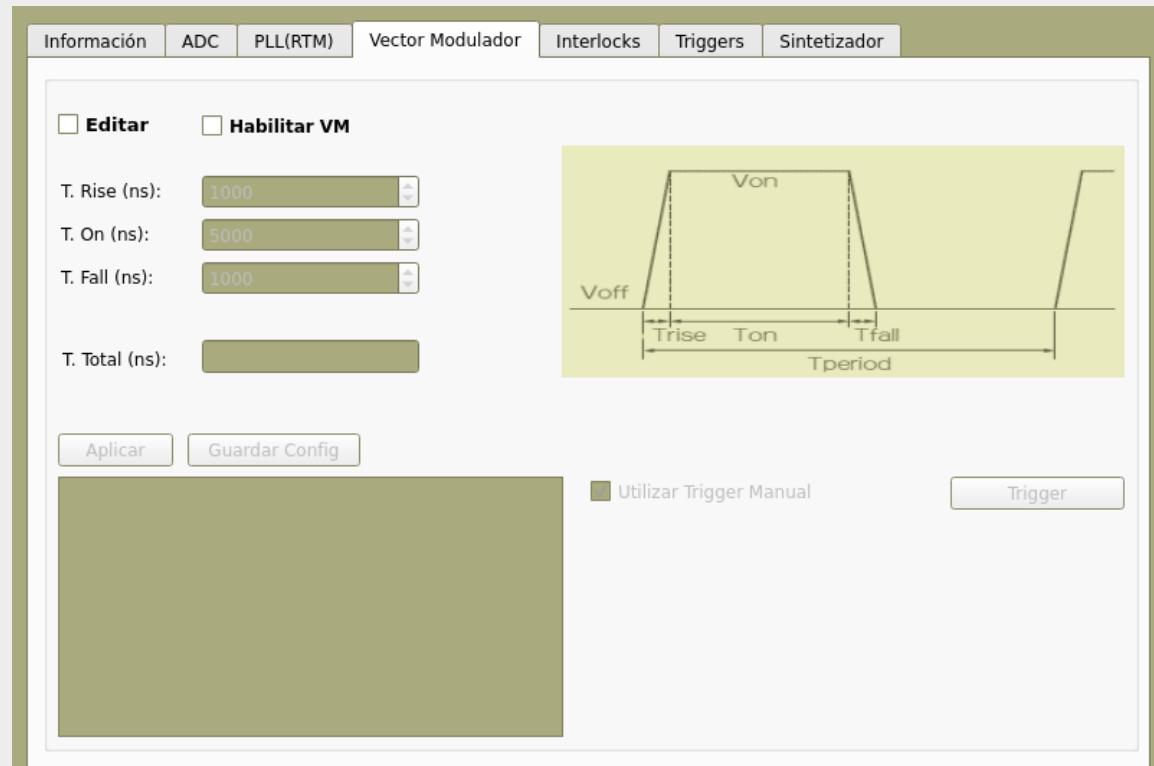
Implementation of Control Software User Interface

- Apart of this, We have implemented a high-level application on the AMC CPU (running Debian Linux) programmed with C++ (using thread methodology and custom classes)
- Although it is under development, it allows us to:
 - Full system access and control (such as Root user).
 - Hardware access and configuration (with specific Drivers).
 - QT Graphical User Interface (GUI) to interact.
 - Generation of custom signals and triggers (communicating with the developed IP cores).
 - Management of Interlocks.
 - Logger
 - ...

