Firmware

Cayetano Santos

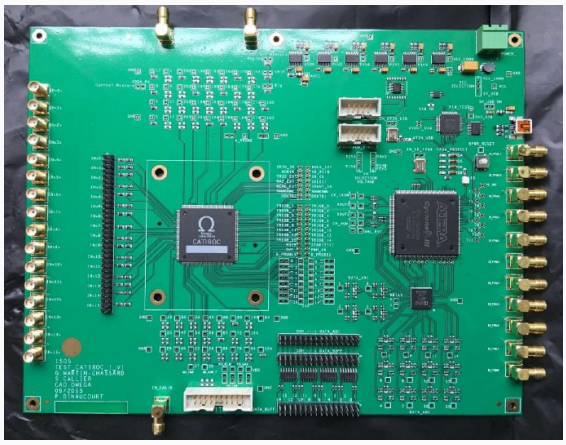
8th & 9th /03/2018 - JUNO sPMT Electronics Review

On behalf of the Juno sPMT team

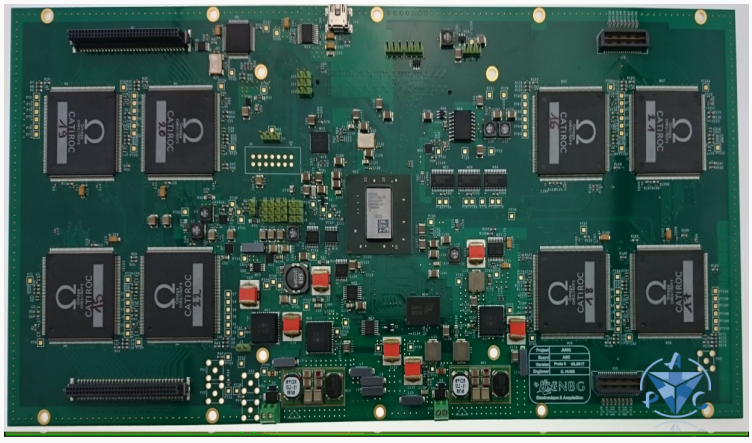
Outline

Overview

Context



Omega test card



ABC card

Need to develop test and production firmware for two platforms, reducing complexity and optimizing code reuse

- Common VHDL code base (single top level) for different hardware
- *Omega* (Altera Cyclone III based) test board - 16 ch.
- *ABC* (Xilinx Kindex 7 based) card - 128 ch
- C++/Matlab slow control & monitoring
- Top-down description of hierarchy blocks
- Developed from scratch, full stack
- In depth code testing achieved with help of dedicated acq. software
- Quartus 13, ISE 14.7 and Vivado 2017.2 compliant

Features

Basic features

Trigerless driven data flow
Huge pipelining

- Event count tagging
- 64 / 80 bits event modes
- Phy. event / data rate online monitoring
- Trigger rate up to 5 MHz.
- Dead time computing
- Optional ASIC data gray decoding
- Sync ASIC and FPGA time stamps
- Discriminator capture and time tagging

Advanced features

Management of variable ASIC latency
(Not an oscilloscope !)

- Extended time stamping (N bits)
- Optional DDS flow and TOT measurement
- Coarse time window scan (fine time INL)
- Parallel (all channels) S-Curve
- IP Bus ready, etc.

Code

VHDL Coding

```
-- *** P1: simple counter...
-- *** P2: asic data...
-- *** P2: asic data...
-- *** P2: asic data...
-- *** P3: asic slow control...
-- *** P4: adc data...
-- *** P5: adc slow control...
-- *** P6: s-curve computing...
-- ** P1: Counter generator...
-- ** P2 & P3 - CatiROC manager...
-- *** Diff to single ended...
-- *** Catiroc manager...
-- *** Catiroc manager

   U_catiroc : for i in 0 to nb_asics-1 generate
      U0_catiroc : catiroc
         generic map (num_of_discri_channels => num_of_discri_channels,
                      p2_data_width          => p2_data_width)
         port map (main_rst          => global_reset,
                   -- input data stream / strobe
                   cati_dataout      => cati_dataout(2*(i+1)-1 downto 2*i),
                   cati_strobe       => cati_strobe(2*(i+1)-1 downto 2*i),
                   -- Static values
                   triggers          => triggers(16*(i+1)-1 downto 16*i),
                   ovf               => ovf,
                   --
                   -- CatiROC
                   --
                   NewEvent          => open,  -- all events combined
                   NewEventByChannel => NewEventByChannel(i),
                   resetb            => resetb_tmp(i),
                   StartSys          => StartSys_tmp(i),
                   pll_locked        => locked_sig,
                   discri_falling    => discri_falling,



-- **** Read clock...
-- **** Read clock...
-- **** Clocks...
-- **** P2 - Data capture interface...
-- **** P2 - Data capture interface...
-- **** P3 - Slow control interface...
-- **** P3 - Slow control interface...
-- *** Catiroc-usb If : data capture...
-- **** Juno...
-- *** Catiroc-usb If : slow control...
-- **** Juno...
-- ** P6: S Curve...
-- *** Juno...
-- ** Control delay...
-- *** Main pll...
-- **** Juno...
```

```
:-kindex7/top.vhd[CatiROC-test-firmware] 51% -tdc vtool -Chg +1 SP/s WK (*) Projectile FlyC:1/0 ARev Outl
```

