HGCAL DPG - Raw Data Handling

Calibration algorithms, Alpaka algorithms,

and HGCAL DQM

Yu-Wei Kao

National Taiwan University

TIDC workshop

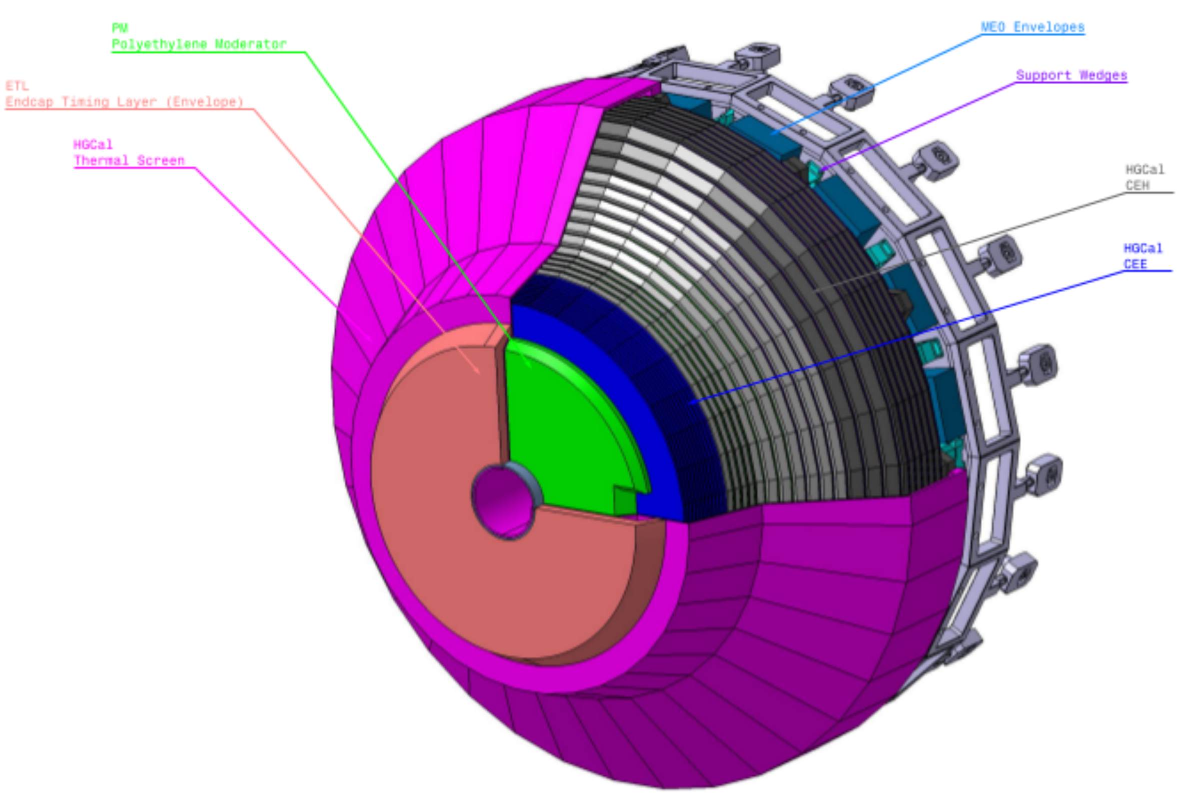
25. November. 2023

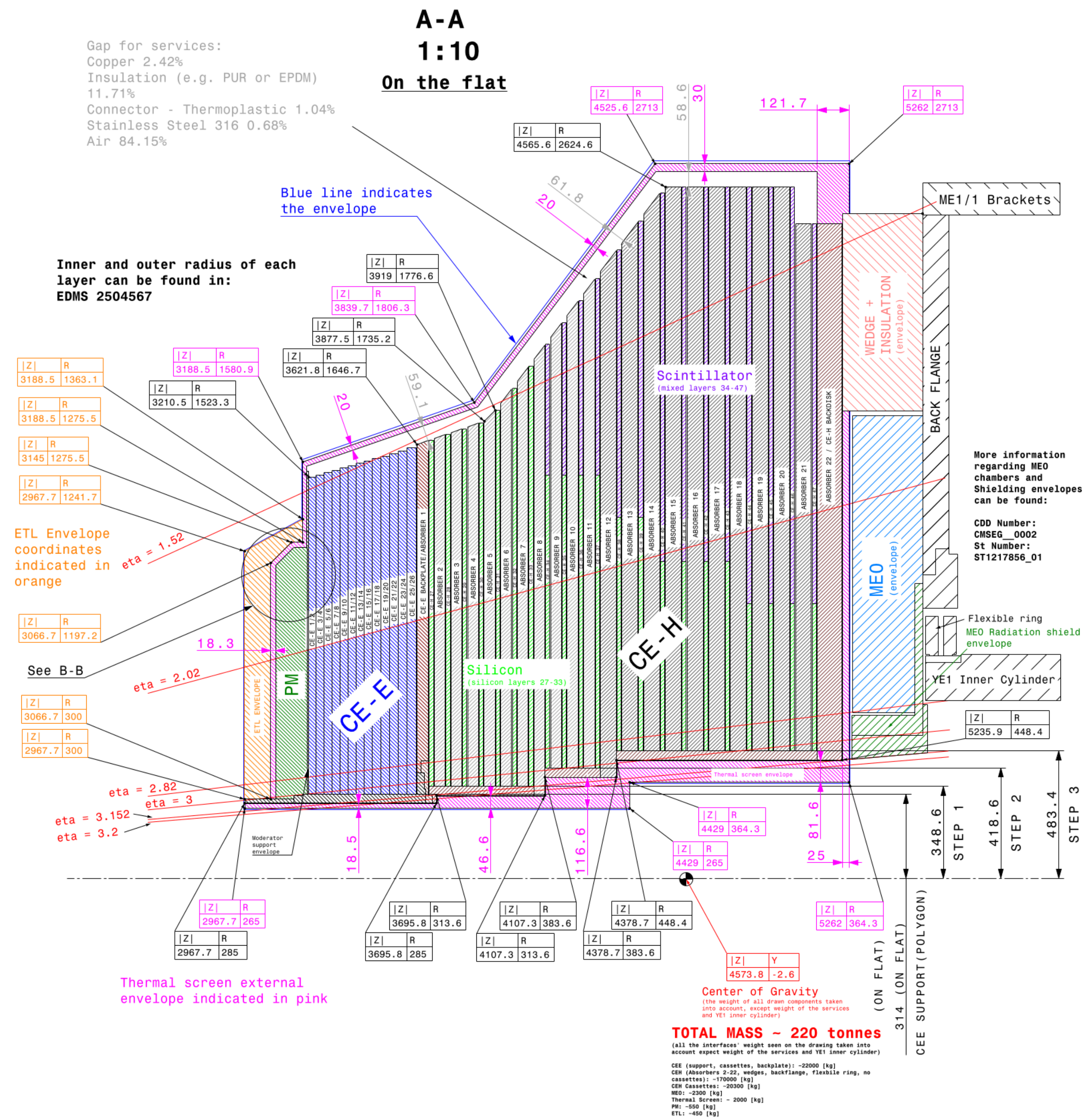
Agenda

- Reconstruction in High Granularity Calorimeter (HGCAL)
- Raw data handling in the HGCAL Detector Performance Group
  - Level-0 calibration algorithms
  - RecHitProducer with heterogeneous computing
  - Initialization of HGCAL Data Quality Monitoring
- Summary

High Granularity Calorimeter

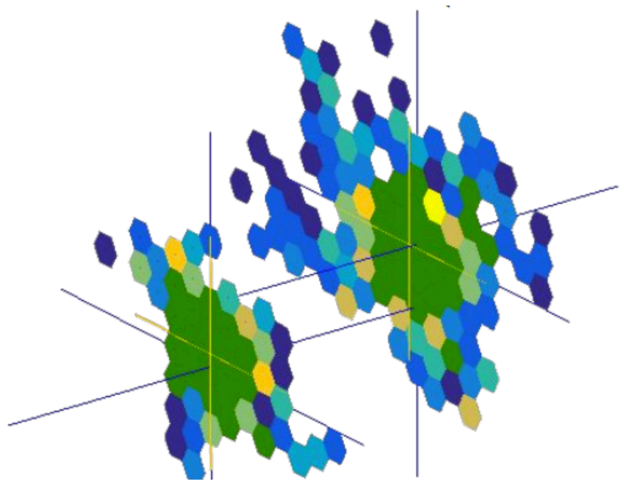
- Forward imaging calorimeter
- Electromagnetic (CEE): 26 layers
- Hadronic (CEH): 21 layers





HGCAL Reconstruction

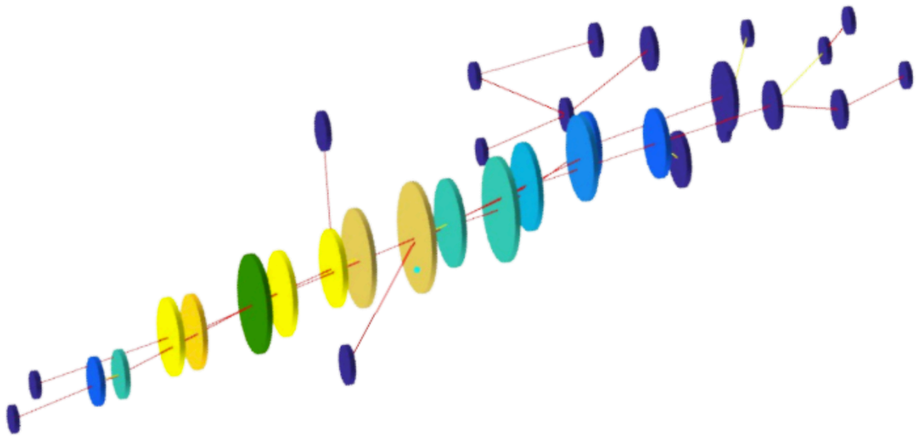
CLUstering of Energy (CLUE) algorithm



- Developed based on Imaging Algorithm
- Input **hits** and output **2D layer clusters**
- Energy density based
- Reduce dimensionality of the problem
  (10^5 hits to 10^4 layer clusters)

"The Iterative CLustering" (TICL) Framework

- Input **2D layer clusters** and output **3D objects / showers (TICL candidates)**
- Iterative algorithm
- Electromagnetic showers are easier to reconstruct
- Hadronic showers are reconstructed after EM showers

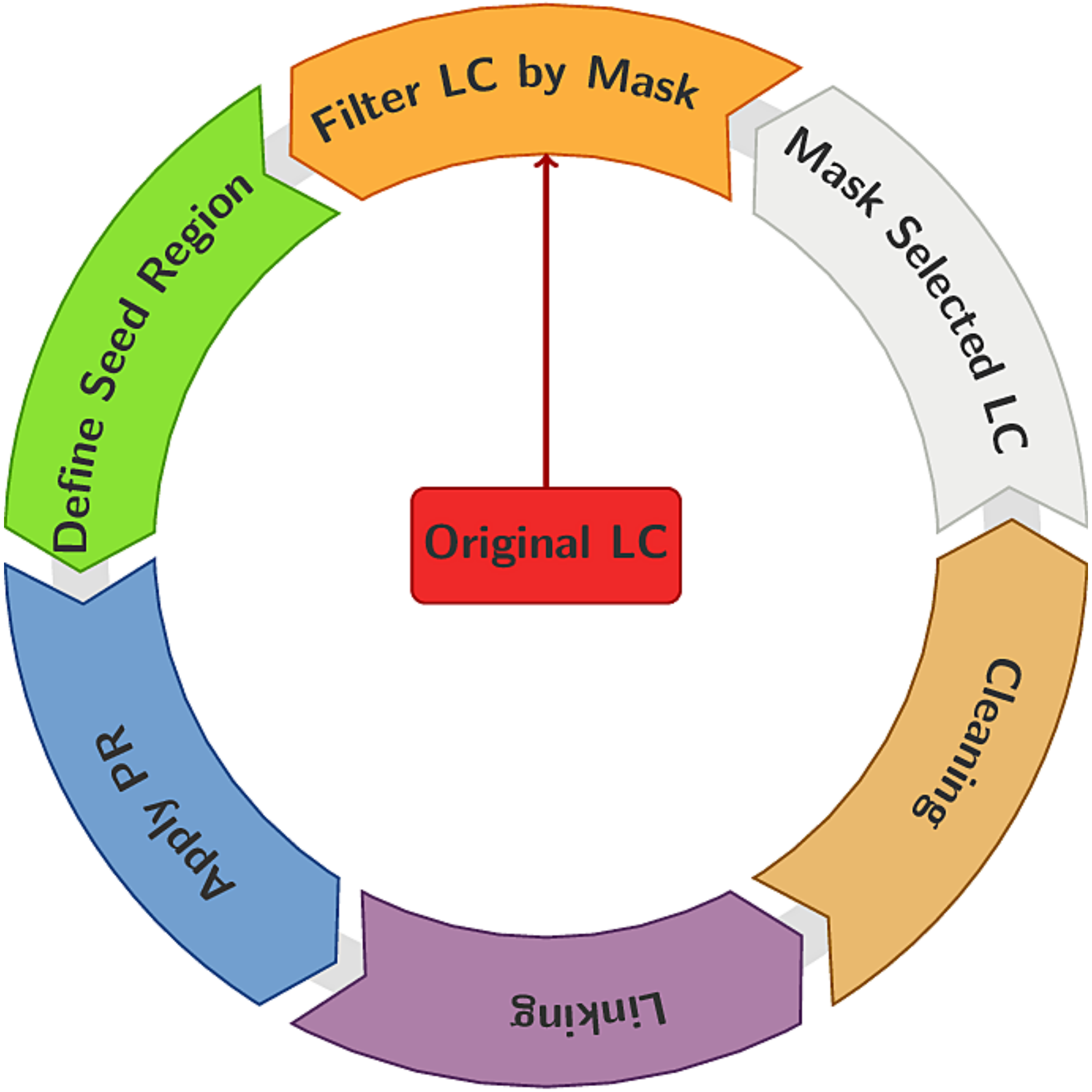


Reference: The HGCAL website and Marco Rovere's slides

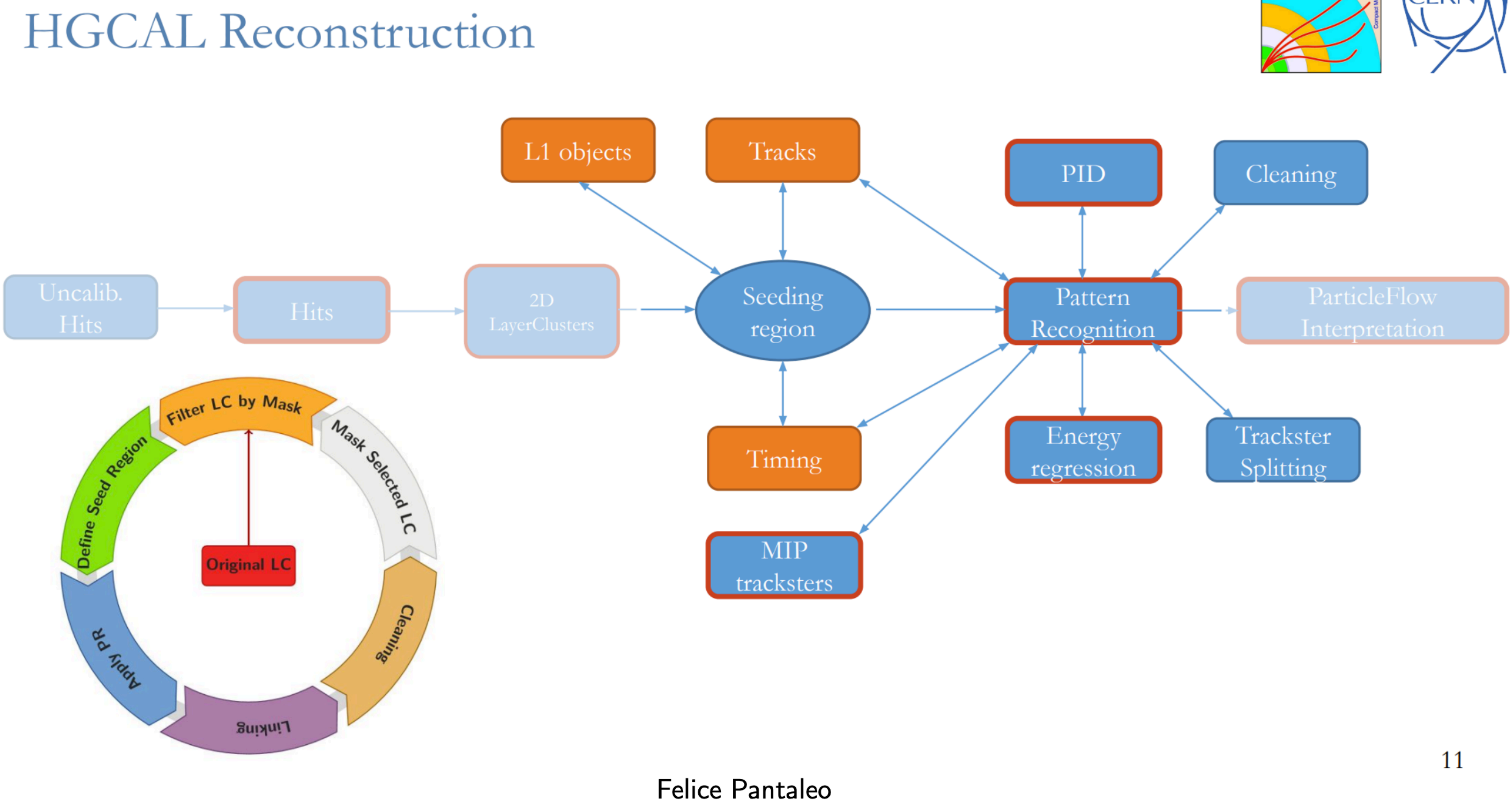
Skeleton of iteration

Five-step procedure of an iteration

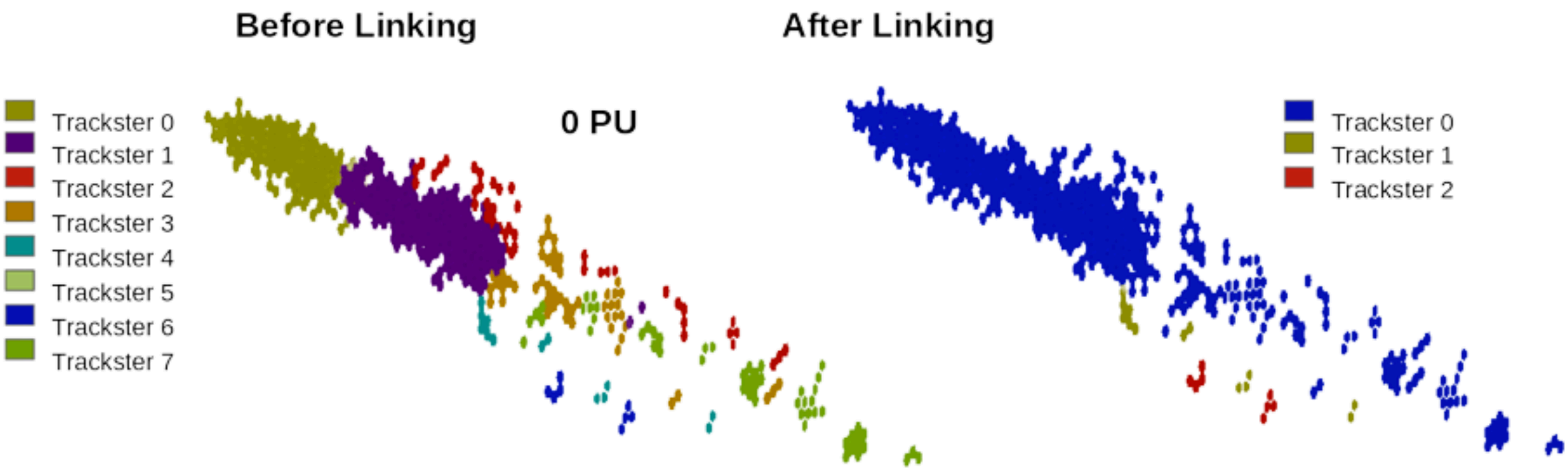
- Filter and mask layer clusters
- Define seeding region
- Pattern recognition
- Link the recognized patterns
- Cleaning and classification



