Software project

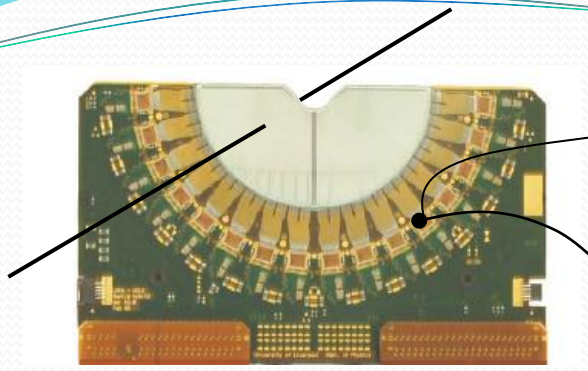
Outline

- VETRA framework – the staff we can re-use
- What is new
- Experience with the current software

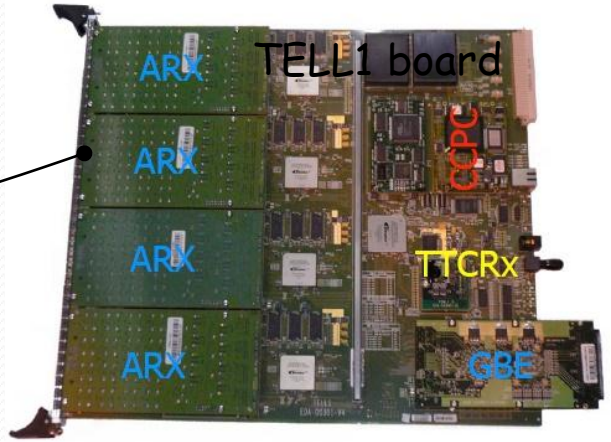
Manpower (usually ~ 30% - 50% time)

Staff: Agnieszka (current off-line monitoring), Kasia, Tomasz
Students: Michal (processing parameters trending), Adam, Mateusz

Postdoc from INP: Piotr



VELO front-end



Common electronic acquisition read-out board – **TELL1**

Digital/analogue input (interfaced to gigabit Ethernet)

FPGA based – Altera Stratix

Main goal – synchronisation, buffering and the data zero suppression (factor ~ 200)

From non-zero suppressed (2048 /sensor)

To zero suppressed (clusters only)

Technically it is a farm of parallel stream processors

Processing 36 threads at the same time

↓

Raw cluster buffer

↓

Reconstruction

How to control the hardware based zero suppression?

- not a trivial task
- processing runs in real time using hundred of threads
- huge amount of data
  88 sensors x 2048 channels x 10 bits x 1 MHz