VHDL code with emacs vhdl-mode

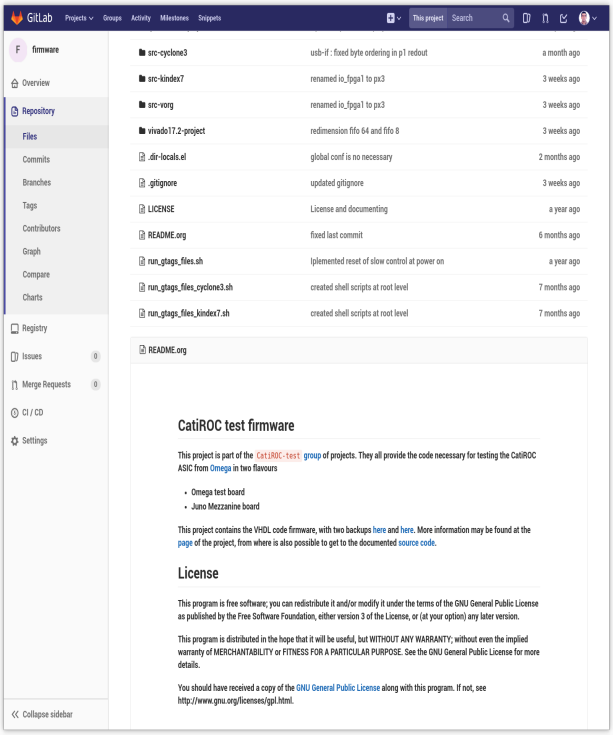
Strategy

Source code extracted from org (markup) files using *vhdl-tools* library

Hints

- Code common to two platforms
- Abuse of modularity + hierarchy of building blocks
- Extended use of generics (data / params buses)
- `nb_asics` between 1 and 8
- Design based on peripherals / address decoding
- Command / interrupt oriented
- Architecture of converging data streams

Version control using `git`



Remote repository

Files under version control

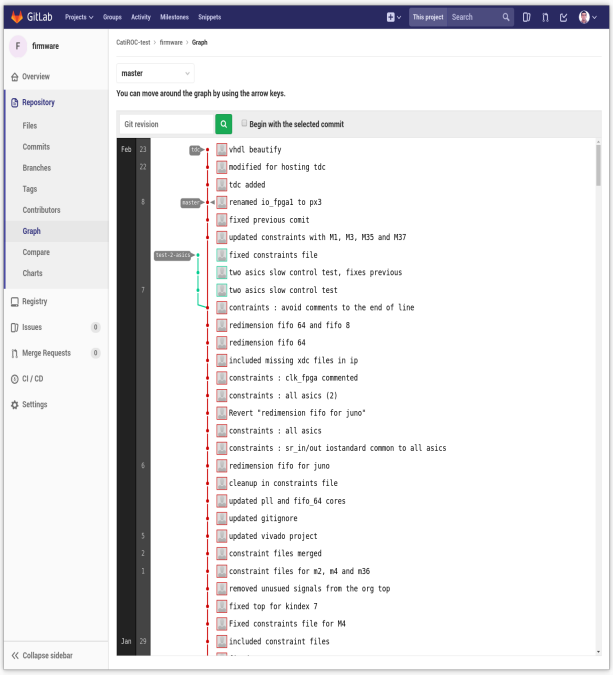
Only text files + text editor

- Quartus : full project
- Vivado 17.2

  `build.tcl` generate project
  `*.xci/xdc` buffers
  … and that's it
- VHDL files
- Constraint files

… plus backup mirrors (in2p3, …)

Version control using `Gitlab` @ cern



Remote repository
commit history

Benefits

Getting back in history, multiple versions in parallel, etc.