Engineering tasks

Implementation of control Software User Interface

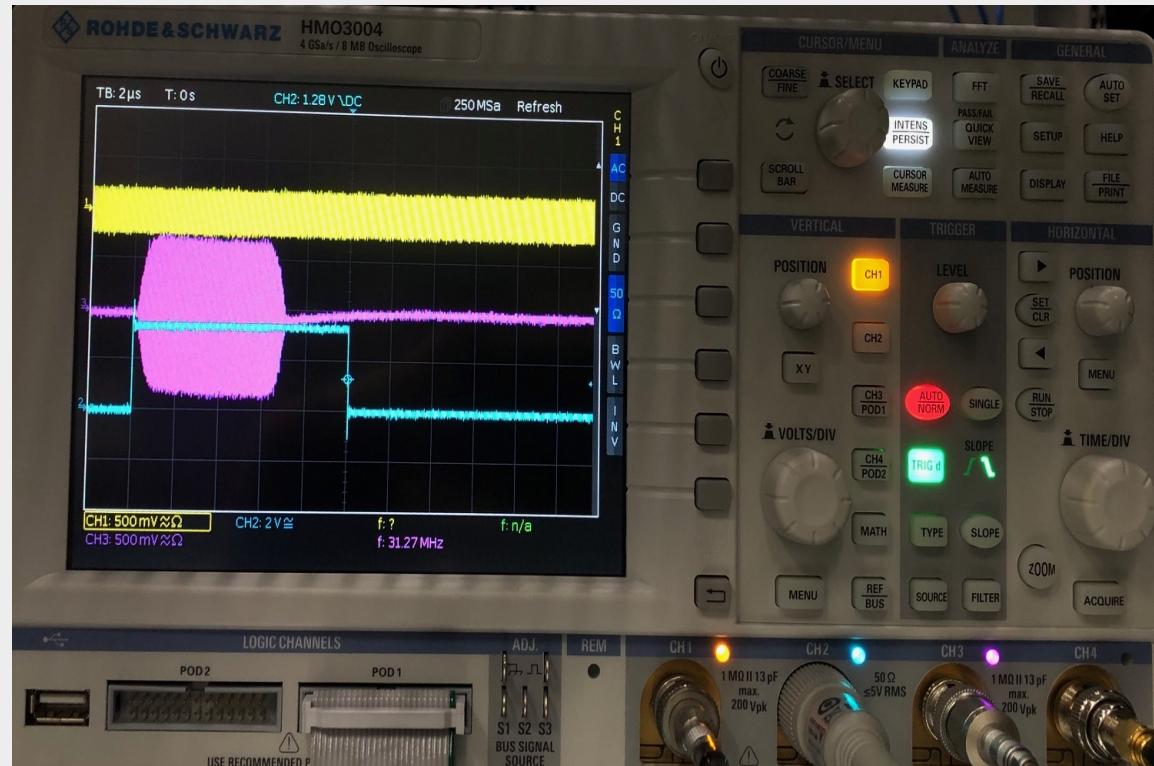
- Screenshot: VM Tab to set pulse format



Engineering tasks

Implementation of control Software User Interface

- **Picture:** Capture of the generated pulse and trigger



Engineering tasks

Implementation of control Software User Interface

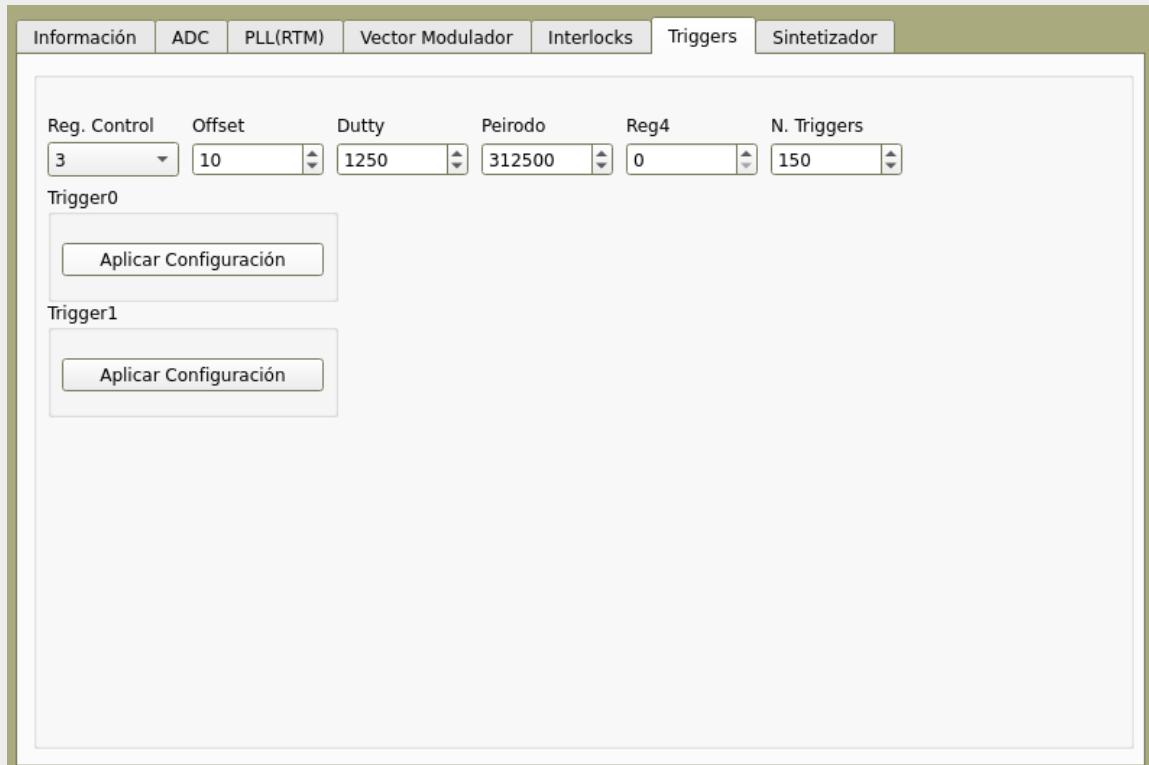
- **Screenshot:** Interlock Tab to manage interlock signals