TICL Framework



Tracksters

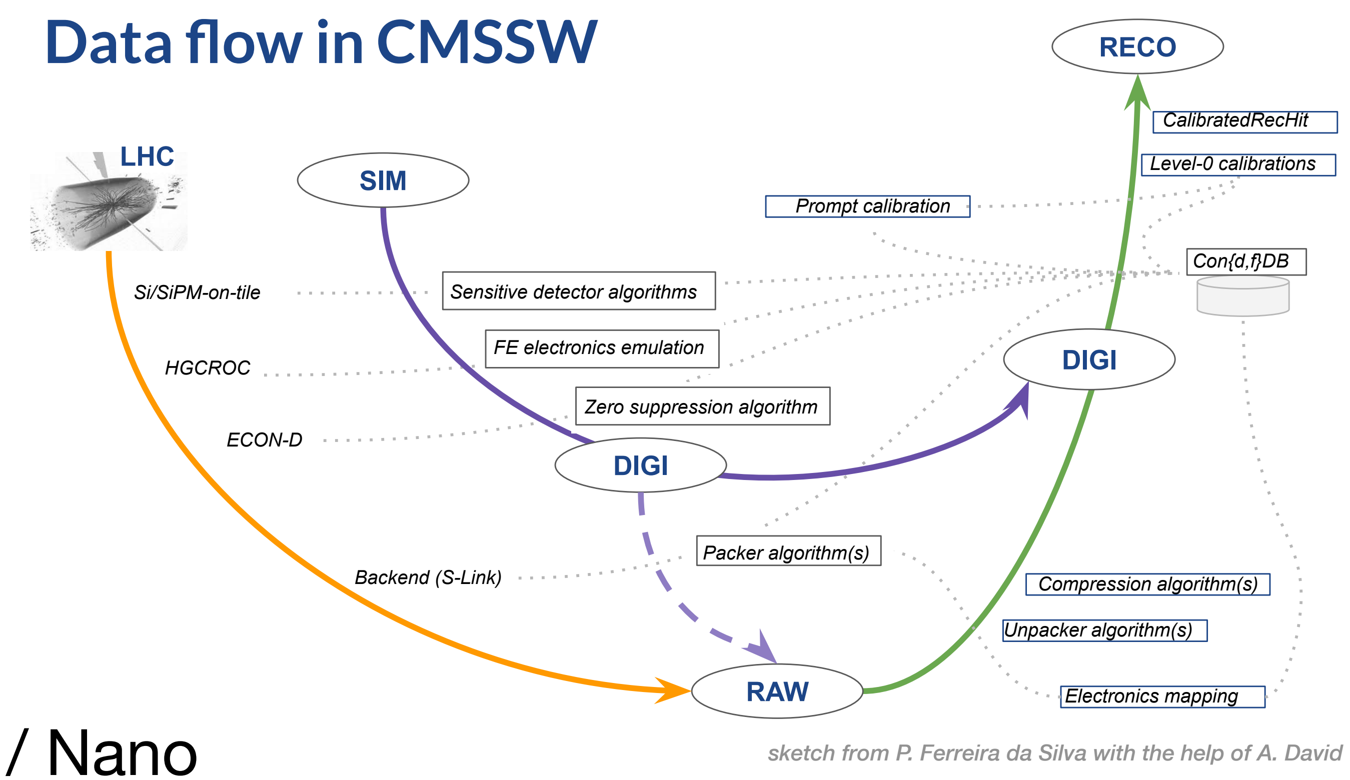

Before Linking
After Linking
0 PU
Trackster 0
Trackster 1
Trackster 2
Trackster 3
Trackster 4
Trackster 5
Trackster 6
Trackster 7
Trackster 0
Trackster 1
Trackster 2

Raw Data Handling Group

HGCAL Raw Data Handling

Phase-2 upgrade of new end-cap calorimeter

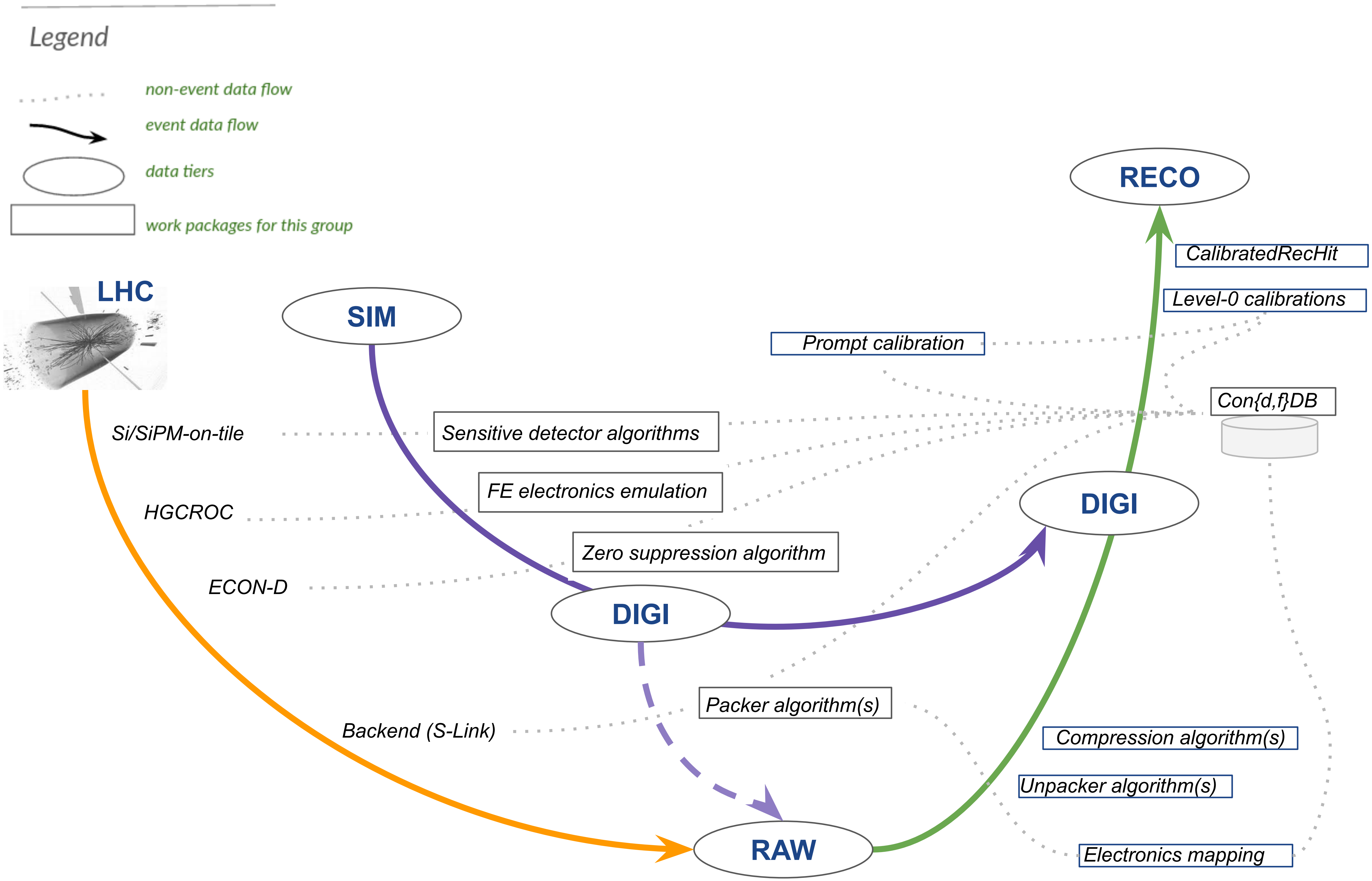
- 6 million channels → $\mathcal{O}$(700k) hits per event
- Heterogeneous computing
- Highly parallelization algorithms



HGCAL Raw Data Handling

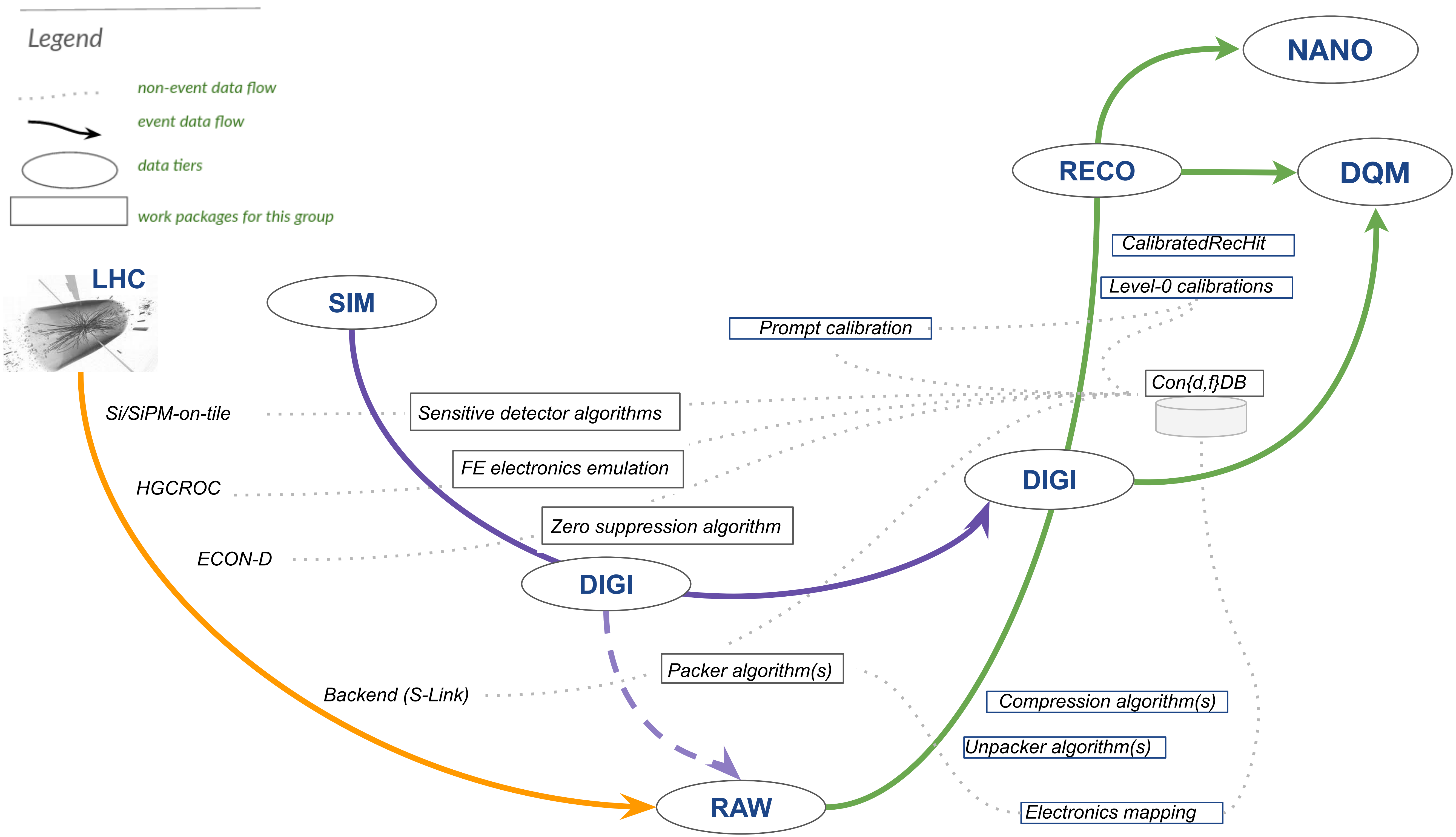
- 2022 October working group built
- 2023 Aug/Sep test-beam events
- Goal: RAW → **DIGI → RECO → DQM** / Nano
  - Level-0 calibration algorithms are developed
  - Algorithms are ported to Alpaka EDProducer for heterogeneous computing
  - HGCAL DQM service is established from scratch for the test beam activities

Data flow in CMSSW



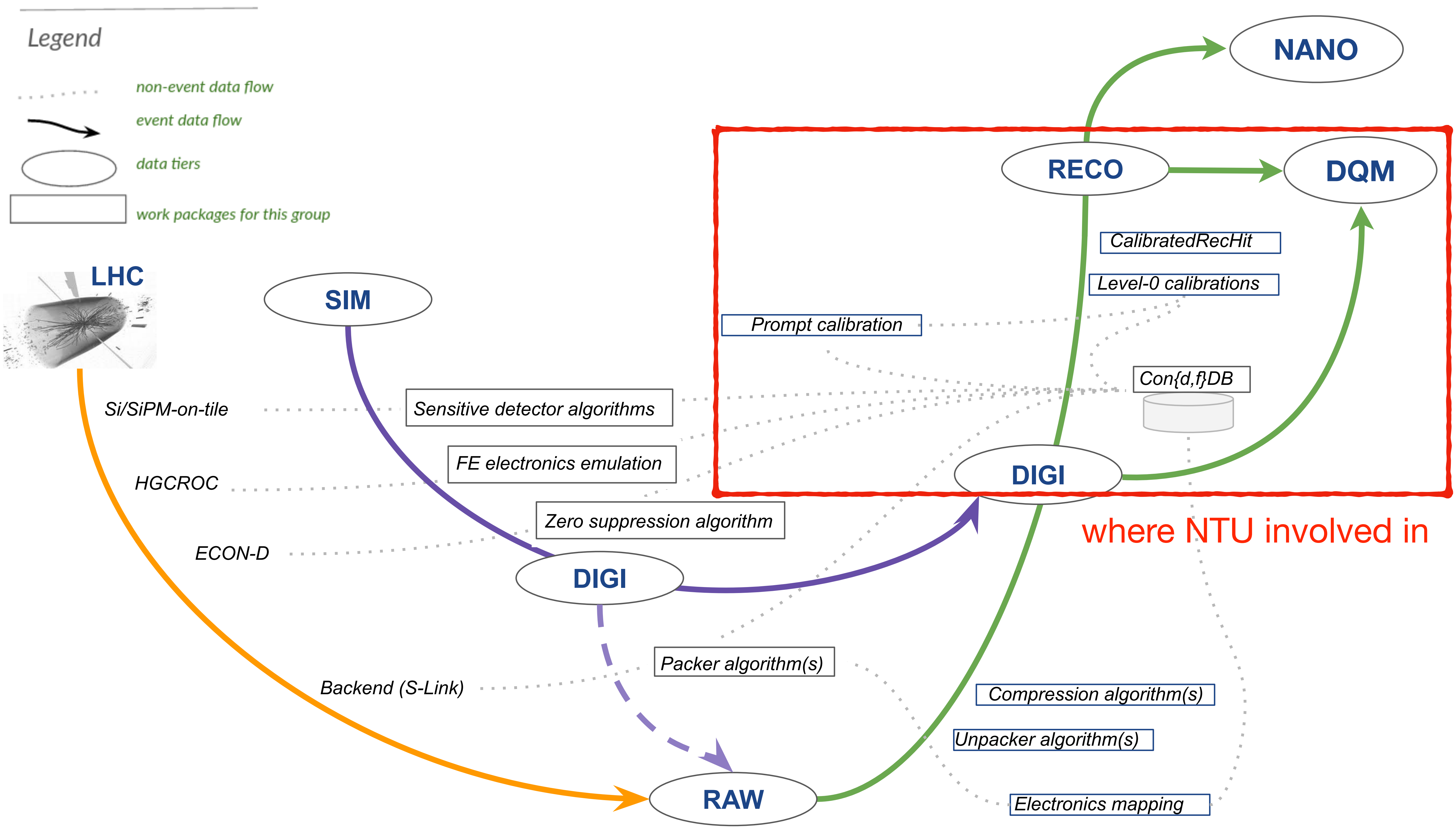
sketch from P. Ferreira da Silva with the help of A. David

Data flow in CMSSW



sketch from P. Ferreira da Silva with the help of A. David

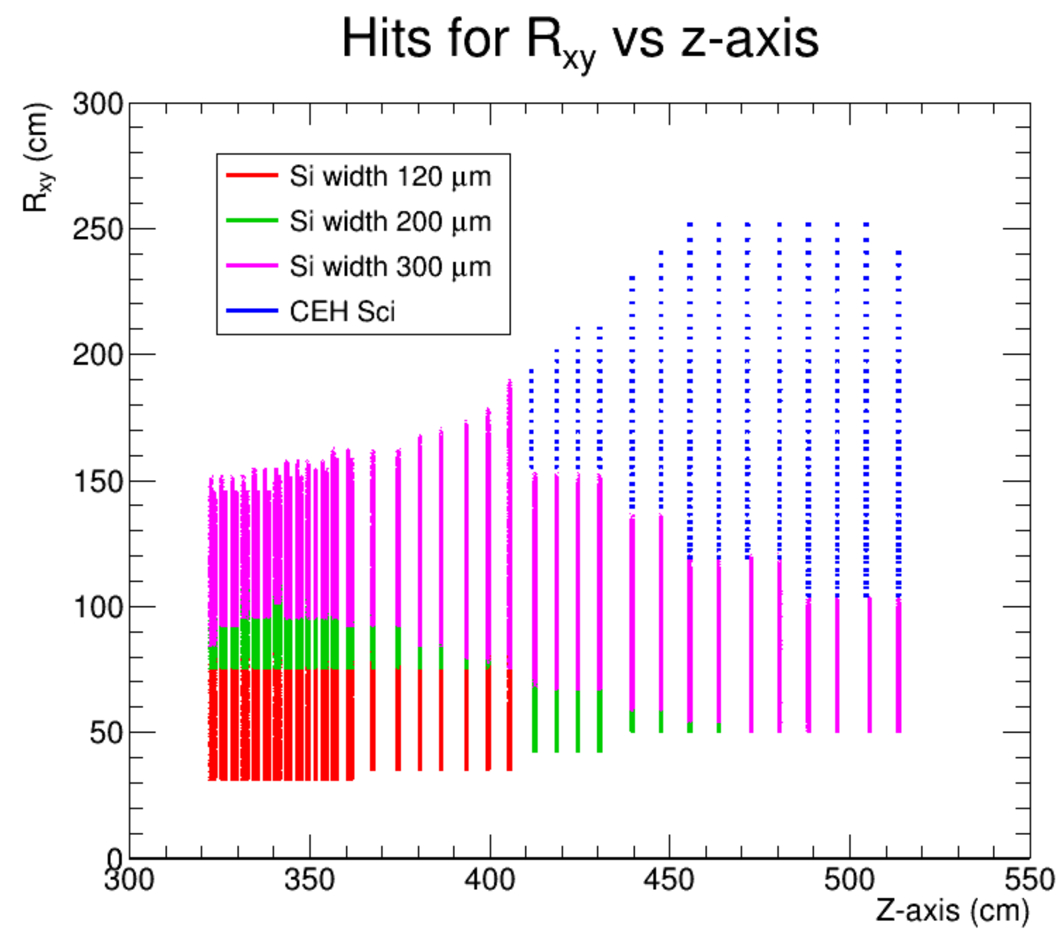
Data flow in CMSSW



sketch from P. Ferreira da Silva with the help of A. David

Calibration Algorithms

What need to be done for local reconstruction?