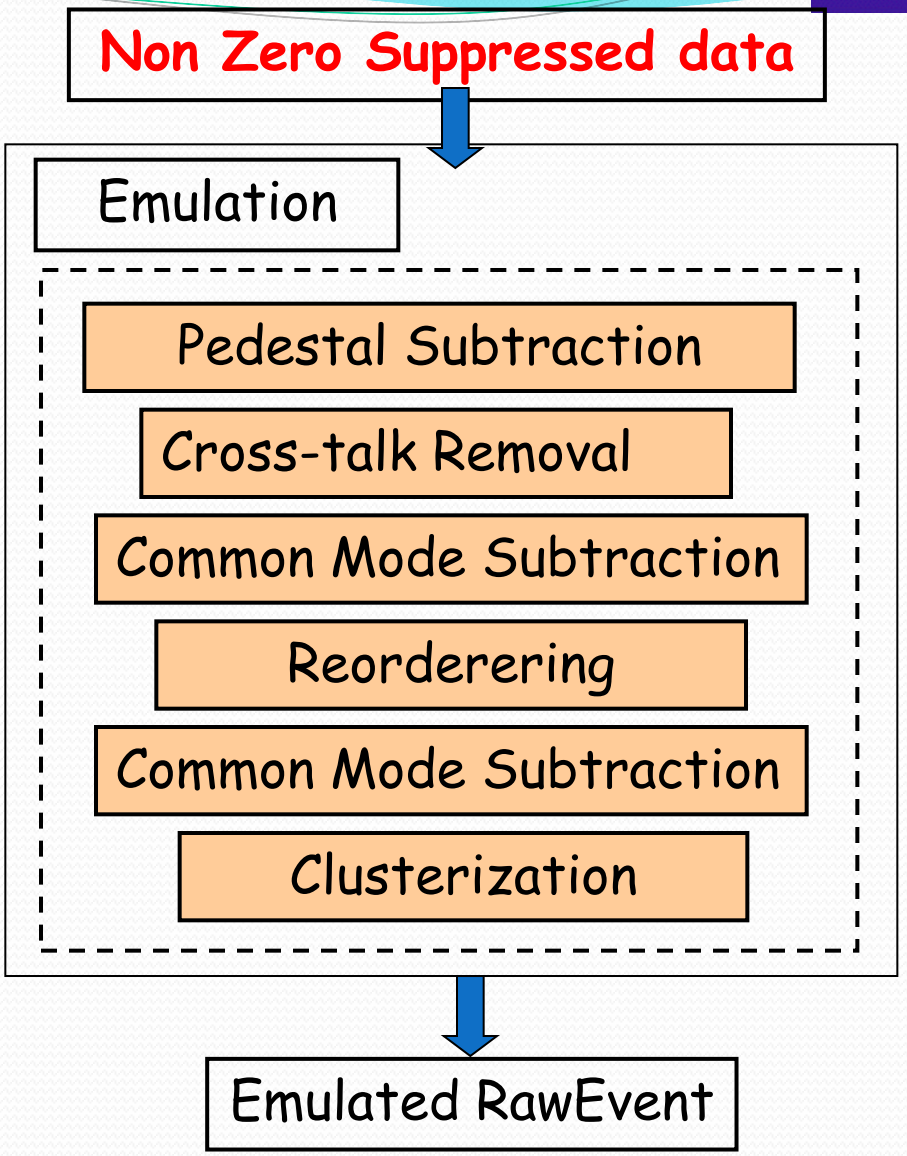
Solution - high level, bit-perfect emulation