- Full commit history
- Tracking / documenting modifications
- Several branches in parallel : **`master`**, **`devel`**, **`pages`**, **`feature`** (tdc, multiasic, etc.)
- Multiple collaborators
- Issues tracking
- Web based diff
- Mirror for CI + pages @ **`gitlab.com`**

Version control using `Gitlab` : pages

`Online site`

In2p3 mirror

`https://gitlab.in2p3.fr/CatiROC-test/firmware`

Web site

Contains all relevant information

- `Gitlab` pages branch in main repository
- Developed in org markup + `css`
- Hosted @ `Gitlab.org`

Share information with users

- Architecture, releases, specific features
- Project structure
- Internals, data format
- How to use, troubleshooting, etc.

Version control using `Gitlab` : documentation

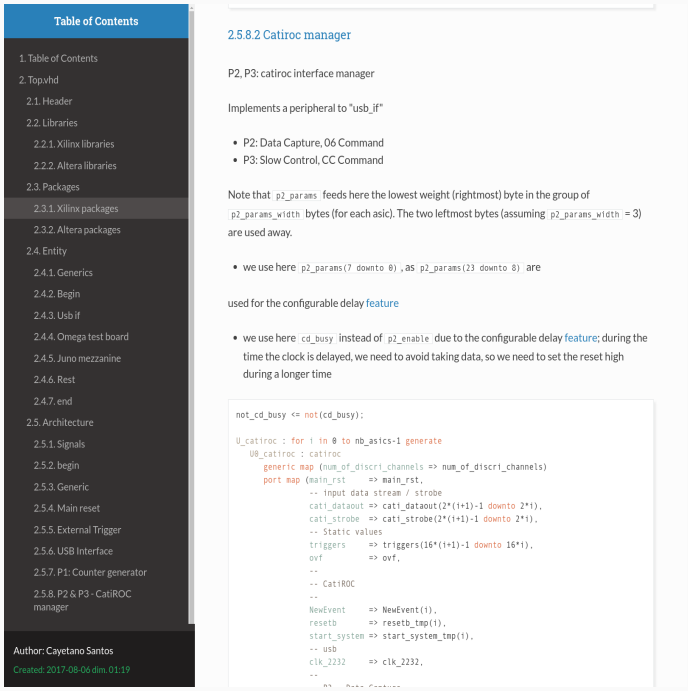


Table of Contents

Author: Cayetano Santos

Created: 2017-08-06 dim. 01:19

2.5.8.2 Catiroc manager

P2, P3: catiroc interface manager

Implements a peripheral to "usb_if"

- P2: Data Capture, 06 Command
- P3: Slow Control, CC Command

Note that `p2_params` feeds here the lowest weight (rightmost) byte in the group of `p2_params_width` bytes (for each asic). The two leftmost bytes (assuming `p2_params_width` = 3) are used away.

- we use here `p2_params(7 downto 0)` ,as `p2_params(23 downto 8)` are

used for the configurable delay feature

- we use here `cd_busy` instead of `p2_enable` due to the configurable delay feature; during the time the clock is delayed, we need to avoid taking data, so we need to set the reset high during a longer time

```
not_cd_busy <= not(cd_busy);

U_catiroc : for i in 0 to nb_asics-1 generate
   U0_catiroc : catiroc
      generic map (num_of_discri_channels => num_of_discri_channels)
      port map (main_rst      => main_rst,
                -- input data stream / strobe
                cati_dataout => cati_dataout(2*(i+1)-1 downto 2*i),
                cati_strobe  => cati_strobe(2*(i+1)-1 downto 2*i),
                -- Static values
                triggers     => triggers(16*(i+1)-1 downto 16*i),
                ovf          => ovf,
                --
                -- CatiROC
                --
                NewEvent     => NewEvent(i),
                resetb       => resetb_tmp(i),
                start_system => start_system_tmp(i),
                -- usb
                clk_2232     => clk_2232,
                --
```