- The front-ends (FE) provides
  - Energy for the current bunch crossing (BX)
  - Energy for the previous bunch crossing (BX-1)
  - Time measurements for BX
- Necessary calibrations
  - Pedestal and common mode subtraction
  - Linearization
  - Energy setting (charge → MIPs → GeV)
- Some of these procedures depend on the sensor type (Silicon and SiPM-on-tile)
- Some depend on the electronics configuration (characterization mode, etc.)

HGCAL local reconstruction

Legend in the RECO chain
- Raw data
- EDProducer
- CMSSW collection
- Alpaka kernel functions
- Calibration data/method

What need to be done for local reconstruction?

Raw data

Unpacker
drop non-connected/bad channels
assign DetId

DIGIs
in 128-bit words
Several ECOND packets
ToA, TcTp, ADC (BX-1), ADC/ToT
identified with DetId

RecHit Producer
apply level-0 corrections
assign RecHit status flags
assign RecHit correction level

Rechit collection
Energy in fC / MIP / GeV
Time in ns
Data quality and correction-level flags
identified with DetId

- HGCAL prompt calibration loop

Energy calibration

Pedestal estimation	Pedestals - 1 param per channel per gain
Common mode studies	Common mode correction - ≥1 param per ROC per gain
BX-1 correction	BX-1 correction - 1 param per ROC per gain
Charge non-linearity study	Gain linearization - 2 constants per ROC per gain
SiPM non-linearity study	SiPM linearization - functional form per tile
MIP distribution fit	ADC to MIP - 1 param per channel
EM scale studies	EM scale - eγ, $\pi^0 \to \gamma\gamma$, $J/\psi \to \mu\mu$, $Z \to ee$

Time calibration

ToA time slew (= time walk)	Time slew - 1 param per ROC per gain
Time zero-offset	Time zero-offset - 1 param per 60 degree per run
TDC non-linearity	ToA integral non-linearity (INL) - 2 const. per ROC per gain

What need to be done for local reconstruction?

Legend in the RECO chain
- Raw data
- EDProducer
- CMSSW collection
- Alpaka kernel functions
- Calibration data/method

Raw data

Unpacker
drop non-connected/bad channels
assign DetId

DIGIs
in 128-bit words
Several ECOND packets
ToA, TcTp, ADC (BX-1), ADC/ToT
identified with DetId

RecHit Producer
apply level-0 corrections
assign RecHit status flags
assign RecHit correction level

Rechit collection
Energy in fC / MIP / GeV
Time in ns
Data quality and correction-level flags
identified with DetId

- HGCAL prompt calibration loop

Energy calibration

Pedestal estimation	Pedestals - 1 param per channel per gain
Common mode studies	Common mode correction - ≥1 param per ROC per gain
BX-1 correction	BX-1 correction - 1 param per ROC per gain
Charge non-linearity study	Gain linearization - 2 constants per ROC per gain
SiPM non-linearity study	SiPM linearization - functional form per tile
MIP distribution fit	ADC to MIP - 1 param per channel
EM scale studies	EM scale - eγ, $\pi^0 \to \gamma\gamma$, $J/\psi \to \mu\mu$, $Z \to ee$

Time calibration

ToA time slew (= time walk)	Time slew - 1 param per ROC per gain
Time zero-offset	Time zero-offset - 1 param per 60 degree per run
TDC non-linearity	ToA integral non-linearity (INL) - 2 const. per ROC per gain

Running statistics for subtractions

Considerations

- Prompt calibration @ CMS P5
- Memory consumption

Table: comparison of memory consumption

	Total heap usage (kB)
Single TH2D	84.7
Running statistics	2.95

Running statistics

- Record only statistics and update them event by event
- Evaluate pedestal and common-mode parameters
- Implemented in cms-hgcal/cmssw

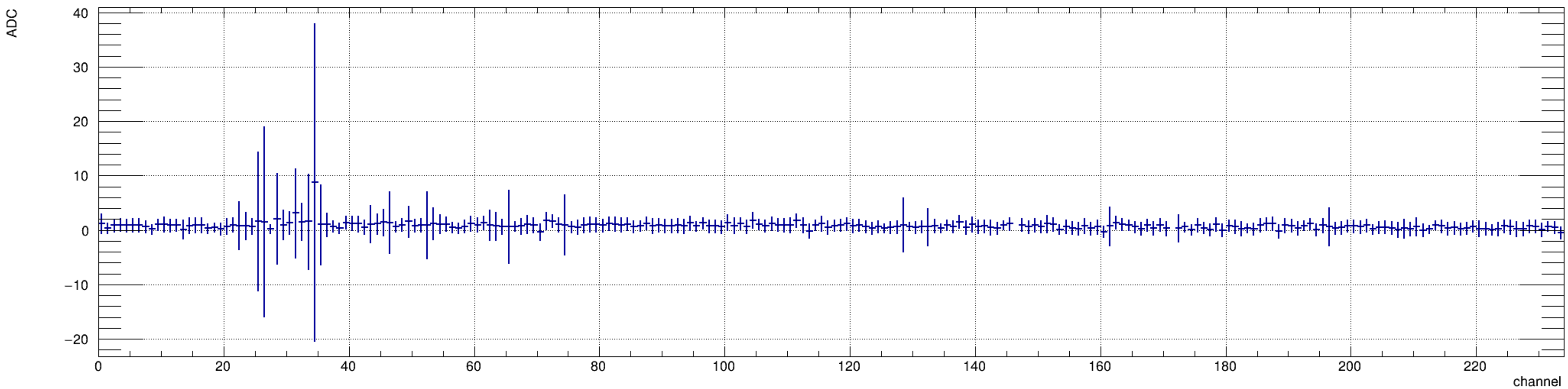
Mean: $$\bar{x}_{i+1} = \frac{n}{n+1}\bar{x}_i + \frac{1}{n+1}x_{i+1}$$

Variance: $$V_{i+1} = \frac{n}{n+1}V_i + \frac{n}{n+1}\bar{x}_i^2 - \bar{x}_{i+1}^2 + \frac{1}{n+1}x_{i+1}^2$$

Pedestal

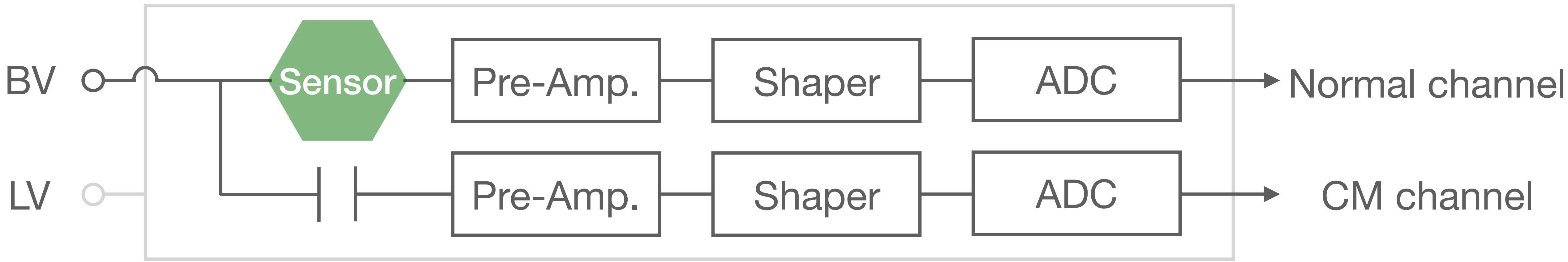
- Level of electronic response when there is no signal
- Use mean value as an estimate for prompt evaluation
- Sample: 2022 test beam data

Beam run ADC after subtraction of mean pedestal derived from pedestal run



Common mode noise

- Caused by fluctuations from bias voltage or low voltage
- Bias voltage (BV or HV): voltage applied to sensors
- Low voltage (LV): power supply of chips

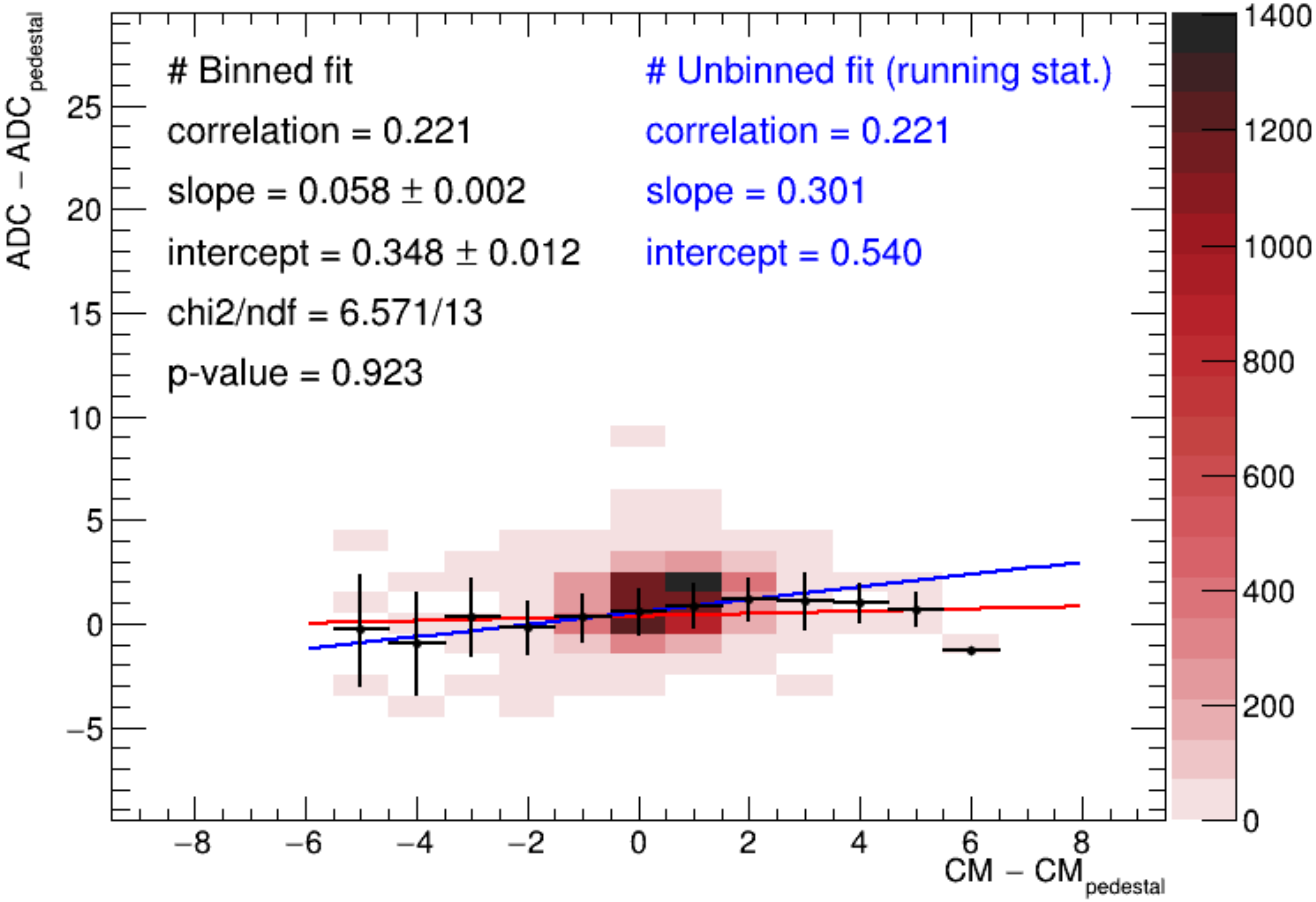


Simplified from Geliang Liu's sketch

Common Mode noise subtraction (2/4)

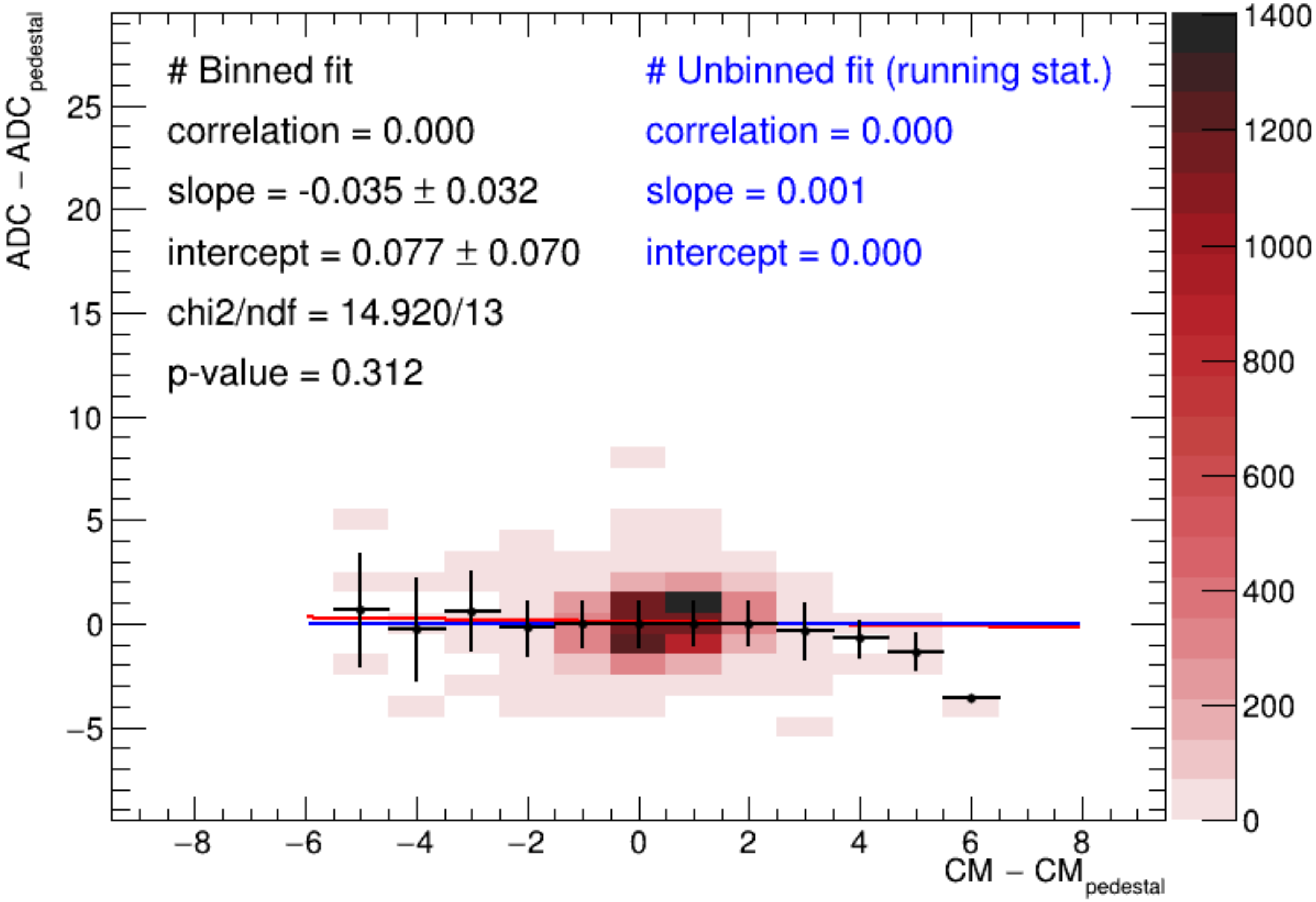
Common mode noise

- Caused by fluctuations from bias voltage or low voltage
- Correlation exists between a normal channel and CM channel because of the noise