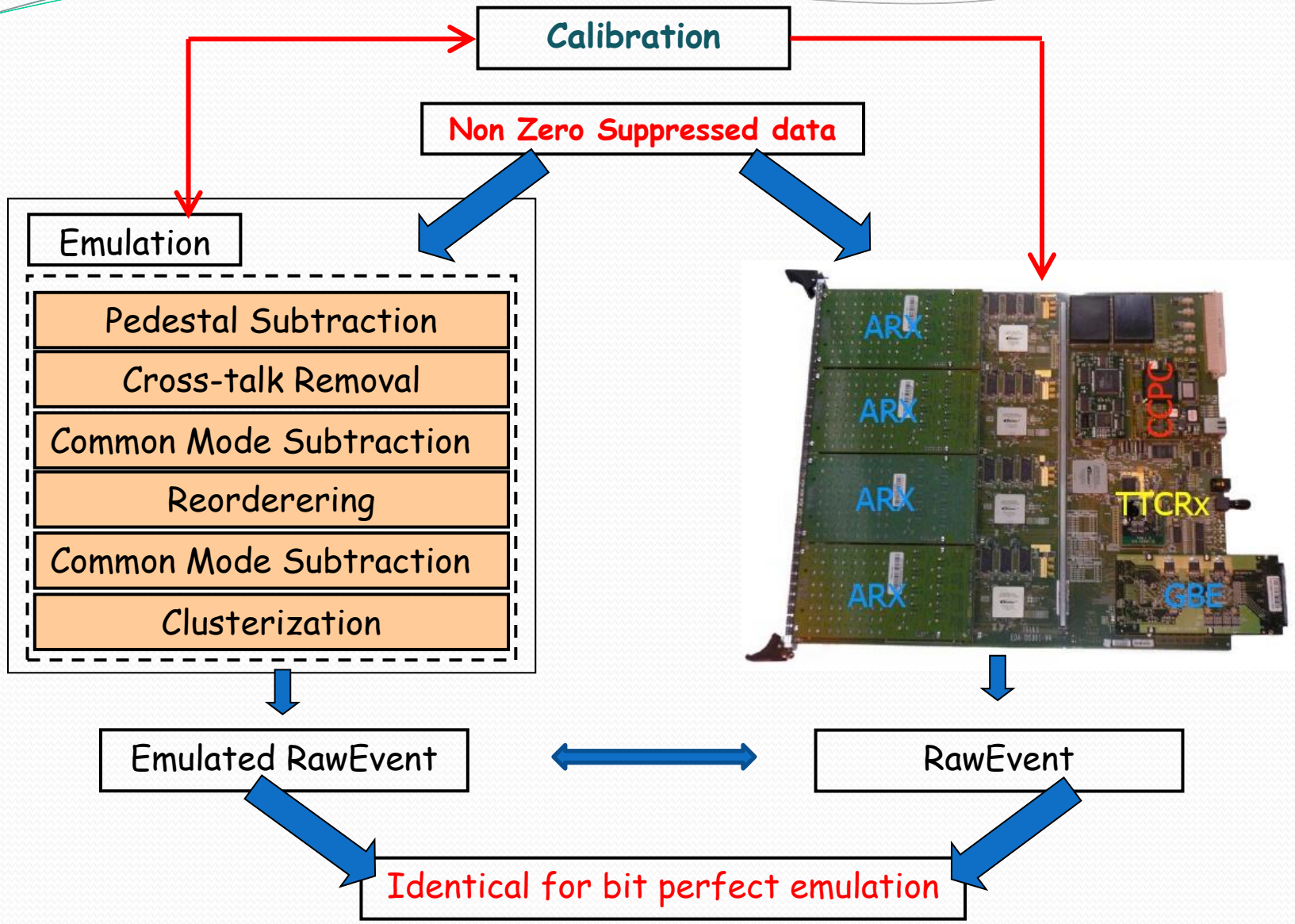
- runs off-line on single CPU
- uses real non-zero suppressed data
- high level model of the VHDL machine code from FPGAs
- calibration and monitoring

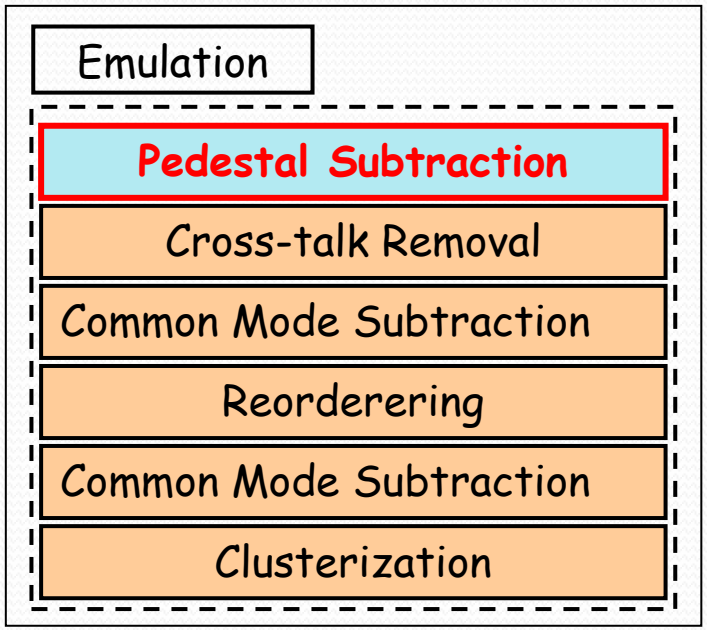
Emulation has the same structure and operates on the same data as the real processing done in FPGAs

Parameters used for readout board fixed in emulation

The full set of parameters that are required for the proper operation of the TELL1 boards amounts to ~ 10^6