Online view @ gitlab

```
#+end_src

- Catiroc manager

  P2, P3: catiroc interface manager

  Implements a peripheral to "usb_if"

   - P2: Data Capture, 06 Command
   - P3: Slow Control, CC Command

  Note that p2_params feeds here the lowest weight (rightmost) byte in the group of
  p2_params_width bytes (for each asic). The two leftmost bytes (assuming
  p2_params_width = 3) are used away.

  - we use here p2_params(7 downto 0), as p2_params(23 downto 8) are
  used for the configurable delay feature

   - <<cd_busy>> we use here cd_busy instead of p2_enable due to the
     configurable delay feature; during the time the clock is delayed,
     we need to avoid taking data, so we need to set the reset high
     during a longer time

  #+begin_src vhdl-tools
    not_cd_busy <= not(cd_busy);
  #+end_src

  Loop over the number of asics, instantiating one catiroc module each time, and
  feeding the right signals. Note the the parameter num_of_discri_channels is
  necessary to limit the fpga resources.

  #+begin_src vhdl-tools
    U_catiroc : for i in 0 to nb_asics-1 generate
       U0_catiroc : catiroc
          generic map (num_of_discri_channels => num_of_discri_channels,
                       p2_data_width          => p2_data_width)
          port map (main_rst          => global_reset,
                    -- input data stream / strobe
                    cati_dataout      => cati_dataout(2*(i+1)-1 downto 2*i),
                    cati_strobe       => cati_strobe(2*(i+1)-1 downto 2*i),
                    -- Static values
                    triggers          => triggers(16*(i+1)-1 downto 16*i),
                    ovf               => ovf,
                    --
                    -- CatiROC
                    --
                    NewEvent          => open,  -- all events combined
                    NewEventByChannel => NewEventByChannel(i),
                    resetb            => resetb_tmp(i),
                    StartSys          => StartSys_tmp(i),
  #+end_src

  . Read clock...
- Catiroc manager...
  . Clocks ...
  . P2 - Data capture interface...
  . P3 - Slow control interface...
- Catiroc-usb If : data capture...
  . Omega                                                    :altera:...
  . Juno                                                     :xilinx:
vorg/top.org    53% -test-2-asics vOrg -Chg ing WK (*) Projectile - ARev
```

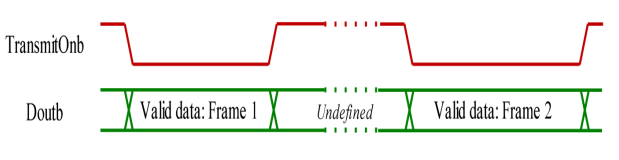
Org markup file

Documentation is extracted for web rendering.
Withing the org files, code is embedded in source blocks, along with the documentation.

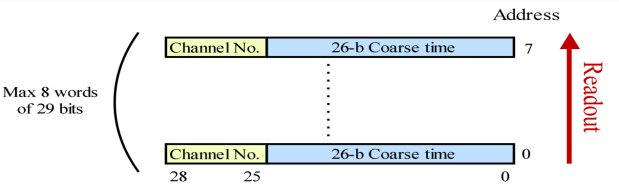
Architecture (bottom to top)