Common mode noise

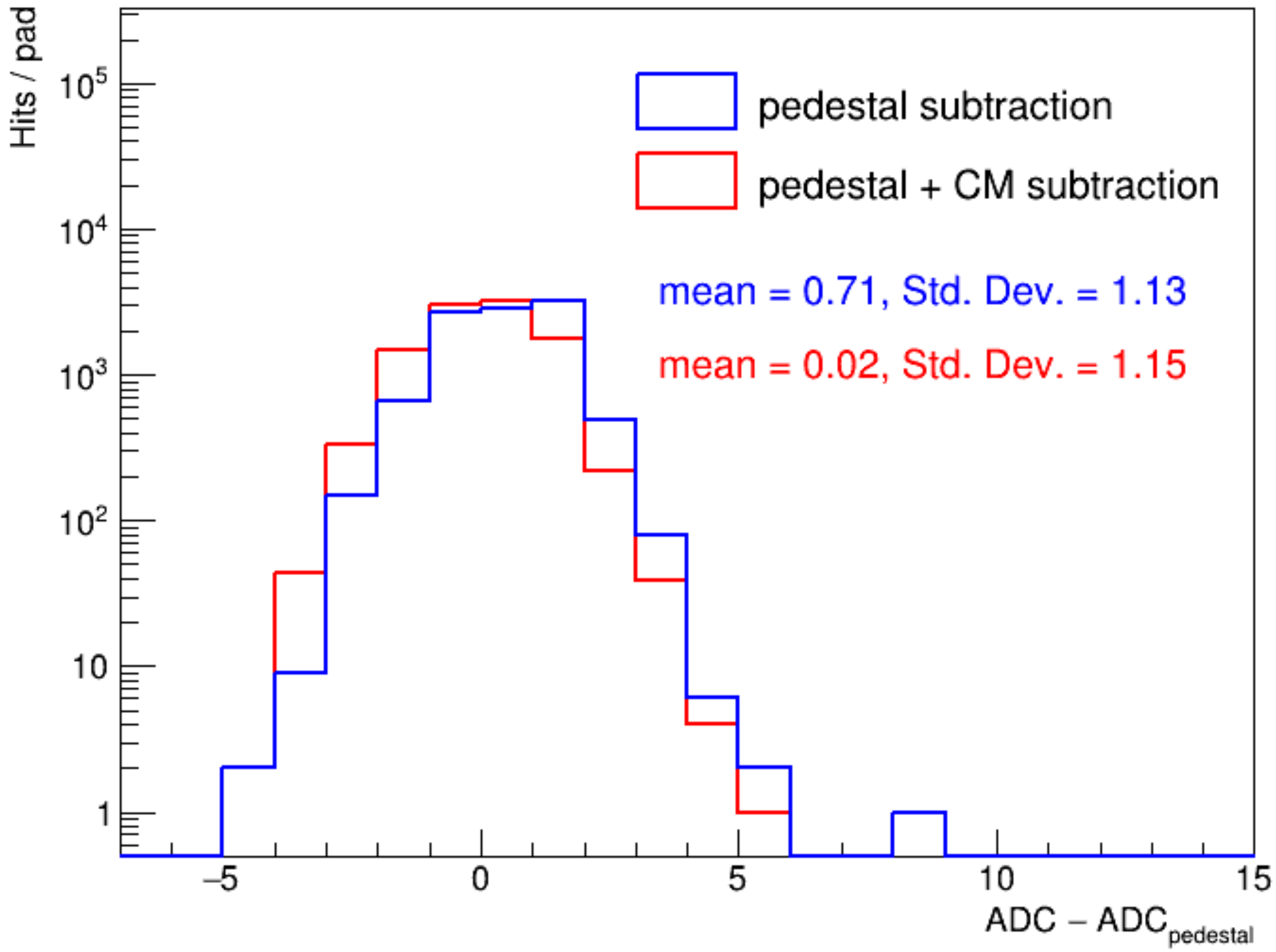
- Caused by fluctuations from bias voltage or low voltage
- CM noise is removed by decorrelating the normal channel and the CM channel



Common Mode noise subtraction (4/4)

Common mode noise

- Caused by fluctuations from bias voltage or low voltage
- Comparison before and after the CM subtraction
- Other sophisticated methods for CM noise removal are under studied by experts

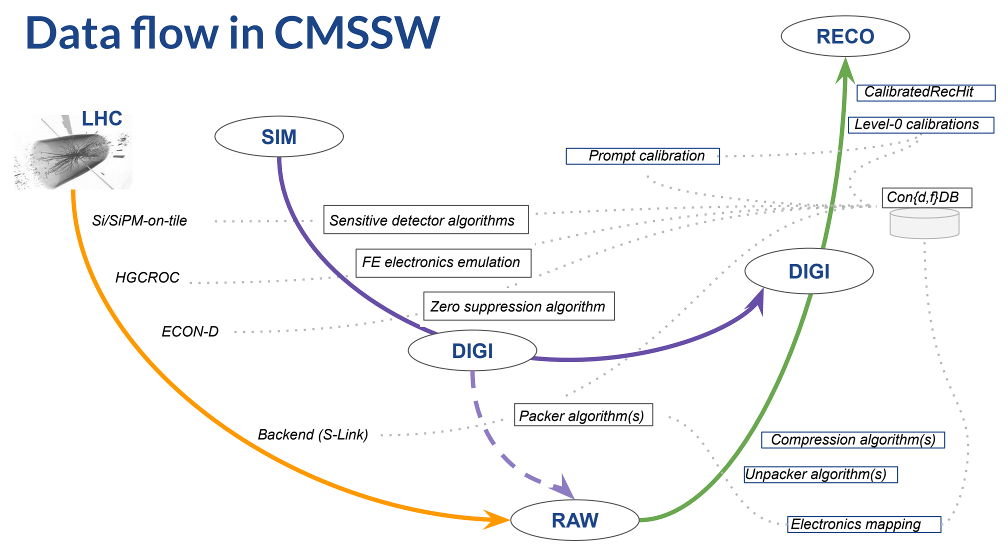


Heterogeneous Computing

Heterogeneous computing

Phase-2 upgrade of new end-cap calorimeter

- 6 million channels → $\mathcal{O}$(700k) hits per event
- Heterogeneous computing → Use the Alpaka library!

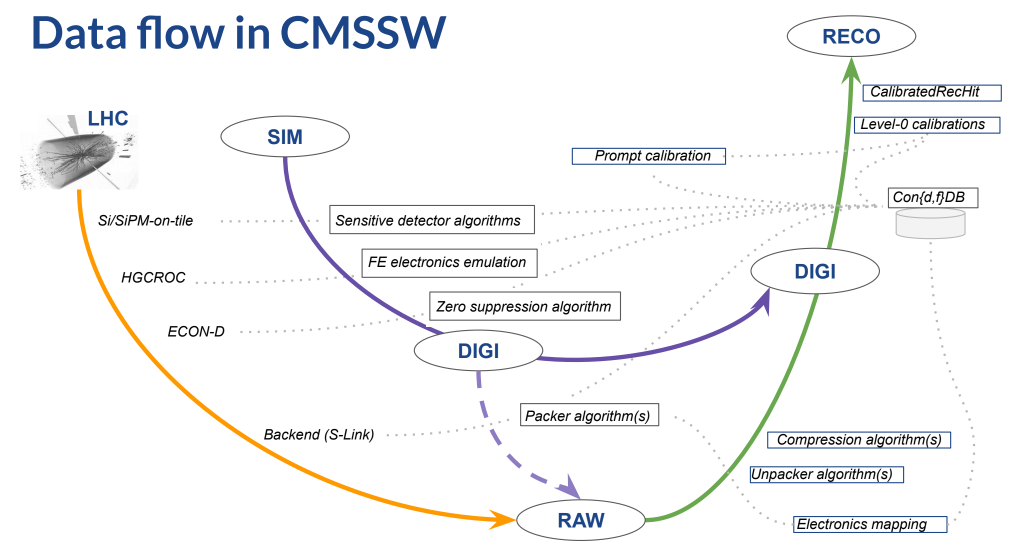


sketch from P. Ferreira da Silva with the help of A. David

Heterogeneous computing

Phase-2 upgrade of new end-cap calorimeter

- 6 million channels → $\mathcal{O}$(700k) hits per event
- Heterogeneous computing → Use the Alpaka library!



sketch from P. Ferreira da Silva with the help of A. David

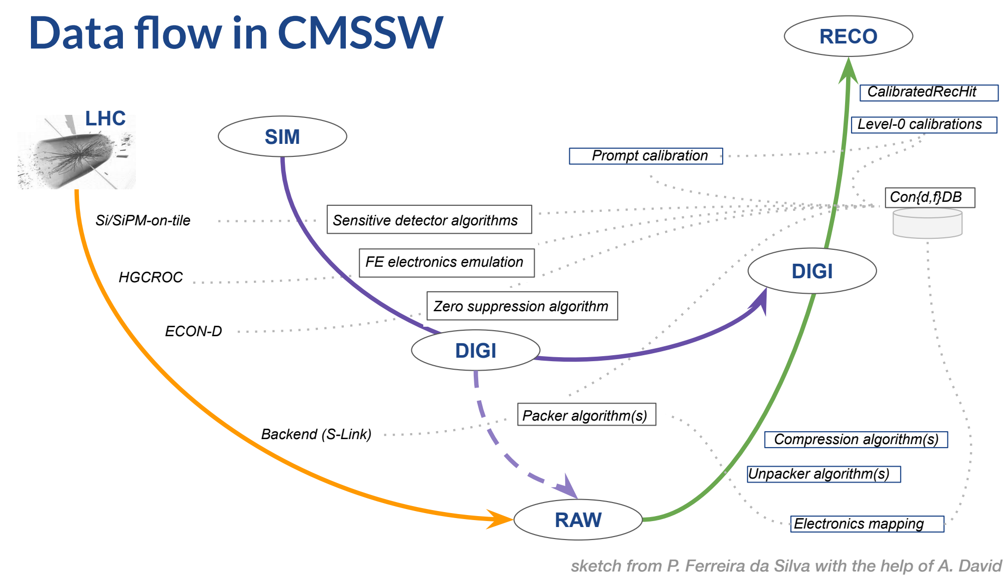
What is Alpaka?

- **A**bstraction **L**ibrary for **Pa**rallel **K**ernel **A**cceleration
- "Aim to provide **performance portability** across accelerators through the abstraction of the underlying levels of parallelism"
- Data format is based on structure of arrays (SoAs)

Heterogeneous computing

Phase-2 upgrade of new end-cap calorimeter

- 6 million channels → $\mathcal{O}$(700k) hits per event
- Heterogeneous computing → Use the Alpaka library!

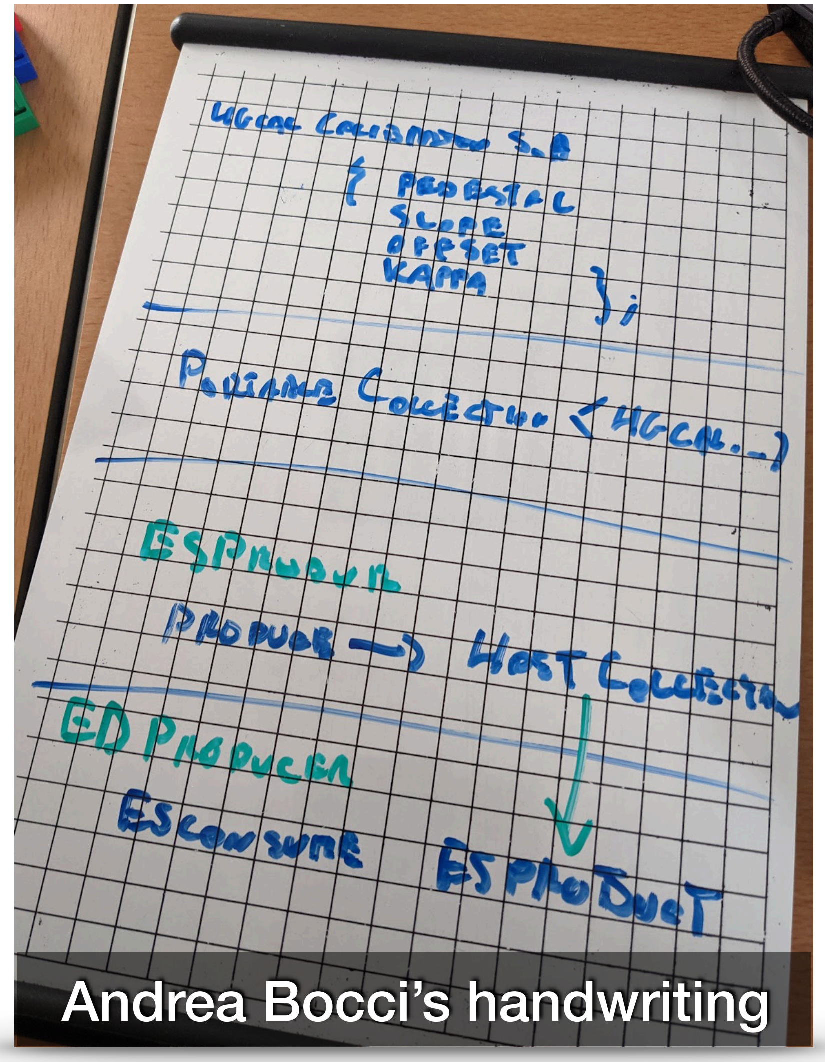


What is Alpaka?

- **A**bstraction **L**ibrary for **Pa**rallel **K**ernel **A**cceleration
- "Aim to provide **performance portability** across accelerators through the abstraction of the underlying levels of parallelism"
- Data format is based on structure of arrays (SoAs)

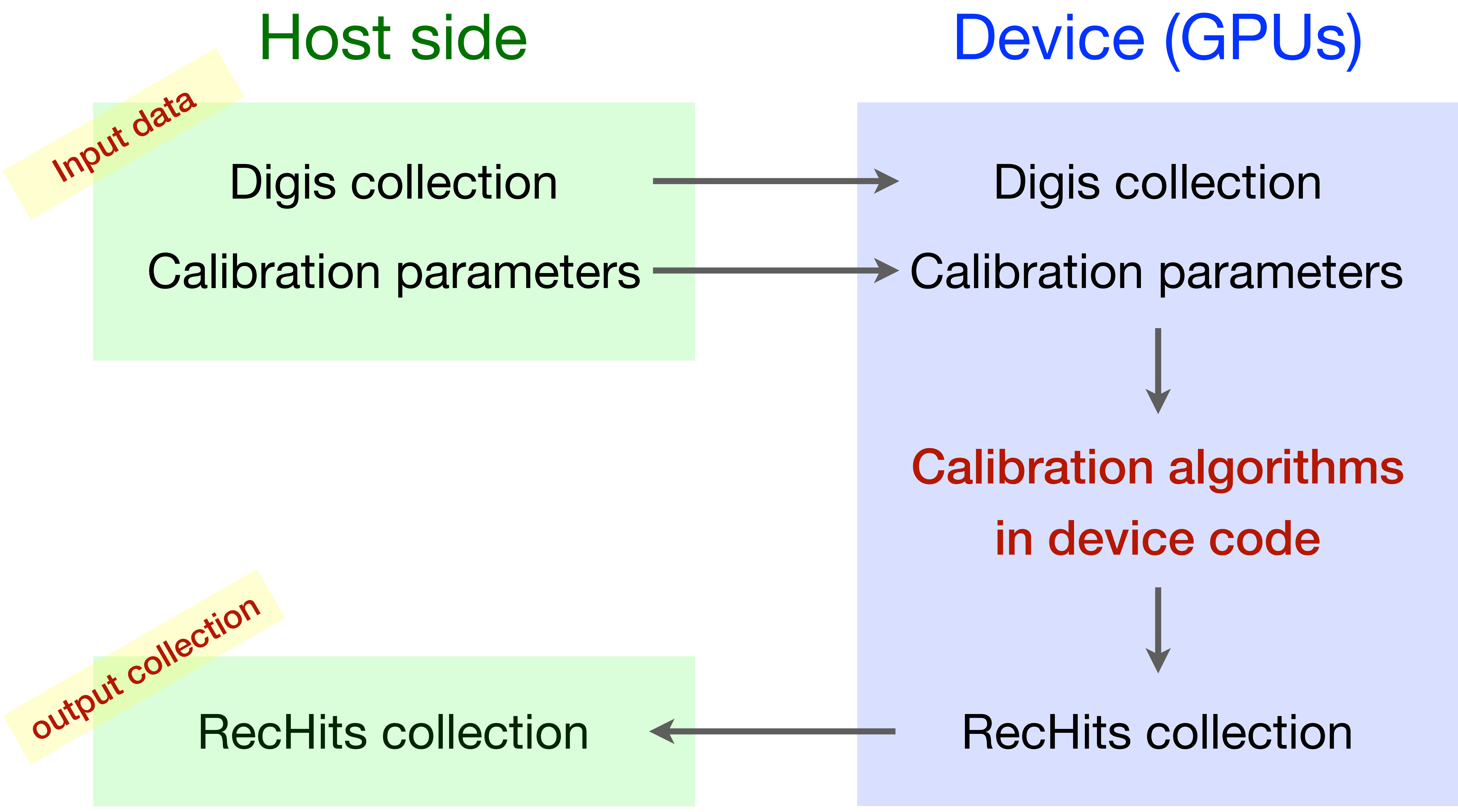
Alpaka modules in CMSSW "GPU framework"

- Alpaka ESProducer → passing conditions
- Alpaka EDProducer → produce RecHits



Andrea Bocci's handwriting

Data transfer & GPU Kernels



CMSSW Setup

CMSSW

- Based on 13_2_0_pre2 (link)

Alpaka ESProducer

- Calibration parameters in SoA

Alpaka EDProducer (RecHitProducer)

- Input: Digis in SoA layout
- Output: RecHits in SoA layout

```
DataFormats/HGCalDigi/
|-- BuildFile.xml
|-- interface
|  |-- HGCalDigiHostCollection.h
|  |-- HGCalDigiSoA.h
|  `-- alpaka
|      `-- HGCalDigiDeviceCollection.h
`-- src
   |-- alpaka
   |  |-- classes_cuda.h
   |  |-- classes_cuda_def.xml
   |  |-- classes_rocm.h
   |  `-- classes_rocm_def.xml
   |-- classes.h
   `-- classes_def.xml
```

```
DataFormats/HGCalRecHit/
|-- BuildFile.xml
|-- interface
|  |-- HGCalRecHitHostCollection.h
|  |-- HGCalRecHitSoA.h
|  `-- alpaka
|      `-- HGCalRecHitDeviceCollection.h
`-- src
   |-- alpaka
   |  |-- classes_cuda.h
   |  |-- classes_cuda_def.xml
   |  |-- classes_rocm.h
   |  `-- classes_rocm_def.xml
   |-- classes.h
   `-- classes_def.xml
```