Calibration
Non Zero Suppressed data
Emulation
Pedestal Subtraction
Cross-talk Removal
Common Mode Subtraction
Reorderering
Common Mode Subtraction
Clusterization
ARX
ARX
ARX
ARX
CCPC
TTCRx
GBE
Emulated RawEvent
RawEvent
Identical for bit perfect emulation



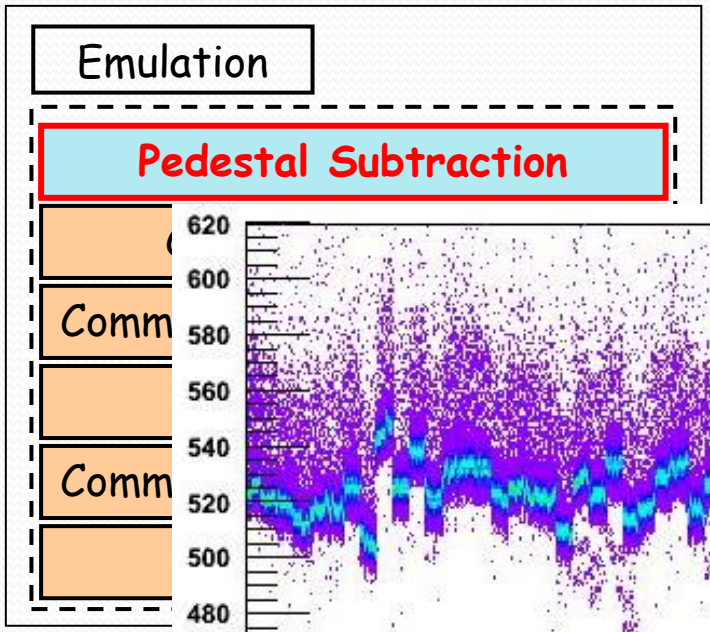
Pedestal following & subtraction

- first algorithm in the sequence
- running average algorithm
- following (training done off-line only)
- pedestals are calculated for each channel
- values are stored in the TELL1 memory

Critical for the quality of the zero-suppressed data

Any problem with pedestals will manifest itself as a change in occupancy

- careful monitoring required

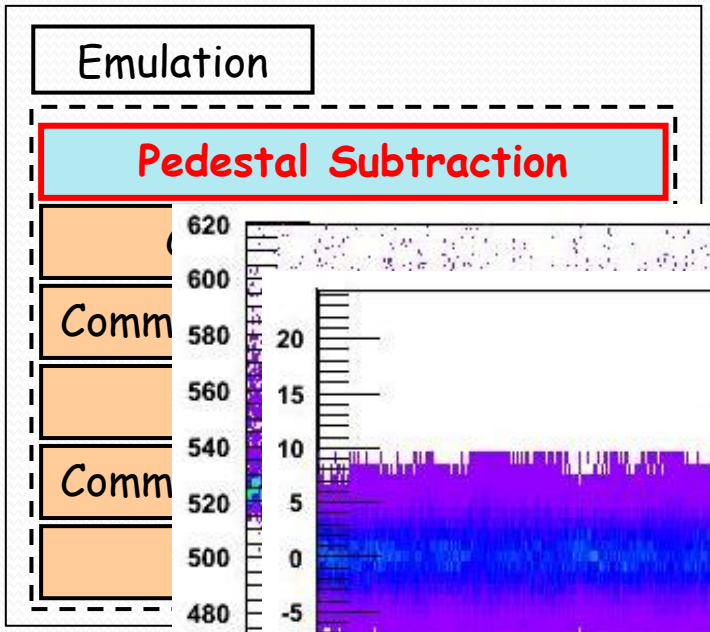


Pedestal following & subtraction

- first algorithm in the sequence
- running average algorithm
- following (training done off-line only)