Lowest level: data capture

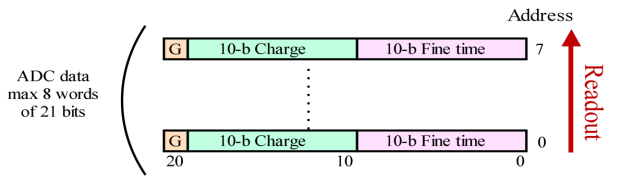
from CatiROC datasheet



Data out



Frame 1



Frame 2

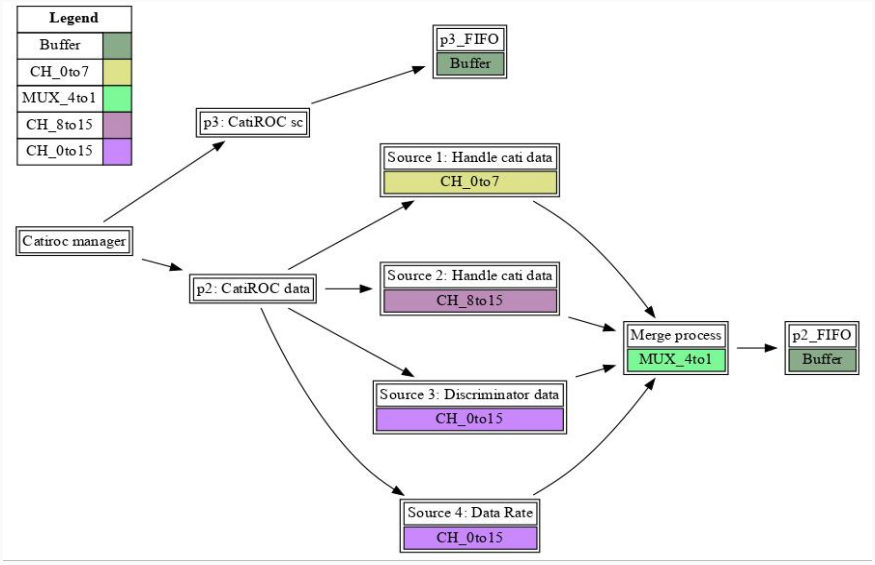
Physical data

- One readout line for 8 channels
- One event split off in two frames: 29 + 21 bits
- Random delay between frames
- Variable number of channels by frame

Additional data computed online

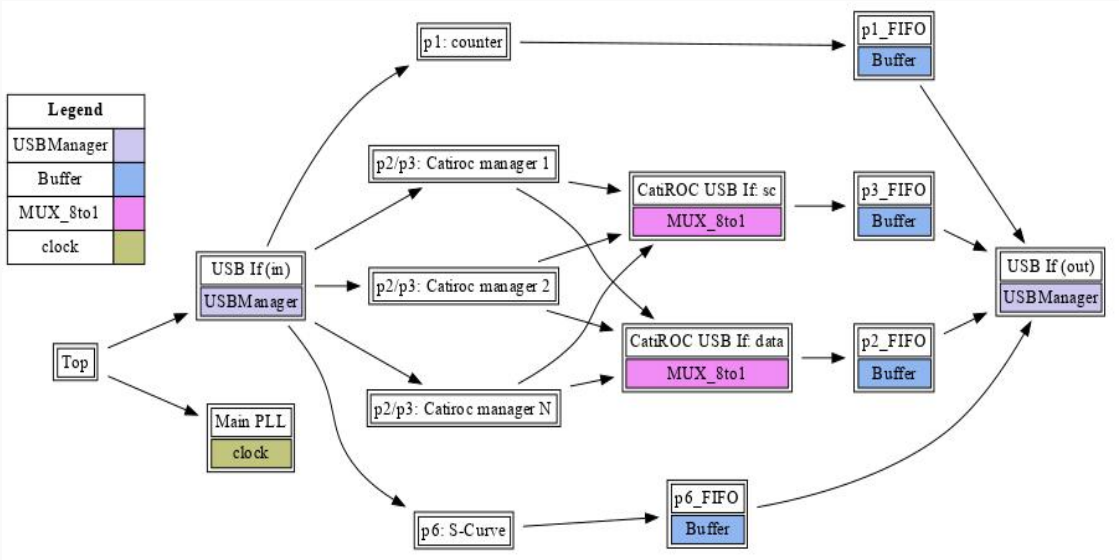
- Event counter by channel
- Card & asic number
- Trigger rate

Middle level: CatiROC controller



- Two peripherals: slow control and data readout
- Stream 1/2 : physical data channels, 1/9 to 8/16
- Stream 3 : discriminator data, channels 1 to 16
- Stream 4 : special info data
- 66 bytes of slow control by controller; slow control as payload
- Time ordered physical data

Top level



- One USB master enables one peripheral
- 1 Input / output USB manager; 1 Buffer (fifo) by peripheral
- N USB managers peripherals (simple counter, s-curve, etc.)
- 8 CatiROC controllers (2 peripherals/controller) in parallel
- 2 multiplexers (slow control and data readout peripherals)
- All CatiROC controllers are enabled in parallel
- Mixed data : time ordered only by channel

Data format

Physical event word, 80 bits, EventID = ’00’

```
'00' - Ch.Nb. (4 bits) - Coarse Time (26 bits) - Gain (1 bit) -
EventCounter (11 bits) - Charge (10 bits) - Fine Time (10 bits) -
'00000' (5 bits) - Card.Nb (8 bits) - ASIC.Nb (3 bits)
```