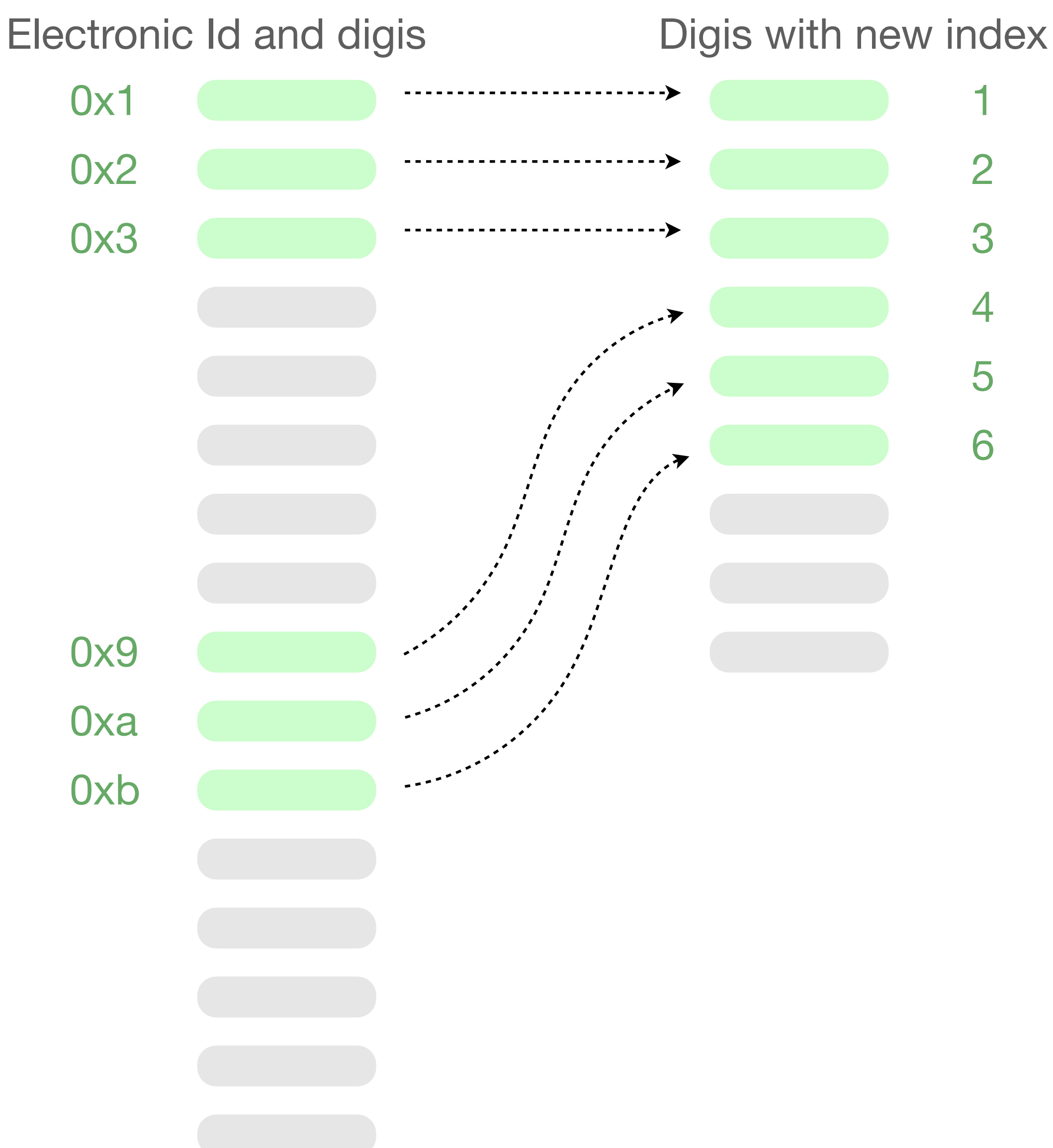
Alpaka related code for HGCAL raw data handling

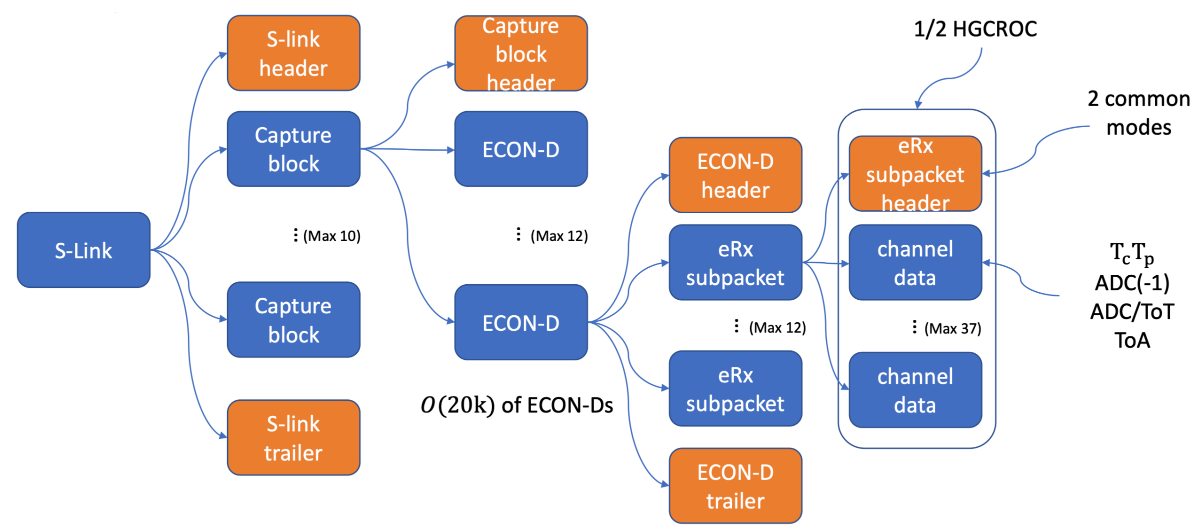
```
RecoLocalCalo/HGCalRecAlgos/
|-- BuildFile.xml
|-- interface
|  |-- HGCalCalibrationParameterHostCollection.h
|  |-- HGCalCalibrationParameterProvider.h
|  |-- HGCalCalibrationParameterSoA.h
|  `-- alpaka
|      `-- HGCalCalibrationParameterDeviceCollection.h
|-- plugins
|  |-- BuildFile.xml
|  `-- alpaka
|      |-- HGCalRecHitCalibrationAlgorithms.dev.cc
|      |-- HGCalRecHitCalibrationAlgorithms.h
|      |-- HGCalRecHitProducer.cc
|      `-- HGCalRecHitCalibrationESProducer.cc
`-- src
   |-- ES_HGCalCalibrationParameter.cc
   `-- alpaka
      `-- ES_HGCalCalibrationParameter.cc
```

A C++ structure resides in the SoA scalar

Contributor of the dense map: Yulun Miao

- Introduce a “dense map” for the indices of calibration parameters
- Size of portable collection is determined from config parameters (e.g. max sizes of capture blocks, econds, etc.)






```
struct HGCalCalibrationParameterProviderConfig {
 …
  constexpr uint32_t denseMap(uint32_t ElectronicsID) const{
    uint32_t sLink = ((ElectronicsID >> kFEDIDShift) & kFEDIDMask);
    uint32_t captureBlock = ((ElectronicsID >> kCaptureBlockShift) & kCaptureBlockMask);
    uint32_t econd = ((ElectronicsID >> kECONDIdxShift) & kECONDIdxMask);
    uint32_t eRx = ((ElectronicsID >> kECONDeRxShift) & kECONDeRxMask);
    uint32_t channel = ((ElectronicsID >> kHalfROCChannelShift) & kHalfROCChannelMask);
    uint32_t rtn = sLink * sLinkCaptureBlockMax + captureBlock;
    rtn = rtn * captureBlockECONDMax + econd;
    rtn = rtn * econdERXMax + eRx;
    rtn = rtn * erxChannelMax + channel;
    return rtn;
  }
};
```

Source: RecoLocalCalo/HGCalRecAlgos/interface/HGCalCalibrationParameterProvider.h

Building blocks to pass parameters on GPUs

Contributor of the idea: Andrea Bocci

Alpaka ESProducer

- SoA
- Portable collections
- ESProducer
- Declaration to framework
- Load in RecHit Producer
- Pass to calibration kernel

```
#include "RecoLocalCalo/HGCalRecAlgos/interface/HGCalCalibrationParameterProvider.h"

namespace hgcalrechit {

  // Generate structure of arrays (SoA) layout with RecHit dataformat
  GENERATE_SOA_LAYOUT(HGCalCalibrationParameterSoALayout,
                      SOA_SCALAR(HGCalCalibrationParameterProviderConfig, config),
                      SOA_COLUMN(float, pedestal),
                      SOA_COLUMN(float, CM_slope),
                      SOA_COLUMN(float, CM_offset),
                      SOA_COLUMN(float, BXm1_kappa)
  )
  using HGCalCalibParamSoA = HGCalCalibrationParameterSoALayout<>;

}  // namespace hgcalrechit
```

Implementation

- 4da2d3b Add SoA and portable collections for calib parameters
- 3f87f22 Add an alpaka ESProducer & pass calibration parameters to kernels

Considerations of SoA for calibration parameters

- Calibration parameters → SoA column of float
- Map between electronics id and calibration parameters → SoA scalar for **a c++ structure**

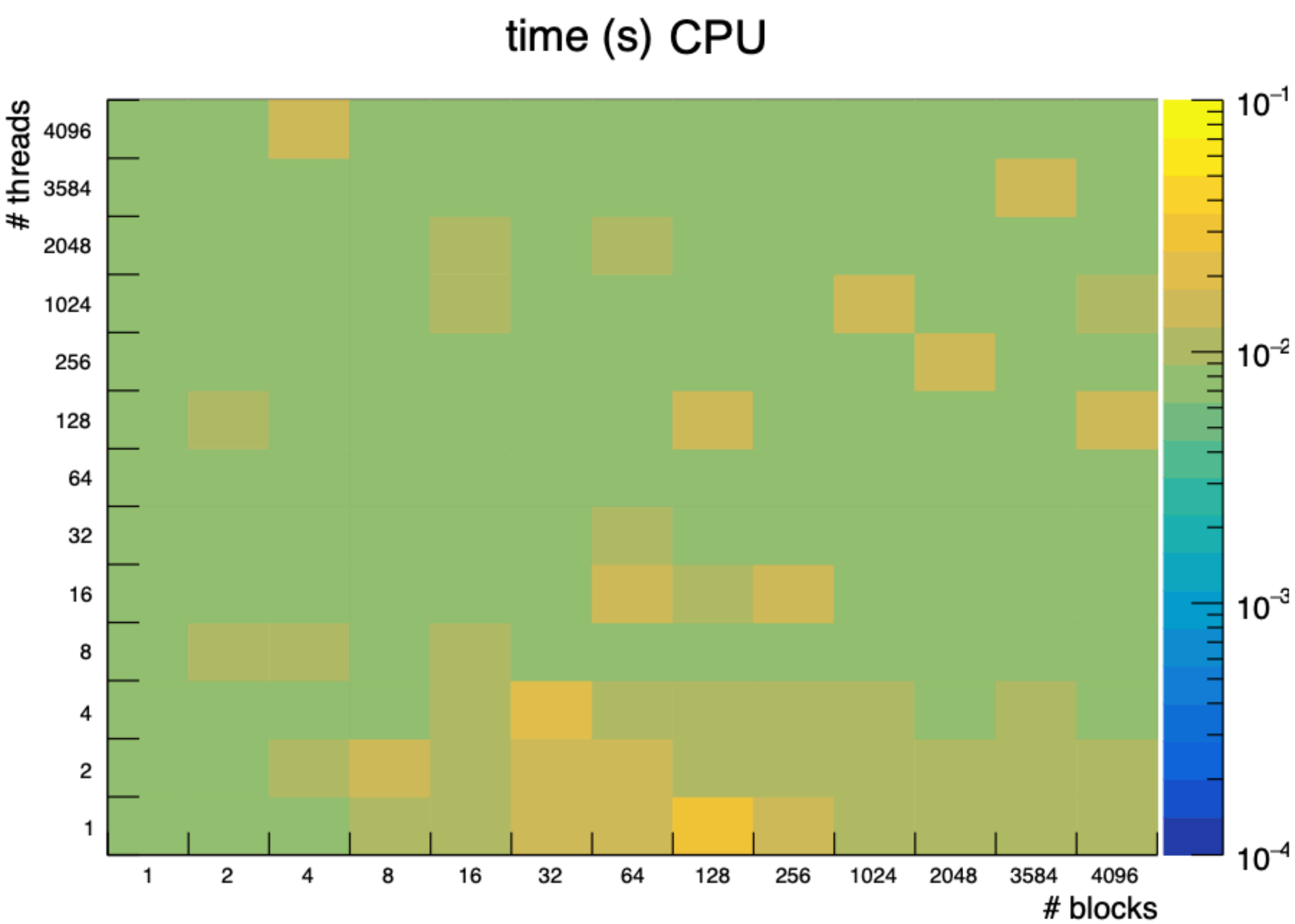
Contributor: Jeremi Niedziela

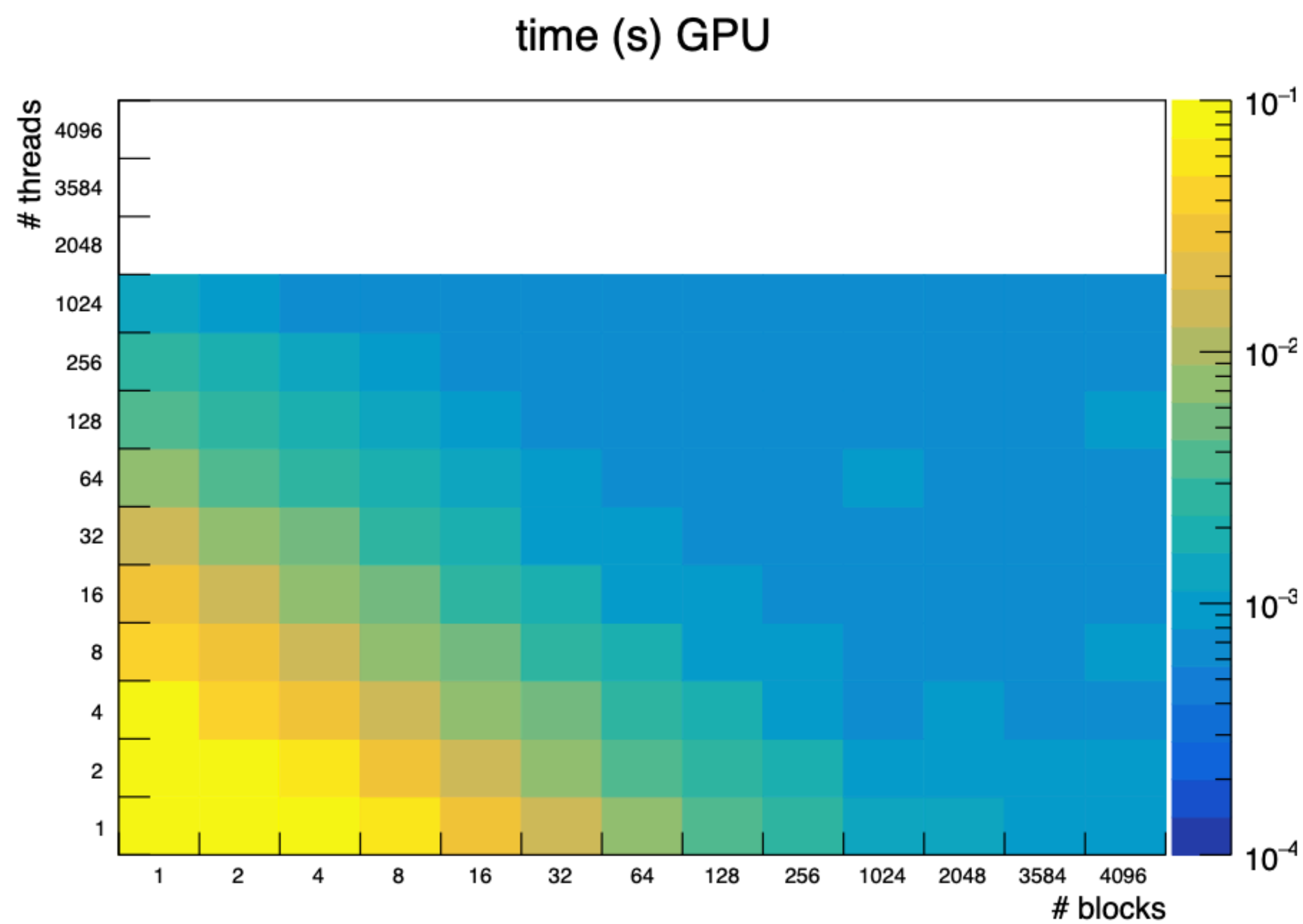
Machines

- CPU: Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz (10 cores, 20 threads)
- GPU: Tesla P100-PCIE-16GB (3584 CUDA cores)

Sample & algorithms

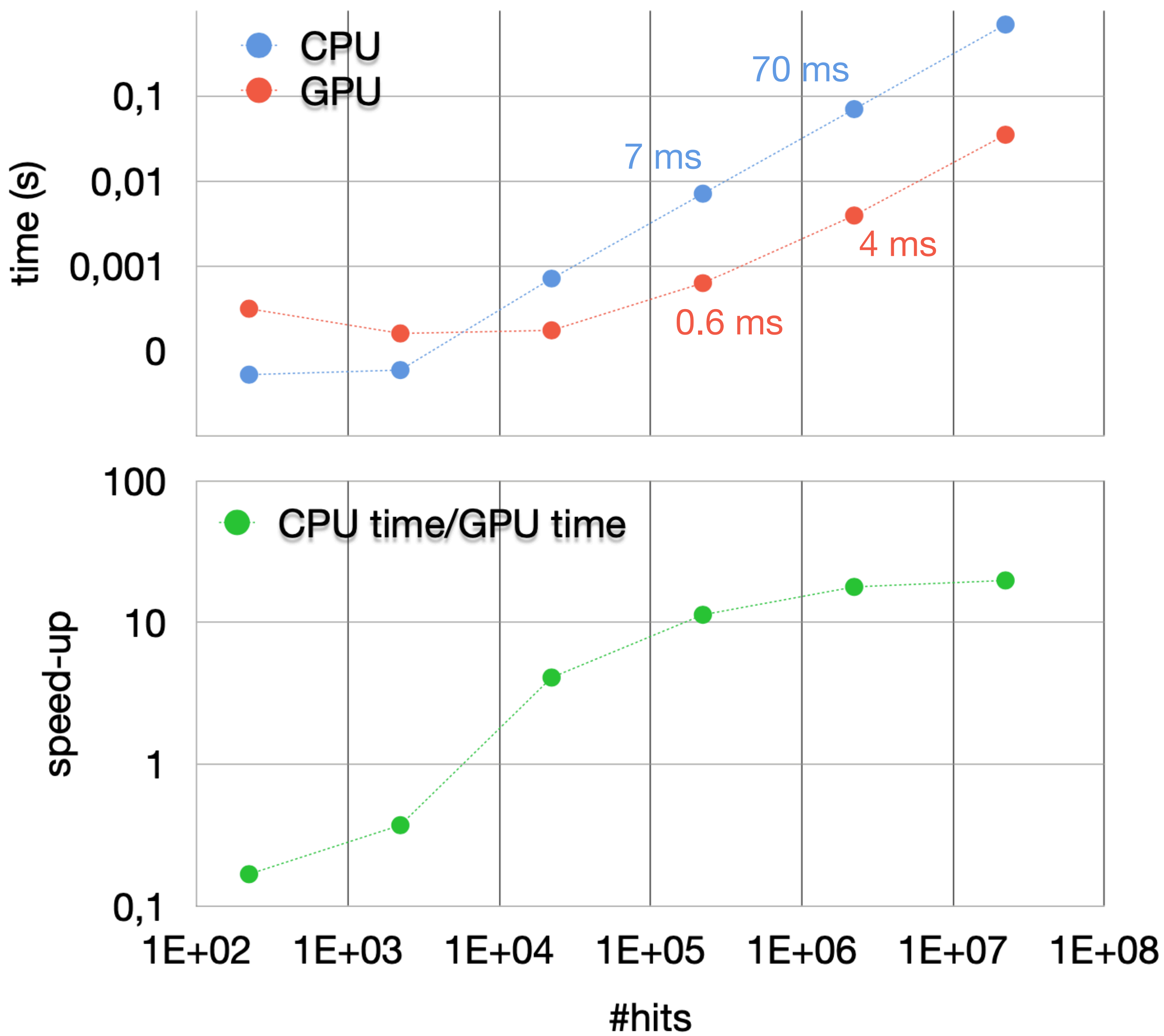
- Data from lab test with 200k hits
- Pedestal & common-mode noise subtractions





Performance with different number of hits

Contributor: Jeremi Niedziela



CPU vs. GPU

- Running the same algorithms with different number of hits
- Comments
  - GPU is rather flat in the beginning → dominated by copying data between CPU and GPU
  - At around 10k hits, GPU starts to outperform CPU
  - At a few million hits, GPU shows an order of magnitude faster!