ch channel
memory

e zero-suppressed data

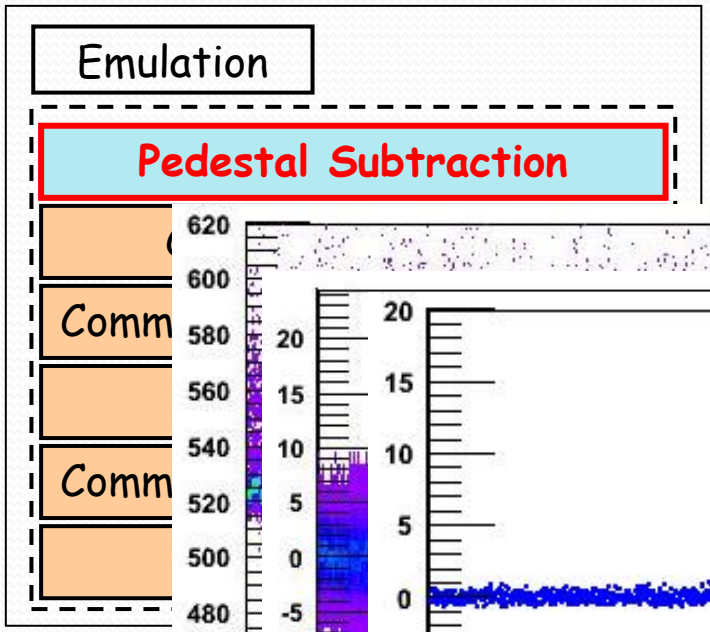


Pedestal following & subtraction

- first algorithm in the sequence
- running average algorithm
- following (training done off-line only)