Información	ADC	PLL(RTM)	Vector Modulador	Interlocks	Triggers	Sintetizador	
Enable							
HARDWARE							
MLVDS0	<input checked="" type="checkbox"/> Enabled	MLVDS1	<input checked="" type="checkbox"/> Enabled	MLVDS2	<input checked="" type="checkbox"/> Enabled	MLVDS3	<input checked="" type="checkbox"/> Enabled
SOFTWARE							
T1	<input type="checkbox"/> Enabled		<input type="checkbox"/> Enabled		<input type="checkbox"/> Enabled		<input type="checkbox"/> Enabled
Polaridad							
MLVDS0	<input checked="" type="checkbox"/> High Level	MLVDS1	<input type="checkbox"/> High Level	MLVDS2	<input type="checkbox"/> High Level	MLVDS3	<input checked="" type="checkbox"/> High Level
Prioritaria							
MLVDS0	<input type="checkbox"/> Prioritaria	MLVDS1	<input type="checkbox"/> Prioritaria	MLVDS2	<input type="checkbox"/> Prioritaria	MLVDS3	<input checked="" type="checkbox"/> Prioritaria
Activar							
MLVDS0	<input type="checkbox"/> ON	MLVDS1	<input type="checkbox"/> ON	MLVDS2	<input type="checkbox"/> ON	MLVDS3	<input type="checkbox"/> ON
Guardar Config							

Engineering tasks

Implementation of control Software User Interface

- **Screenshot:** Triggers Tab to generate triggers



Engineering tasks

Implementation of TANGO platform



Engineering tasks

Implementation of TANGO platform

- Tango is an distributed control system framework to build software for large control systems.
- Defines a communication protocol, an Application Programmers Interface (API) and provides a set of tools and libraries.
- Tango is an Open Source solution for SCADA systems.
- It is build around concept of devices and device classes.