HGCAL DQM

Beginning of the story

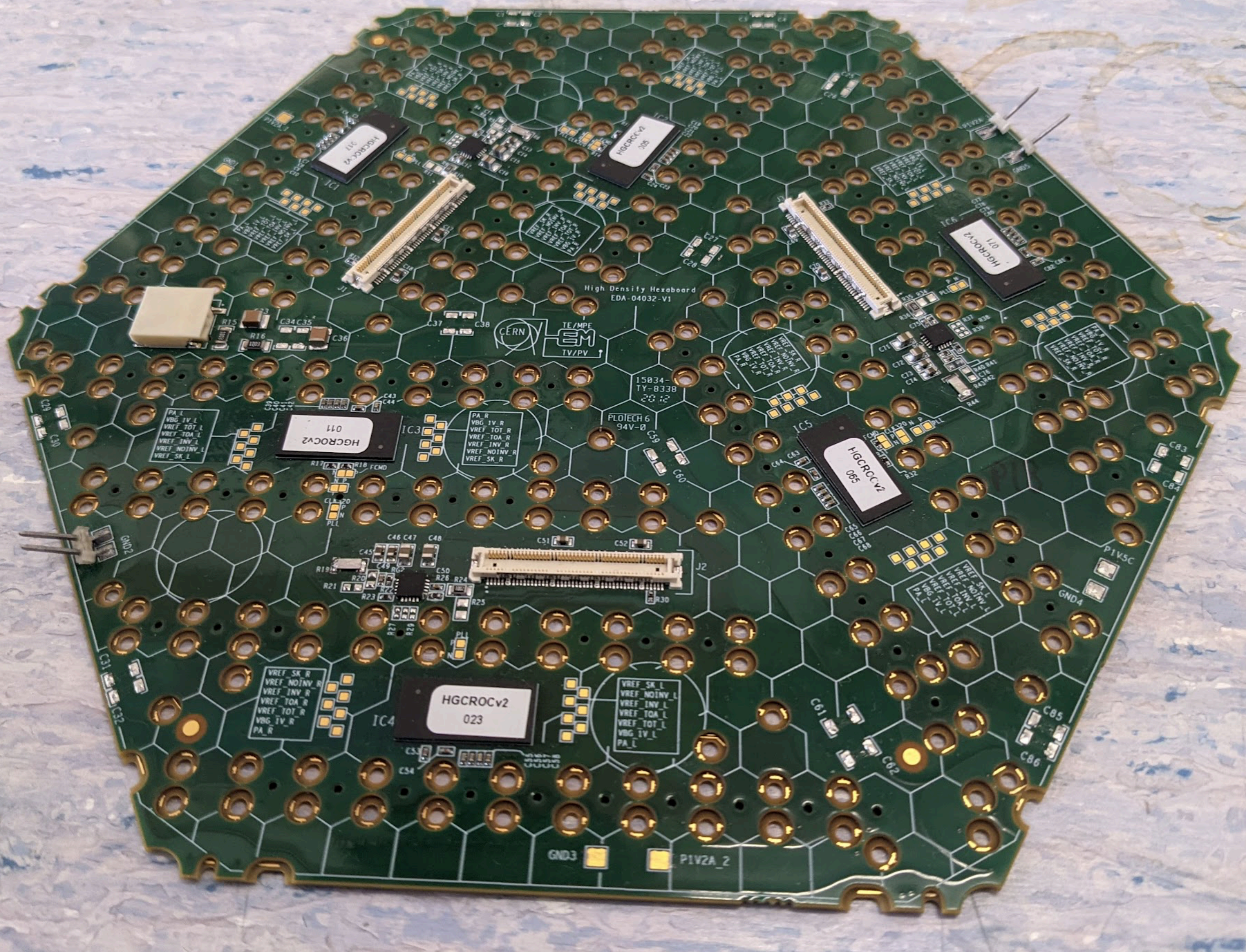
“It will be great if we can display **the wafer map** on **DQM GUI**.”

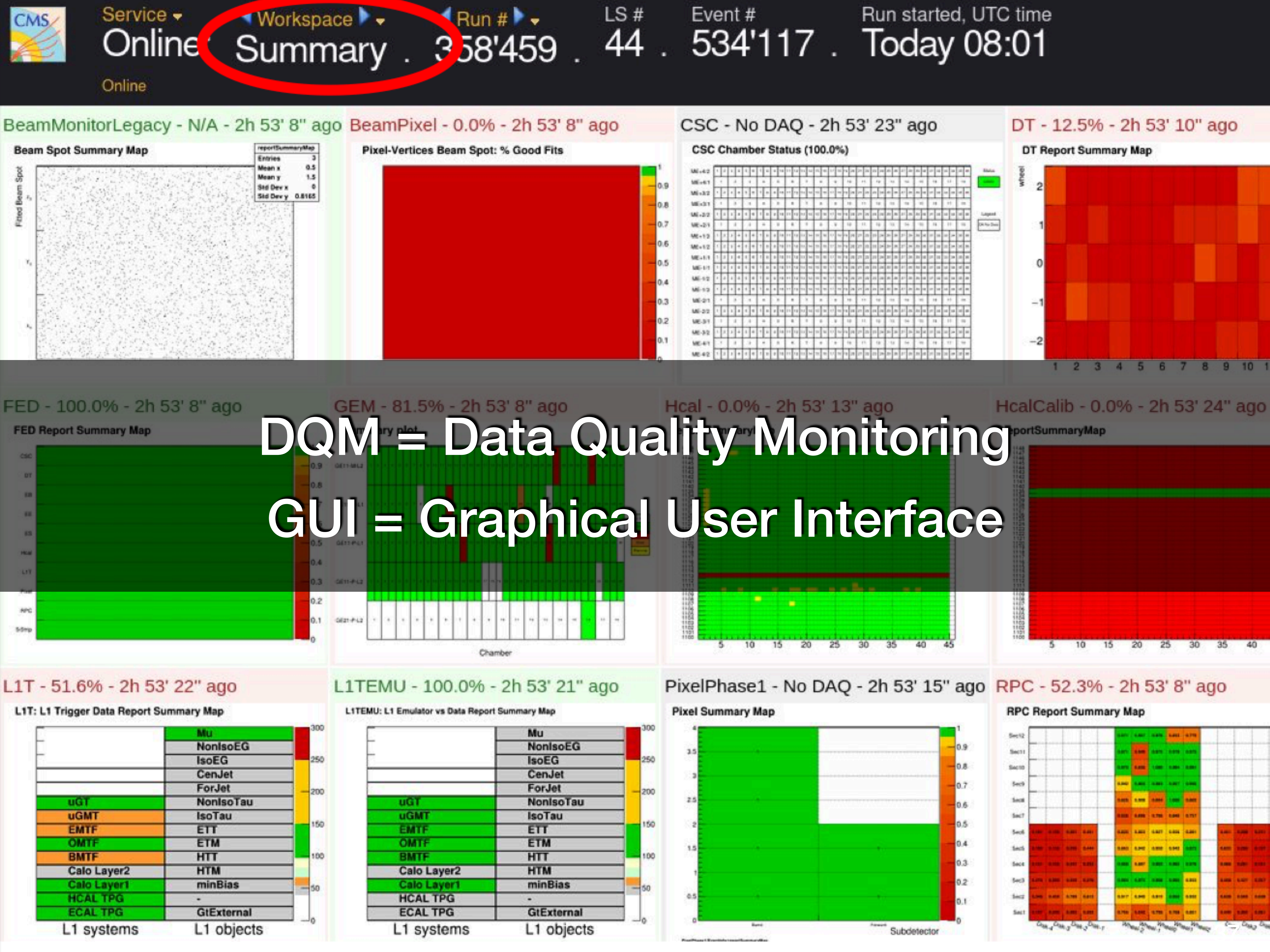
Beginning of the story

"It will be great if we can display **the wafer map** on **DQM GUI**."

"Okay, let me try."