ch channel

pressed data

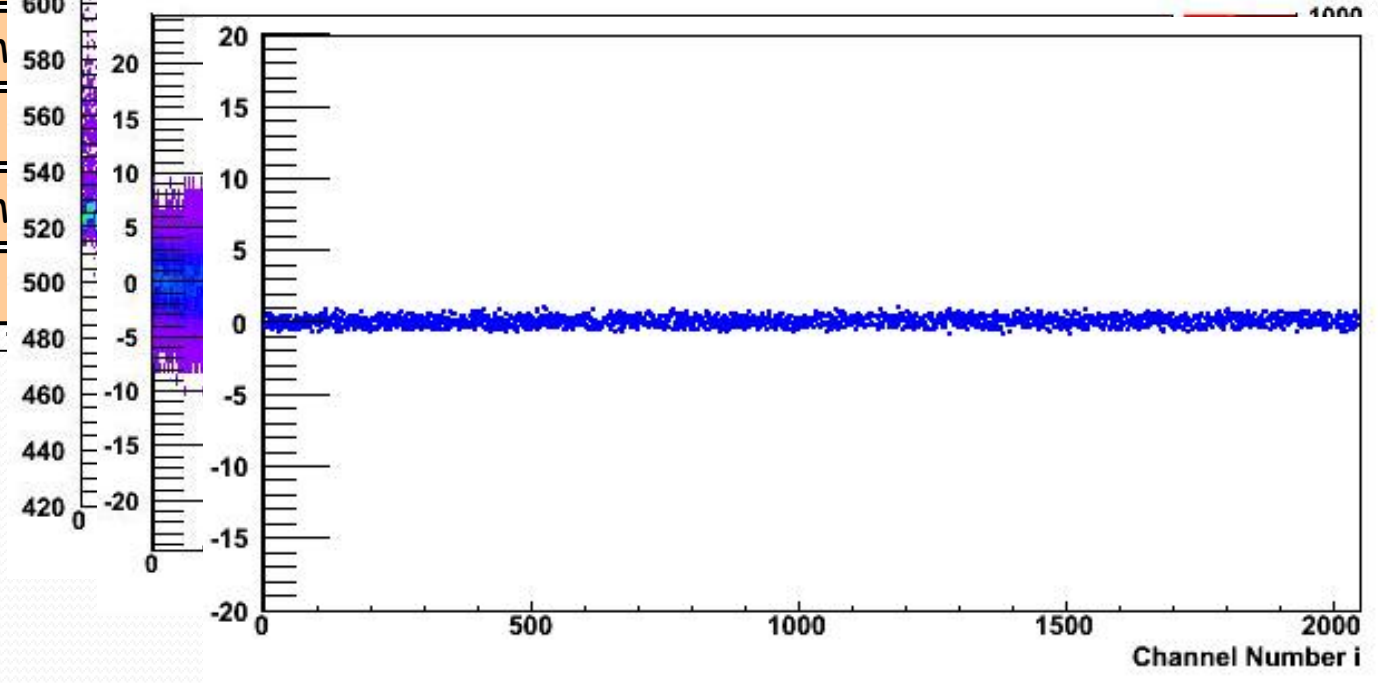


Pedestal following & subtraction

- first algorithm in the sequence
- running average algorithm
- following (training done off-line only)

ch channel

pressed data



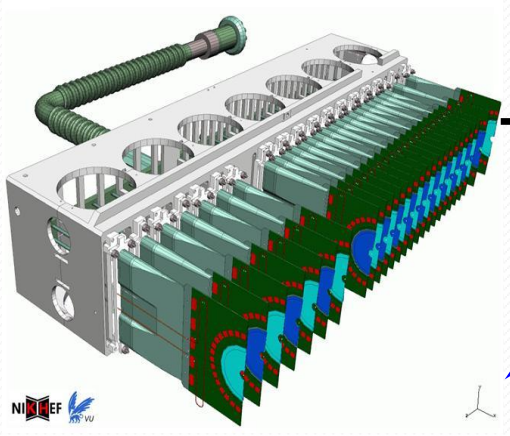
For the present VELO & ST we have a very strong synergy between the on-line processing (TELL1 based) and off-line processing/monitoring

- we cannot run high-level emulation in real time
- we cannot run TELL1 zero suppression without calibration

Solution

- first we need to perform the calibration run with non-suppressed data
- determine a set of processing parameters
- upload them to the TELL1 memory banks (on-line processing)
- create a SQLite data base (off-line processing)
- run the emulation regularly to monitor the processing algorithms
- monitor the output of the TELL1 boards to check the quality of the calibration (cluster rates, occupancies, landaus, …)

The bit-perfect emulation is essential!



Noise Data

VELOCOND

Gaudi Job (Vetra)

Tell1 Emulation

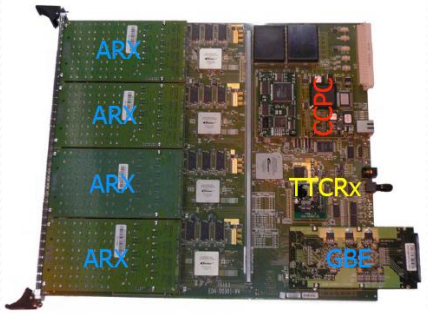
Monitoring/
verification job

Analysis

Tuned Processing Parameters

XML file

XML2PVSS



Tell1

Basic technicalities

- plug-in – wrapper approach
- the plug-in is an „engine" code – aka C module (HL model of VHDL)
- wrapper is C++ class that can communicate with the GAUDI framework (our official experiment software)
- on top of things we have Python configurables
- can compose our job (processing sequence) as we are pleased
- extremely flexible
- can be used to gauge the performance
- can act as a test bed for alternative algorithms
- ultimately provides the bit perfect emulation

No problem with re-using this approach for the upgrade

So what is new?

PLENTY

- on-chip processing (once diffused it cannot be changed)
- no room for mistakes
- current system is very forgiving (reload firmware and you are done!)
- we need hardware emulator
- FPGA based
- develop and test the algorithms with the emulator first

So what is new?

PLENTY

- Tell40
- decoding/re-formatting
- pre-processing (pixel/strips)
- first step of the pattern recognition?
- encoding/MEP packet building
- what needs to be emulated?
- we have nasty problems with „incomplete MEP events"
- no way to debug it, emulation could help I believe
- could we profit from a multi-threaded emulation?

PLANS

Kick-off meeting right after the summer

- will try to get on-board as many people as possible
- this should be a common project
- future test, calibration and monitoring platform
- could run fully emulated tracking (just a dream)