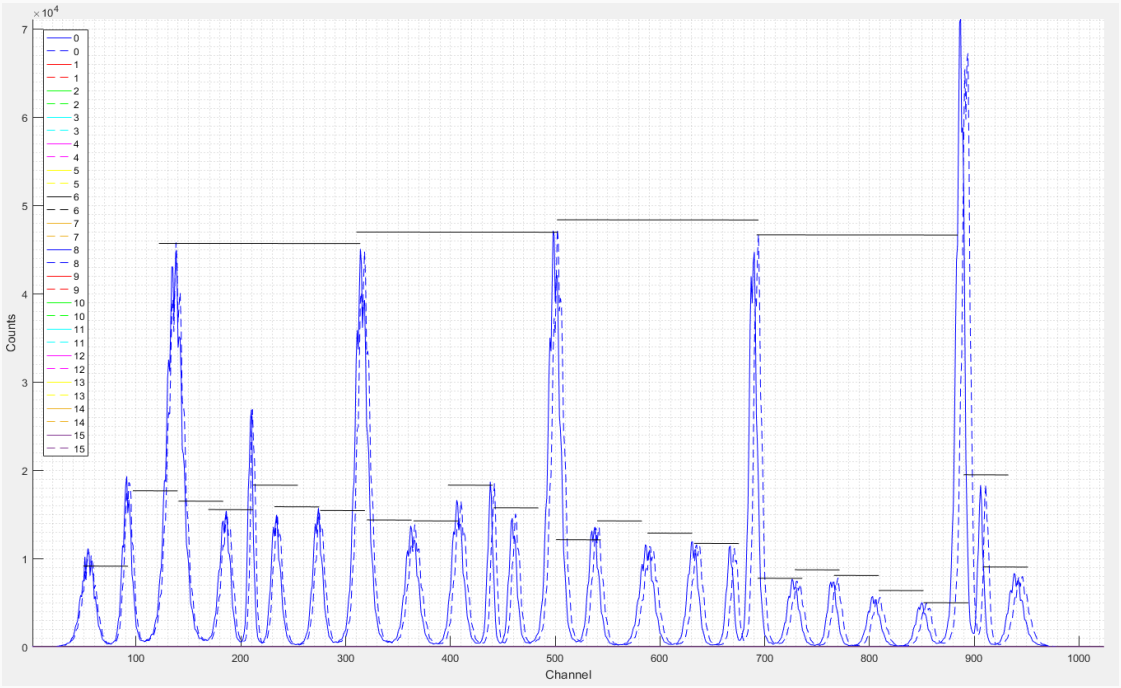
Special information word, 80 bits, EventID = ’01’

```
'01' - '0' (41 bits) - Ch.Nb. (5 bits) - TriggerRate (16 bits) -
'00000' (5 bits) - Card.Nb (8 bits) - ASIC.Nb (3 bits)
```

Discriminator information word, 80 bits, EventID = ’10’

```
'10' - Edge (1 bit) - Ch.Nb. (4 bits) - EventCounter (25 bits) -
Time Stamp (32 bits) - '00000' (5 bits) - Card.Nb (8 bits) -
ASIC.Nb (3 bits)
```

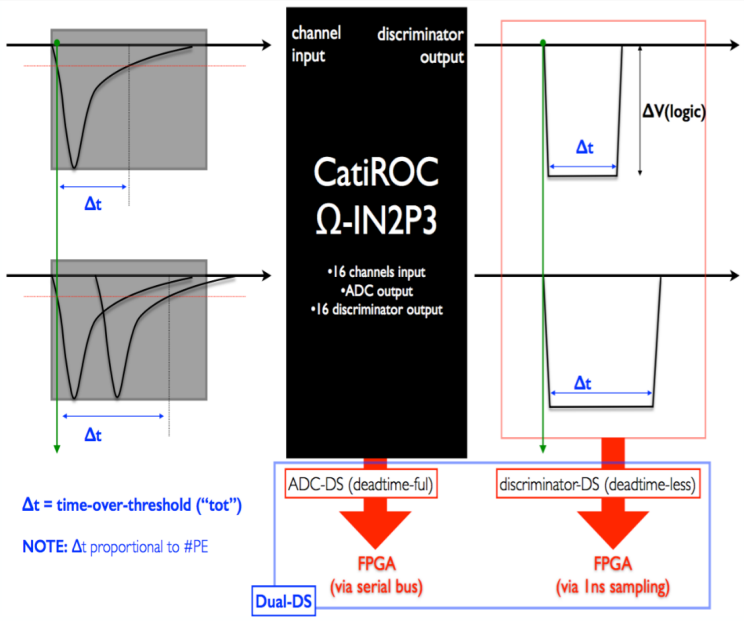
Advanced: Time window scanning



25 ns / 40 MHz time window

- Pulse generator in sync with daq
- Dynamic fine time histogramming / data saving
- Variable x ns. time steps for linearity characterization

Advanced: Dual Data Stream (under development)



from Juno Electronics Meeting 10/2017
M. Settimo, "CatiROC Status"

Physical data out of CatiROC has a dead time of few us.
Discriminator output is instantaneous.