Cells in hexagonal shape or irregular polygons





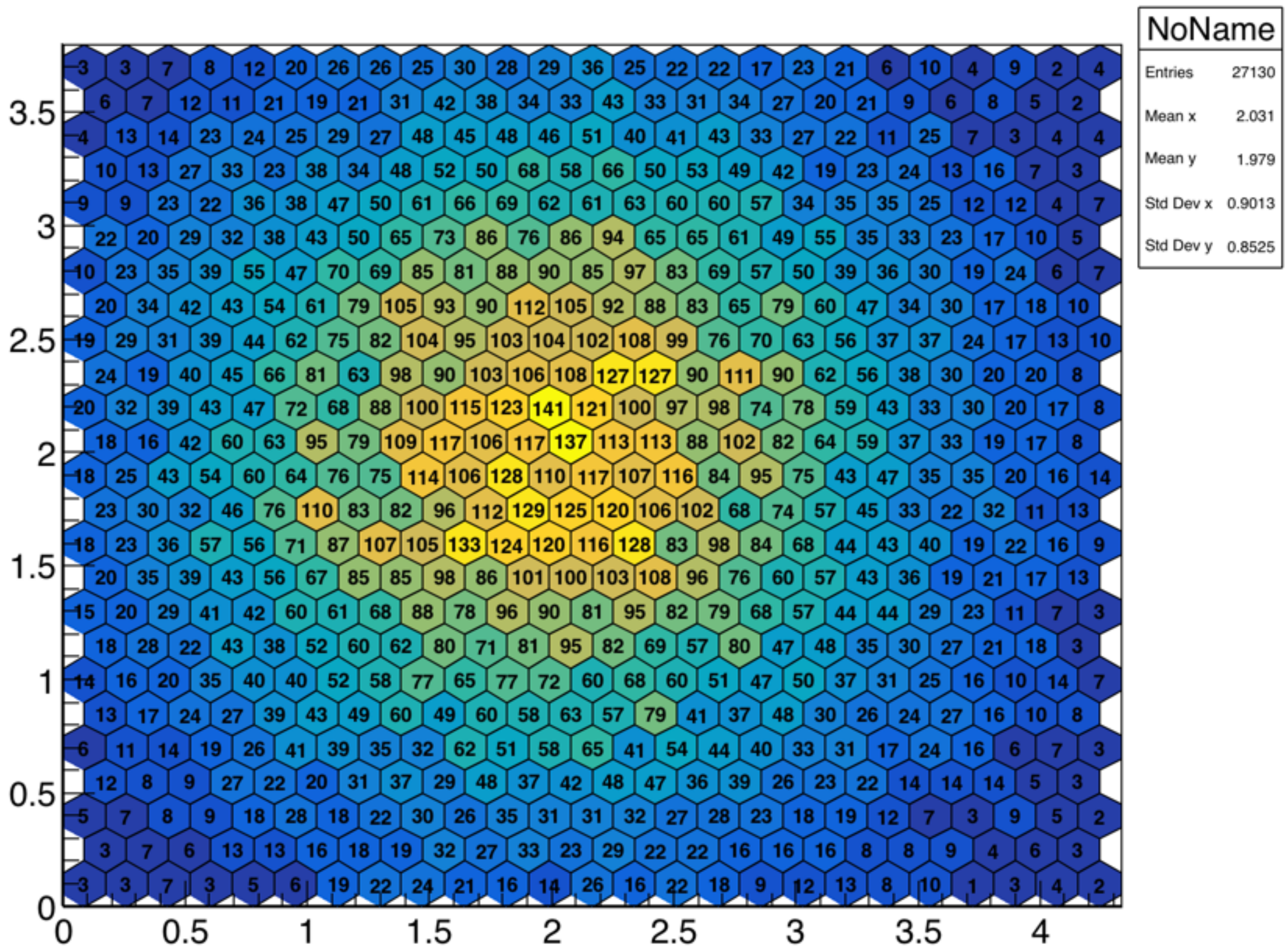
DQM = Data Quality Monitoring

GUI = Graphical User Interface

Reference: hexagonal bins

Possible approaches using ROOT

1. TH2Poly example from ROOT forum

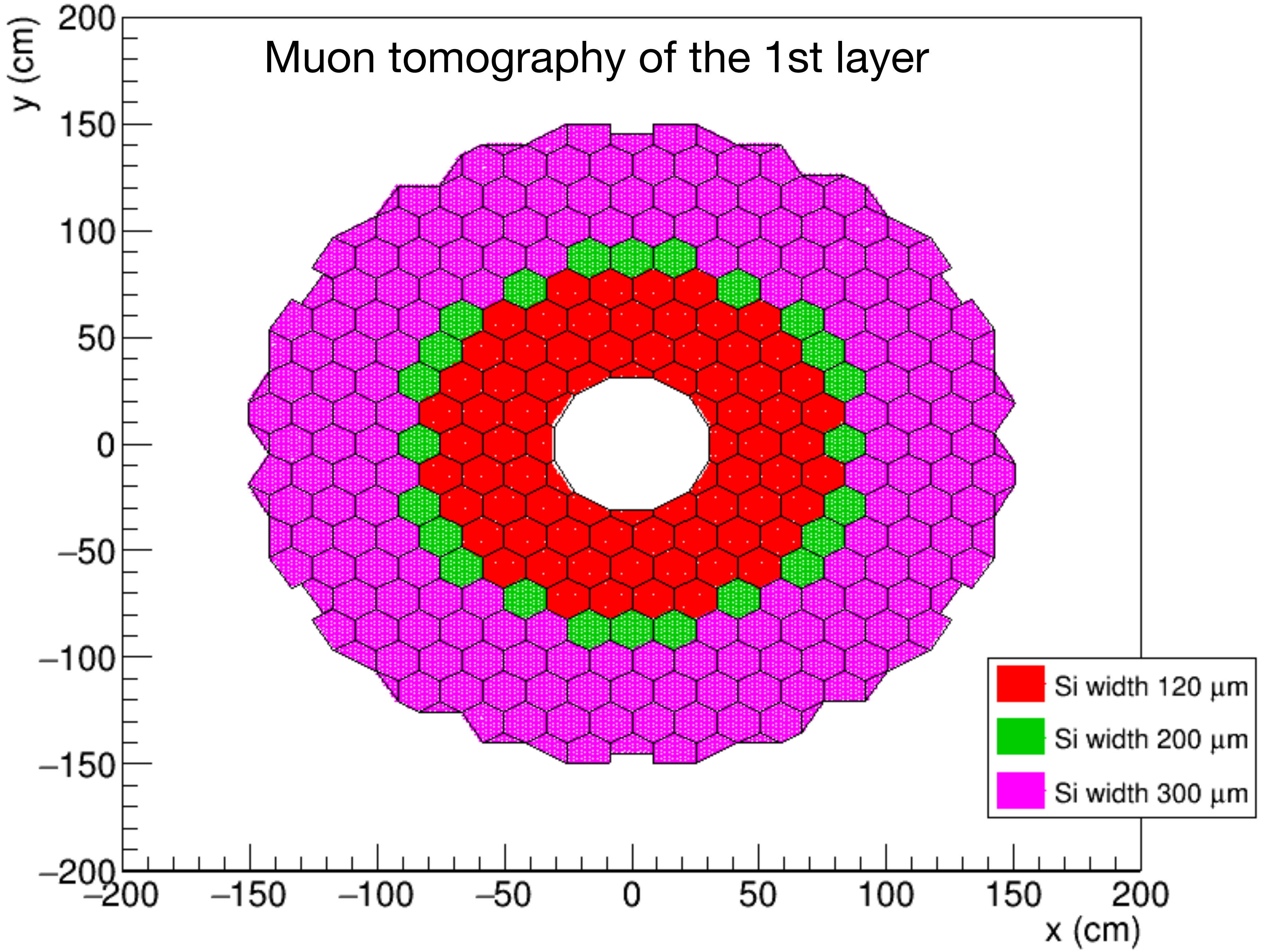


Seeking for a feasible way

Reference: HGCal Geant4 studies

Possible approaches using ROOT

2. TGraph example from a python script

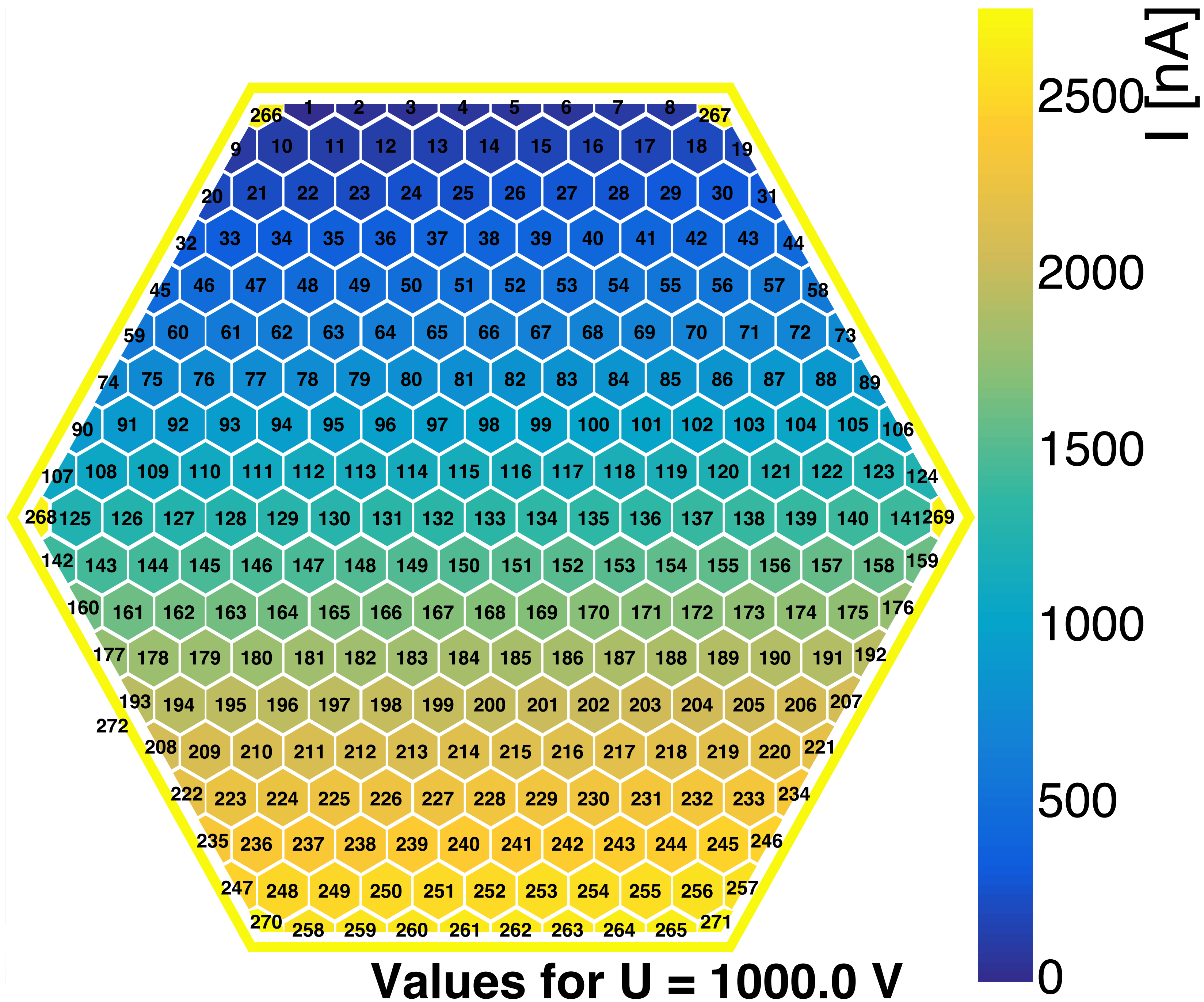


Seeking for a feasible way

Reference: HGCal sensor analysis

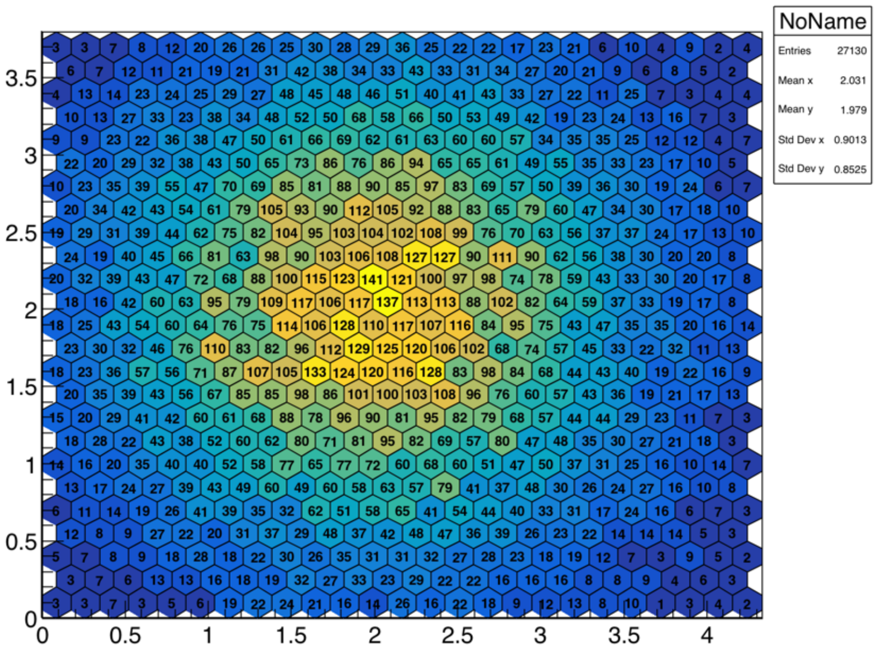
Possible approaches using ROOT

3. TPolyLine example from an independent C++ framework

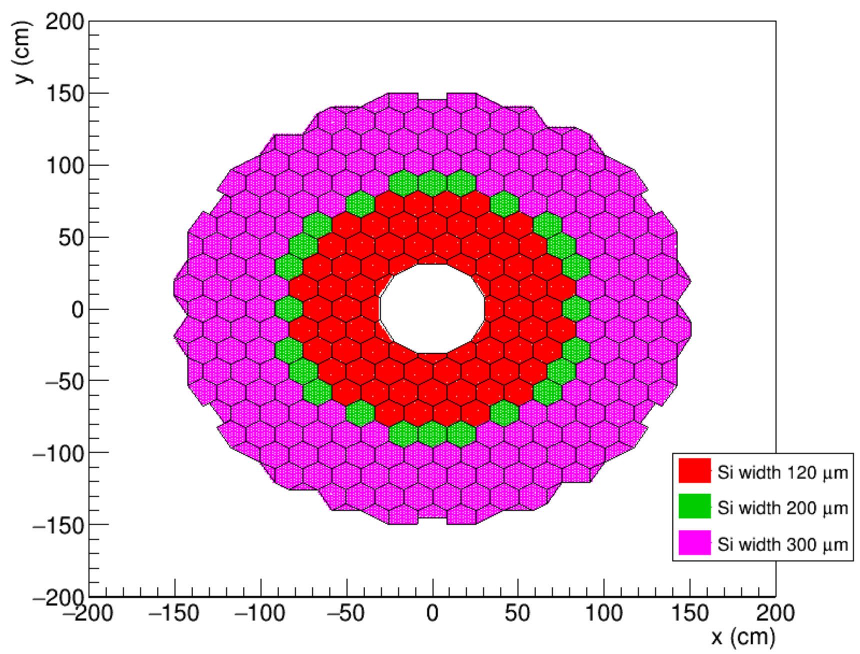


Which is more suitable?

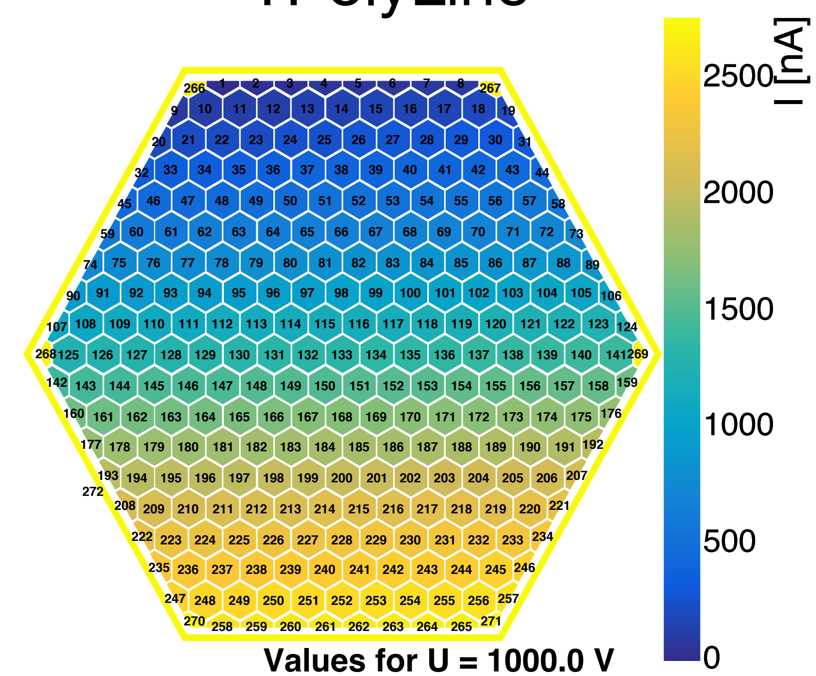
TH2Poly



TGraph



TPolyLine

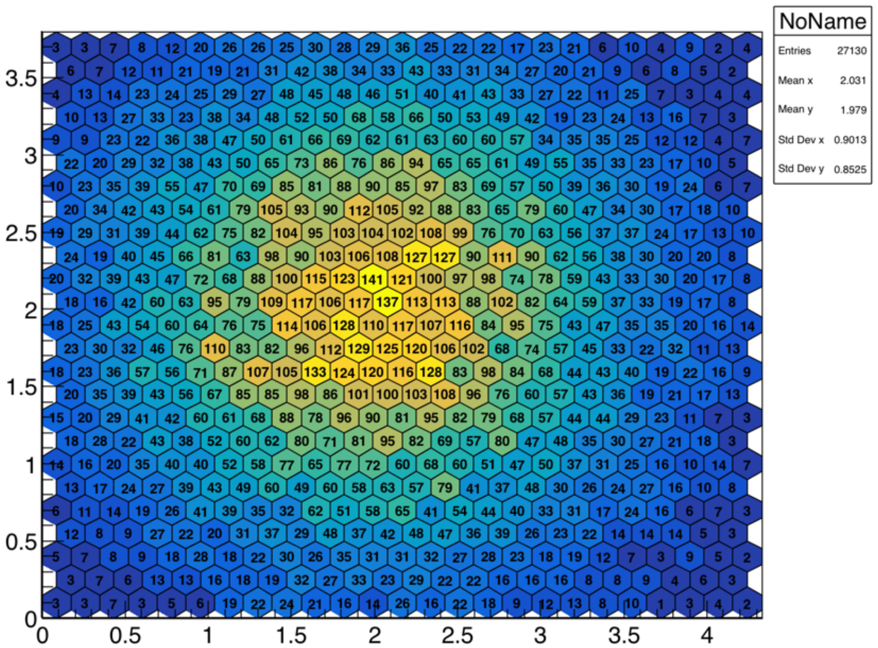


Considerations of implementation

- Can be implemented in CMSSW modules (C++)
- Can fit in DQM monitor elements (TH1, TH2, etc.)

Which is more suitable?

TH2Poly



TGraph

TPolyLine

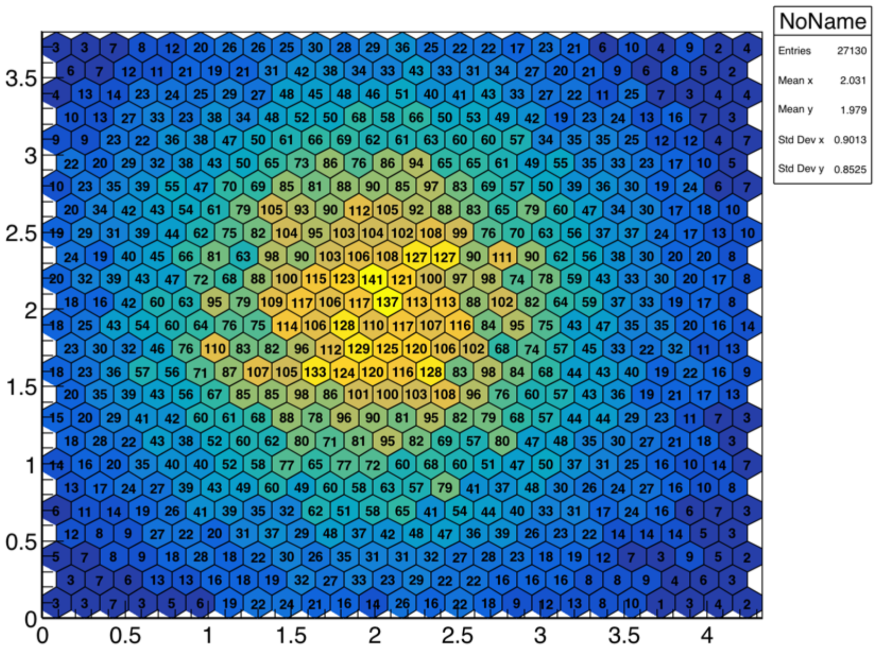
Considerations of implementation

- Can be implemented in CMSSW modules (C++)
- Can fit in DQM monitor elements (TH1, TH2, etc.)

→ TH2Poly is a natural choice among them

Which is more suitable?

TH2Poly



TGraph

TPolyLine

Considerations of implementation

- Can be implemented in CMSSW modules (C++)
- Can fit in DQM monitor elements (TH1, TH2, etc.)

→ TH2Poly is a natural choice among them

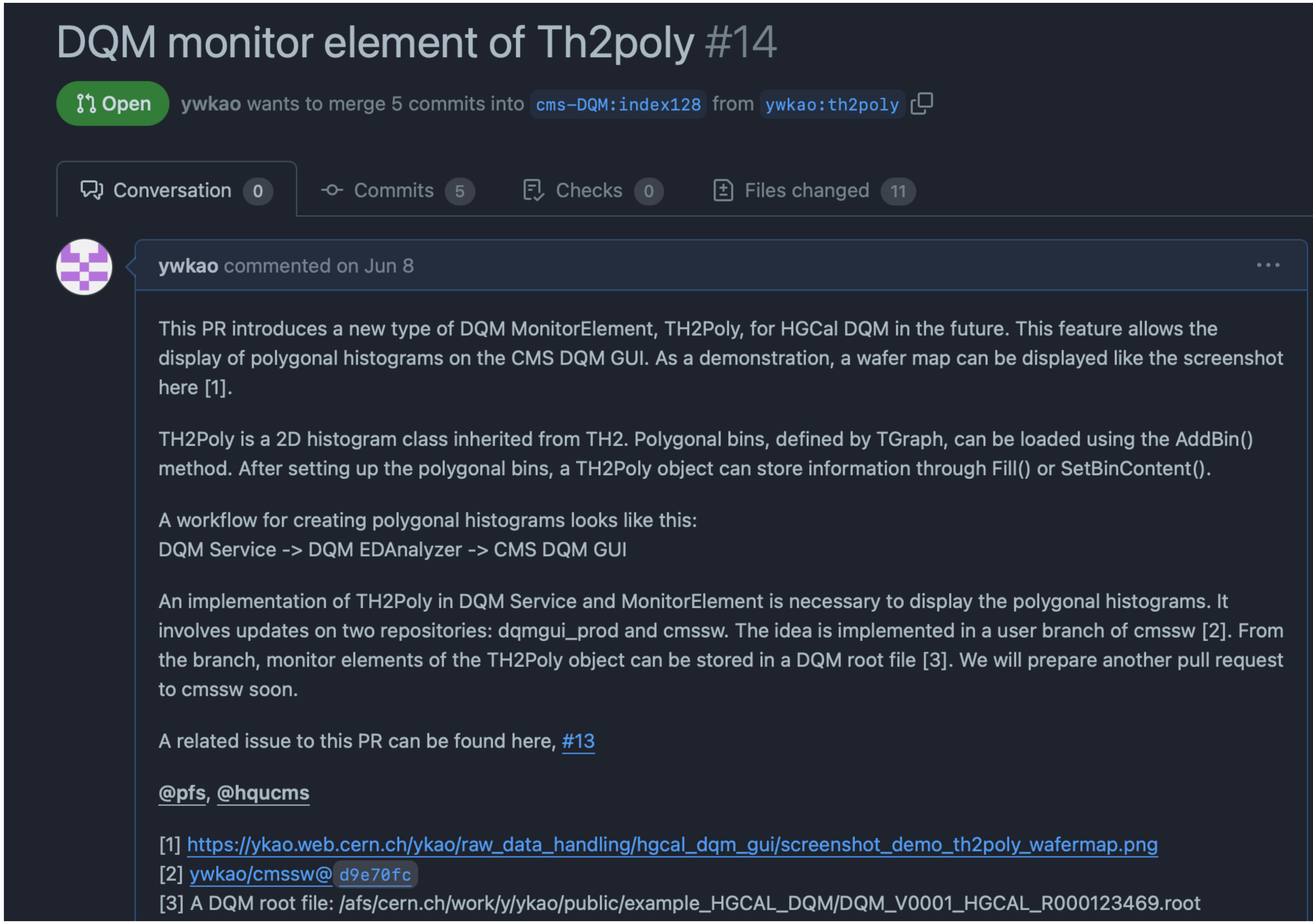
→ Generate polygonal bins from an external python script

Embark on the journey

However…

- TH2Poly has not yet been in the DQM monitor elements in CMSSW
- The CMS DQM GUI is an external package…

Embark on the journey


DQM monitor element of Th2poly #14

Open ywkao wants to merge 5 commits into cms-DQM:index128 from ywkao:th2poly

Conversation 0 | Commits 5 | Checks 0 | Files changed 11

ywkao commented on Jun 8

This PR introduces a new type of DQM MonitorElement, TH2Poly, for HGCal DQM in the future. This feature allows the display of polygonal histograms on the CMS DQM GUI. As a demonstration, a wafer map can be displayed like the screenshot here [1].

TH2Poly is a 2D histogram class inherited from TH2. Polygonal bins, defined by TGraph, can be loaded using the AddBin() method. After setting up the polygonal bins, a TH2Poly object can store information through Fill() or SetBinContent().

A workflow for creating polygonal histograms looks like this:
DQM Service -> DQM EDAnalyzer -> CMS DQM GUI

An implementation of TH2Poly in DQM Service and MonitorElement is necessary to display the polygonal histograms. It involves updates on two repositories: dqmgui_prod and cmssw. The idea is implemented in a user branch of cmssw [2]. From the branch, monitor elements of the TH2Poly object can be stored in a DQM root file [3]. We will prepare another pull request to cmssw soon.