Engineering tasks

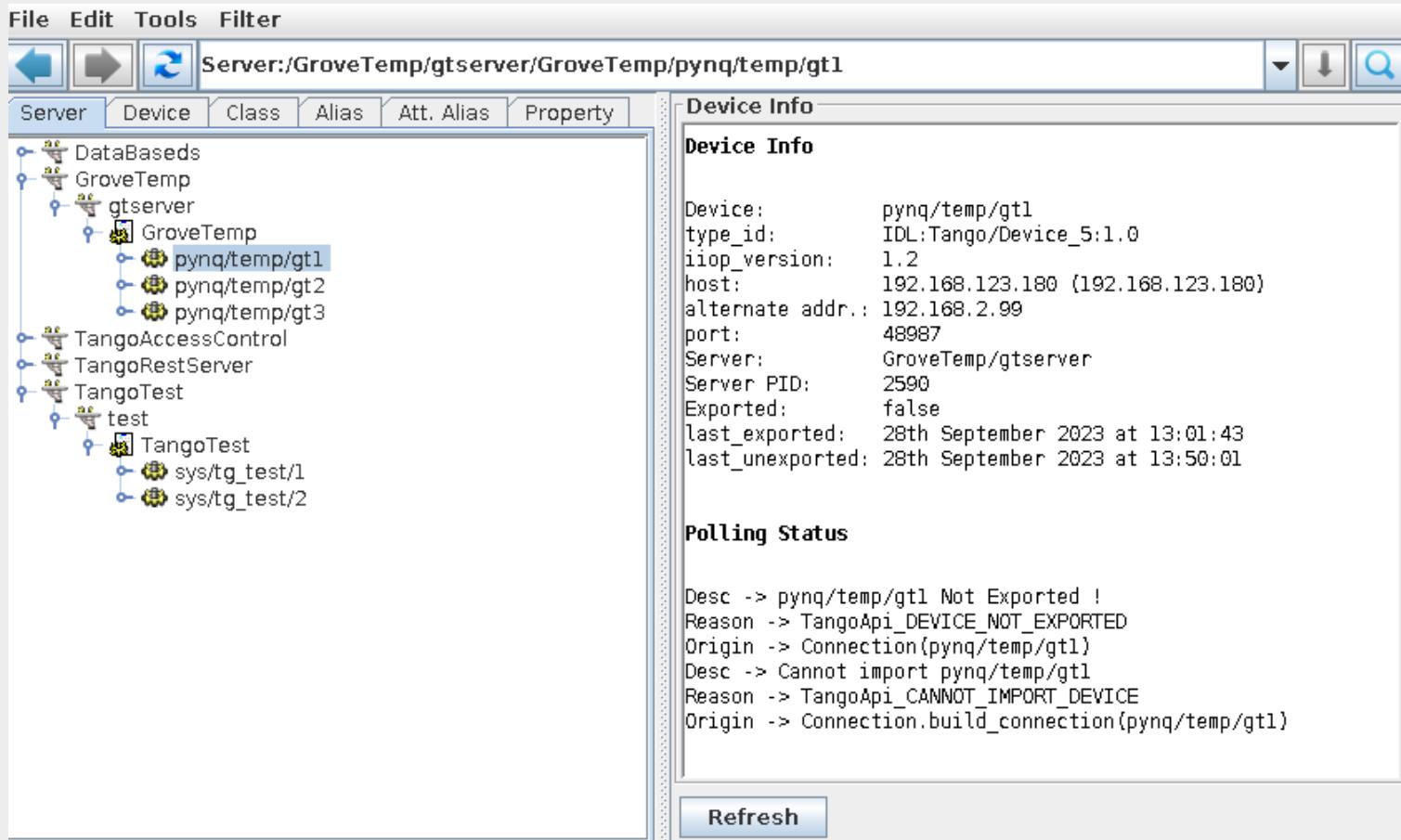
Implementation of TANGO platform

After several tests...

- We have installed, configured and launched the TANGO platform on our server in LLRF system.
- We have learned how to use its set of tools and libraries.
- And We have implemented the necessary devices and device classes to access sensors of the system.
- We can monitor events and alarms.
- Programmed classes in Python.

Engineering tasks

Implementation of TANGO platform



Device Info

Device Info

Device: pynq/temp/gt1
type_id: IDL:Tango/Device_5:1.0
iiop_version: 1.2
host: 192.168.123.180 (192.168.123.180)
alternate addr.: 192.168.2.99
port: 48987
Server: GroveTemp/gtserver
Server PID: 2590
Exported: false
last_exported: 28th September 2023 at 13:01:43
last_unexported: 28th September 2023 at 13:50:01

Polling Status

Desc -> pynq/temp/gt1 Not Exported !
Reason -> TangoApi_DEVICE_NOT_EXPORTED
Origin -> Connection(pynq/temp/gt1)
Desc -> Cannot import pynq/temp/gt1
Reason -> TangoApi_CANNOT_IMPORT_DEVICE
Origin -> Connection.build_connection(pynq/temp/gt1)

Refresh

Conclusions (2023)

- We have extended the first functional hardware version of our LLRF system.
- We have improved our knowledge of the uTCA open standard, and LLRF systems.
- We are close to finish an operational version (hardware + firmware + software) of the system, to be tested at CERN.

Next Steps (2024)

To Do

- We have some tasks to do in the coming months.

We will:

- expand our software control application,
- develop graphical user interfaces to monitor and control the TANGO platform (with PyTango or Qtango frameworks),
- program the rest of main software tasks over a real-time operative system,
- signal calibration,
- multiple capture and data synchronization,

Next Steps (2024)

To Do

- optimize existing firmware and devices,
- develop new hardware or software to control the external equipment of an LLRF system.

- Finally, write a paper about the conclusions of the above content.



**Thanks
for your attention**