Why ?

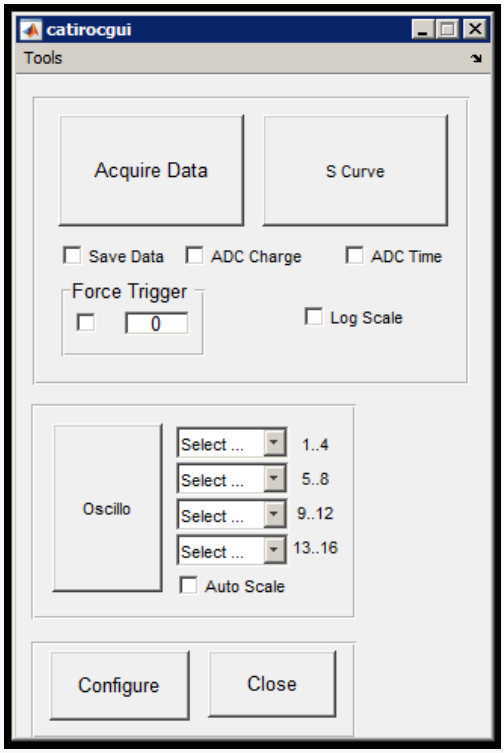
- (Physical data) dead time monitoring
- Very low (auto) dead time (~50 ns.)
- Delta difference between falling / rising edges
- Proportional to the charge

How ?

- 128 TDC in one Kindex 7 FPGA
- Sampling of discriminator signal at 1 GHz
- ISERDES based
- One (artificial) 80-bits event by pulse

Validation

Validation: software



GUI

GUI user front end

- Low level software in C++
- Gui / data visualization in Matlab
- Windows / Linux
- Amplitude/timing histogramming
- Row data recording for offline analysis
- S-curve displaying
- Peak fitting and analysis
- Fine time characterization, etc.

CLI scripting - under development

(with help of Julia [a])

```
using TestCardModule
myTestCard = TestCard()
tc_GetIsOpen(myTestCard)
...
```

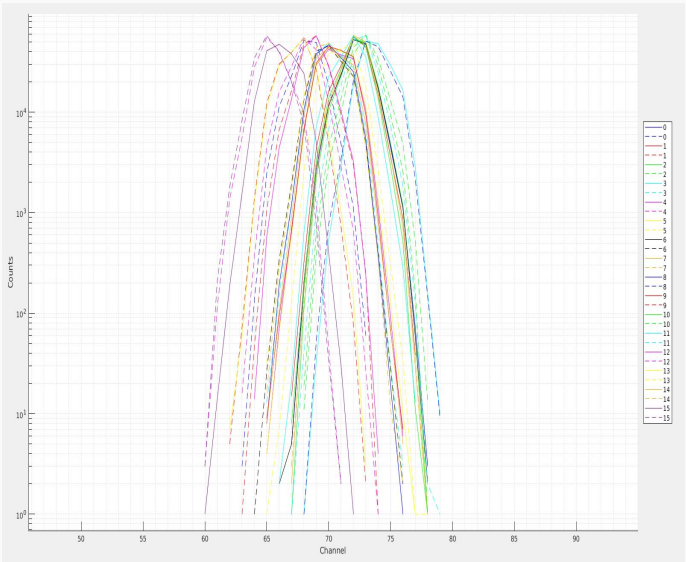
[a] https://julialang.org/

Validation with Omega test card / abc in mono asic mode (i)

Piedestals

One color by channel

Continuous (ping) / dotted (pong)

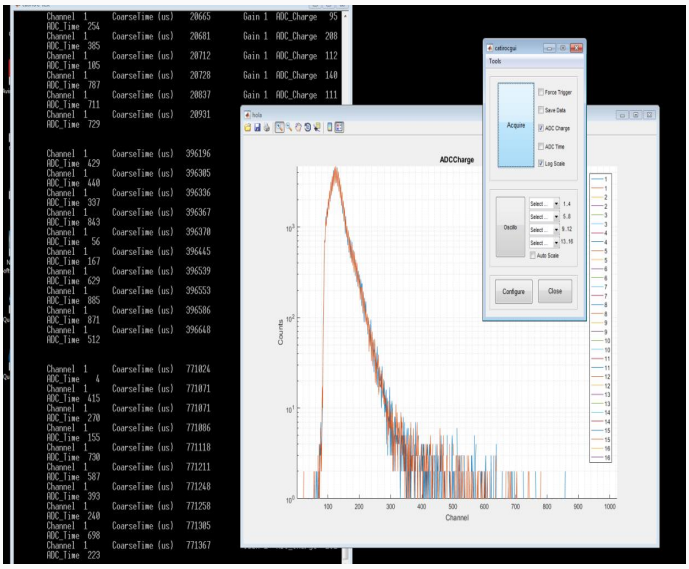


- One asic, 16 channels, external trigger
- All asics tested

Pulse generator

Variable amplitude

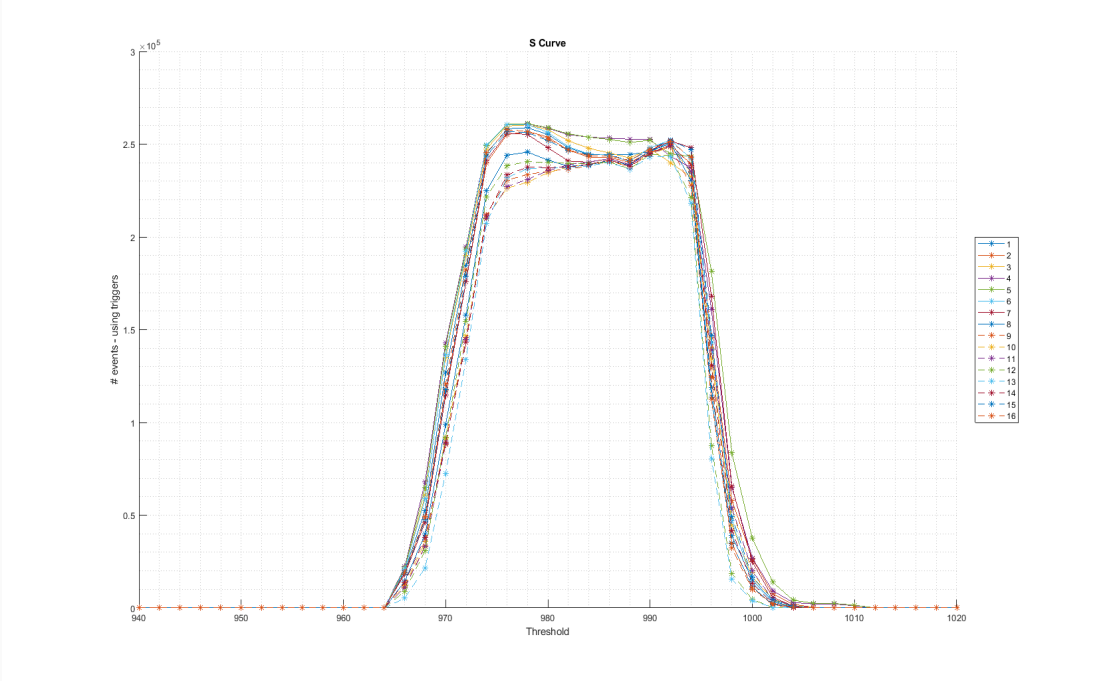
Basic data validity tests



- Internal trigger
- One asic, 16 channels

Validation with Omega test card / abc in mono asic mode (ii)

16 channels S-Curve scan



- All channels in parallel
- Programmable scan : min, max and step soft parameters
- Two methods : using data or discriminator

Conclusion

Conclusion

Summary

- All available online under Under GPLv3
- In depth test using Omega test card
- Preliminary tests using ABC card in one asic mode
- Validation site: `https://catiroc-test.gitlab.io/hardware/`
- Needed help for data analysis ...

What's next ?

- Two / eight asics : slow control and data readout
- Dual data stream : to validate
- Bug fixing
- Increase local buffers capacity with 1 GB DDR
- GCU interfacing
- Adapt for ABC v1