A related issue to this PR can be found here, #13

@pfs, @hqucms

[1] https://ykao.web.cern.ch/ykao/raw_data_handling/hgcal_dqm_gui/screenshot_demo_th2poly_wafermap.png
[2] ywkao/cmssw@d9e70fc
[3] A DQM root file: /afs/cern.ch/work/y/ykao/public/example_HGCAL_DQM/DQM_V0001_HGCAL_R000123469.root

ments in CMSSW

Embark on the journey

DQM monitor element of Th2poly #14

Open ywkao wants to merge 5 commits into cms-DQM:index128 from ywkao:th2poly

Conversation 0 | Commits 5

ywkao commented on Jun 8

This PR introduces a new type of DQM Mon
display of polygonal histograms on the CMS
here [1].

TH2Poly is a 2D histogram class inherited fr
method. After setting up the polygonal bins

A workflow for creating polygonal histogram
DQM Service -> DQM EDAnalyzer -> CMS D

An implementation of TH2Poly in DQM Serv
involves updates on two repositories: dqmg
the branch, monitor elements of the TH2Po
to cmssw soon.

A related issue to this PR can be found here

@pfs, @hqucms

[1] https://ykao.web.cern.ch/ykao/raw_data_
[2] ywkao/cmssw@d9e70fc
[3] A DQM root file: /afs/cern.ch/work/y/yka

Implement TH2Poly in DQM Services for HGCal DQM #41932

Open ywkao wants to merge 3 commits into cms-sw:master from ywkao:hgcal-dqm_with_th2poly-13_2_X

Conversation 23 | Commits 3 | Checks 0 | Files changed 5

ywkao commented on Jun 12

PR description:

This PR introduces a new type of DQM MonitorElement, TH2Poly, for HGCal DQM in the future. This feature allows a display of polygonal histograms on the CMS DQM GUI. As a demonstration, a wafer map can be displayed like the screenshot here [1].

TH2Poly is a 2D histogram class inherited from TH2. Polygonal bins, defined by TGraph, can be loaded using the AddBin() method. After setting up the polygonal bins, a TH2Poly object can store information through Fill() or SetBinContent().

A workflow for creating polygonal histograms looks like this:
DQM Service -> DQM EDAnalyzer -> CMS DQM GUI

An implementation of TH2Poly in DQM Service and MonitorElement is necessary to display the polygonal histograms. It involves updates on two repositories: cmssw and dqmgui_prod. A pull request is created in the dqmgui_prod repository [2] with a relevant issue reported in this link [3], which is about setting up a CMS DQM GUI with the new feature.

PR validation:

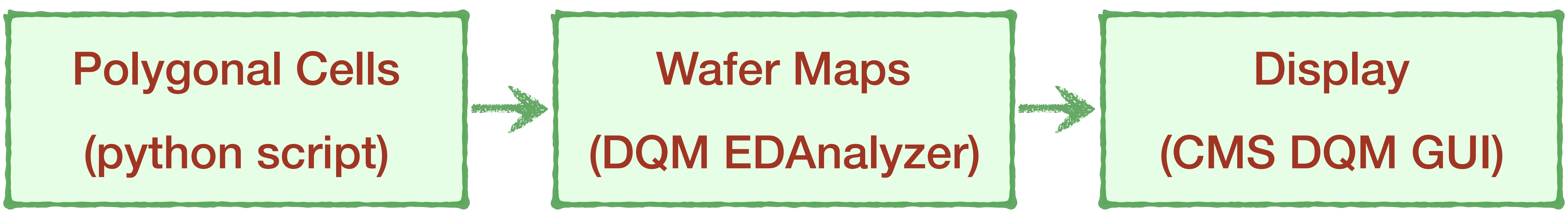
The workflow and the implementation have been tested: (a) From this feature branch, monitor elements of TH2Poly can be stored in a DQM root file [4]. (b) The DQM root file can be uploaded to a CMS DQM GUI, which is built following the steps noted in this issue [3]. Polygonal maps can be displayed on the DQM GUI, as demonstrated in [1].

[1] https://ykao.web.cern.ch/ykao/raw_data_handling/hgcal_dqm_gui/screenshot_demo_th2poly_wafermap.png
[2] cms-DQM/dqmgui_prod#14
[3] cms-DQM/dqmgui_prod#13
[4] A DQM root file containing demo polygonal maps:
/afs/cern.ch/work/y/ykao/public/example_HGCAL_DQM/DQM_V0001_HGCAL_R000123469.root

Wafer maps in TH2Poly

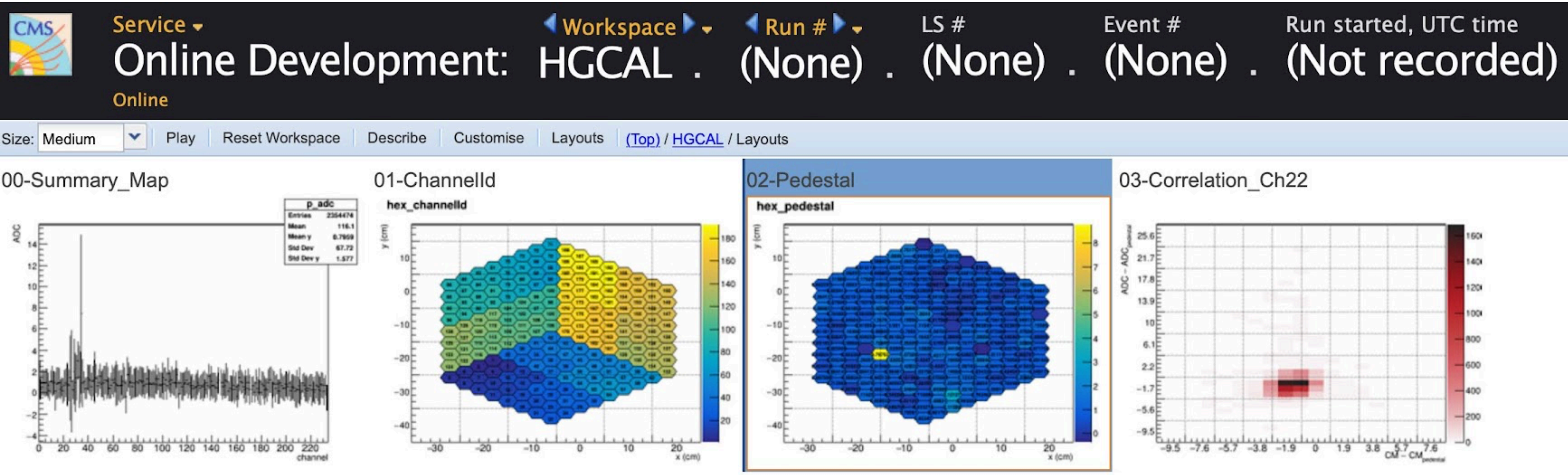
Workflow of the wafer maps

1. Create polygonal cells using an external python script
2. Book wafer maps for data monitoring in DQM EDAnalyzer
3. Display plots on the DQM GUI



GitHub:HexHistograms   PR-CMS-DQM #14   PR-CMSSW #41932

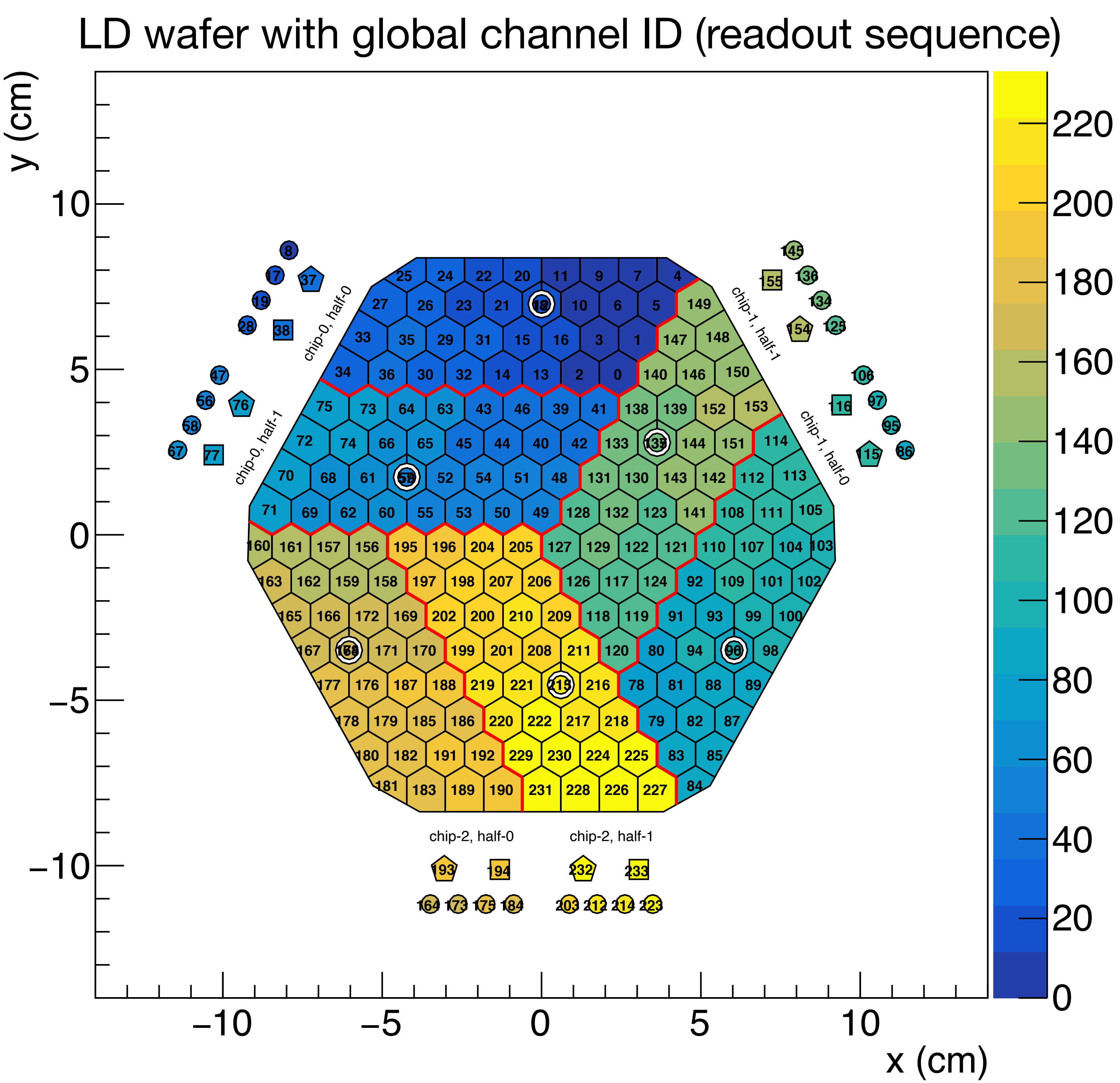
The start of the HGCAL DQM



The first hexagonal histograms on the CMS DQM GUI !

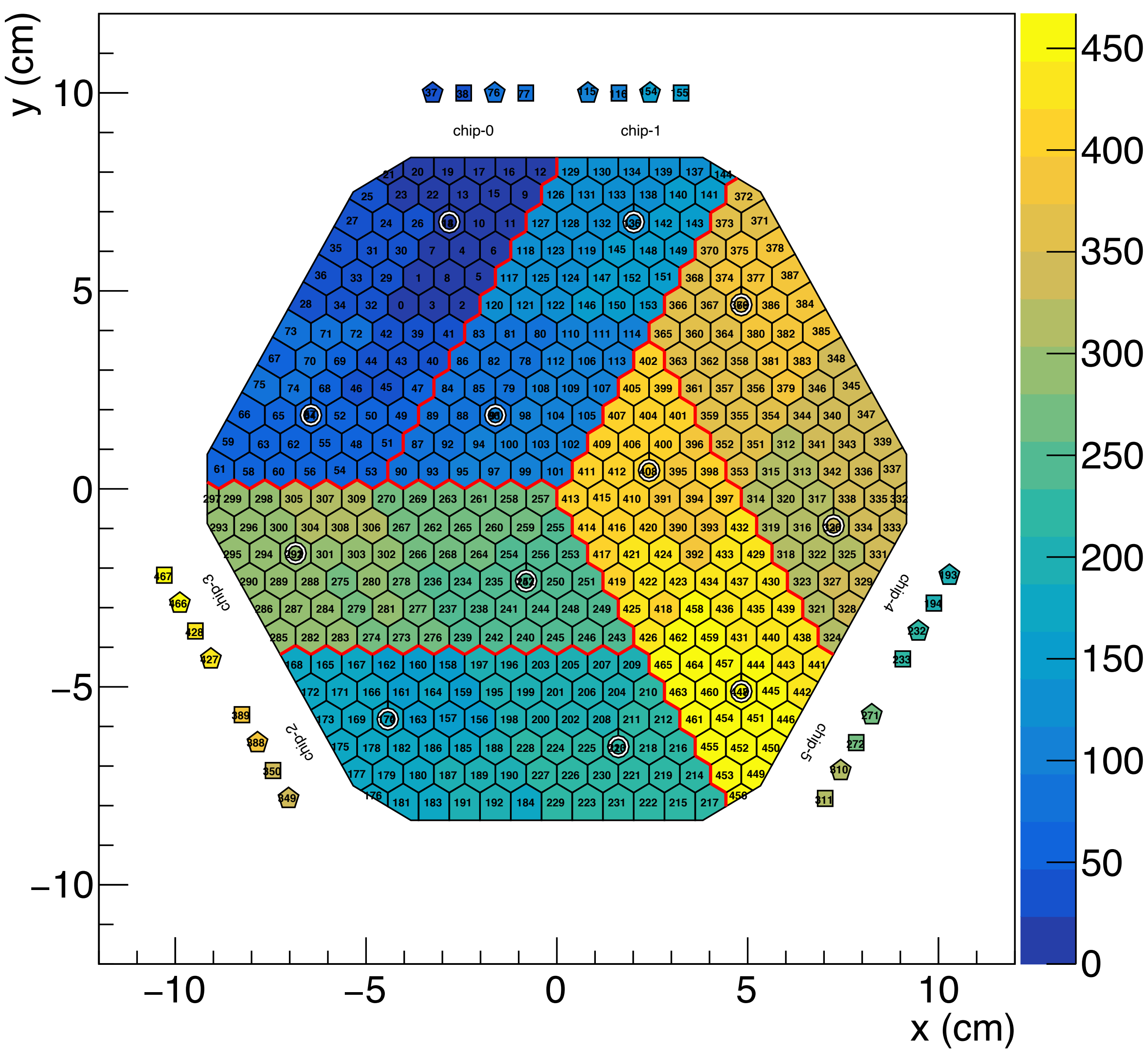
Celebration on the progress

Cosmetics on DQM wafer map

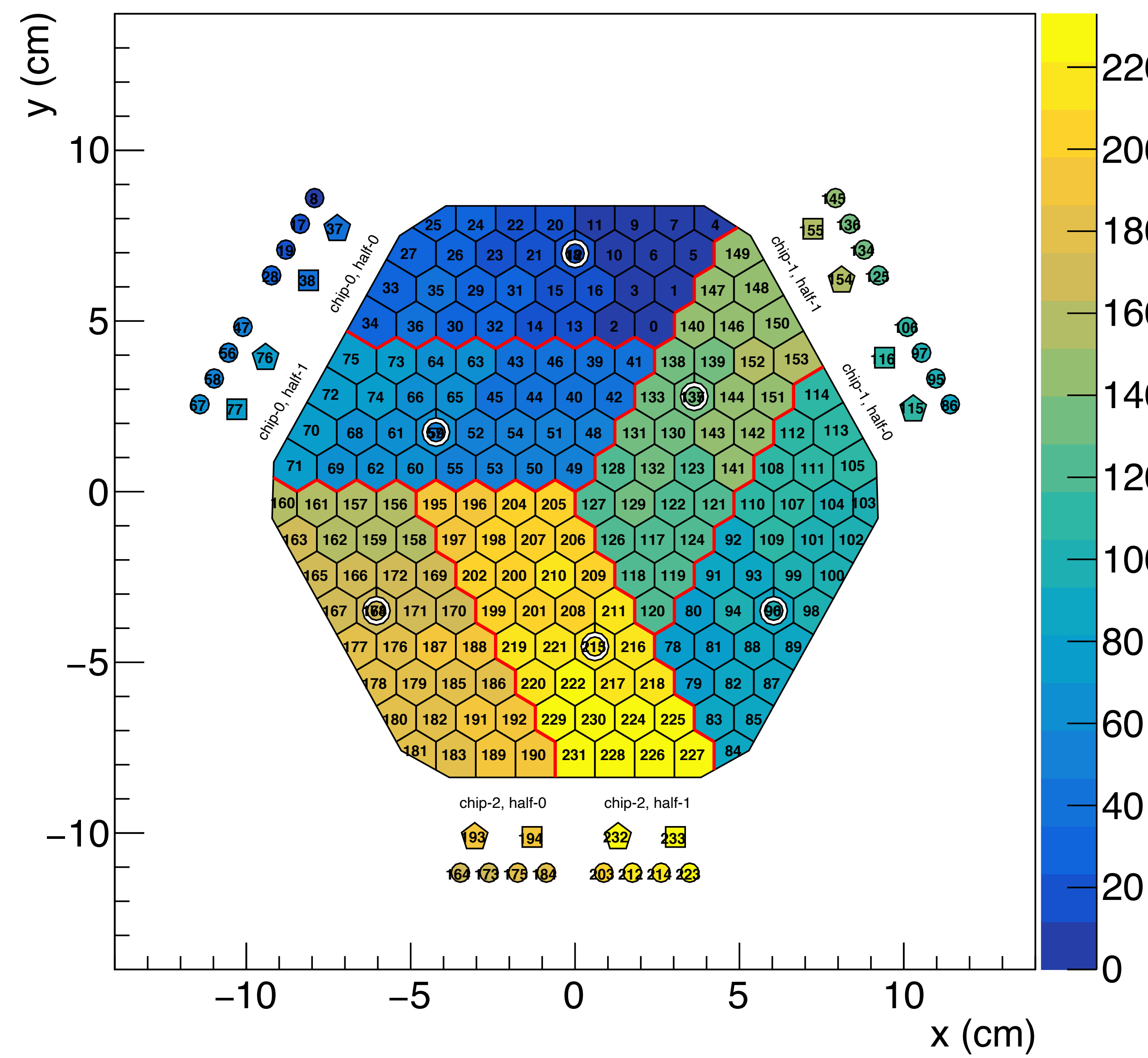


DQM histograms for full wafers

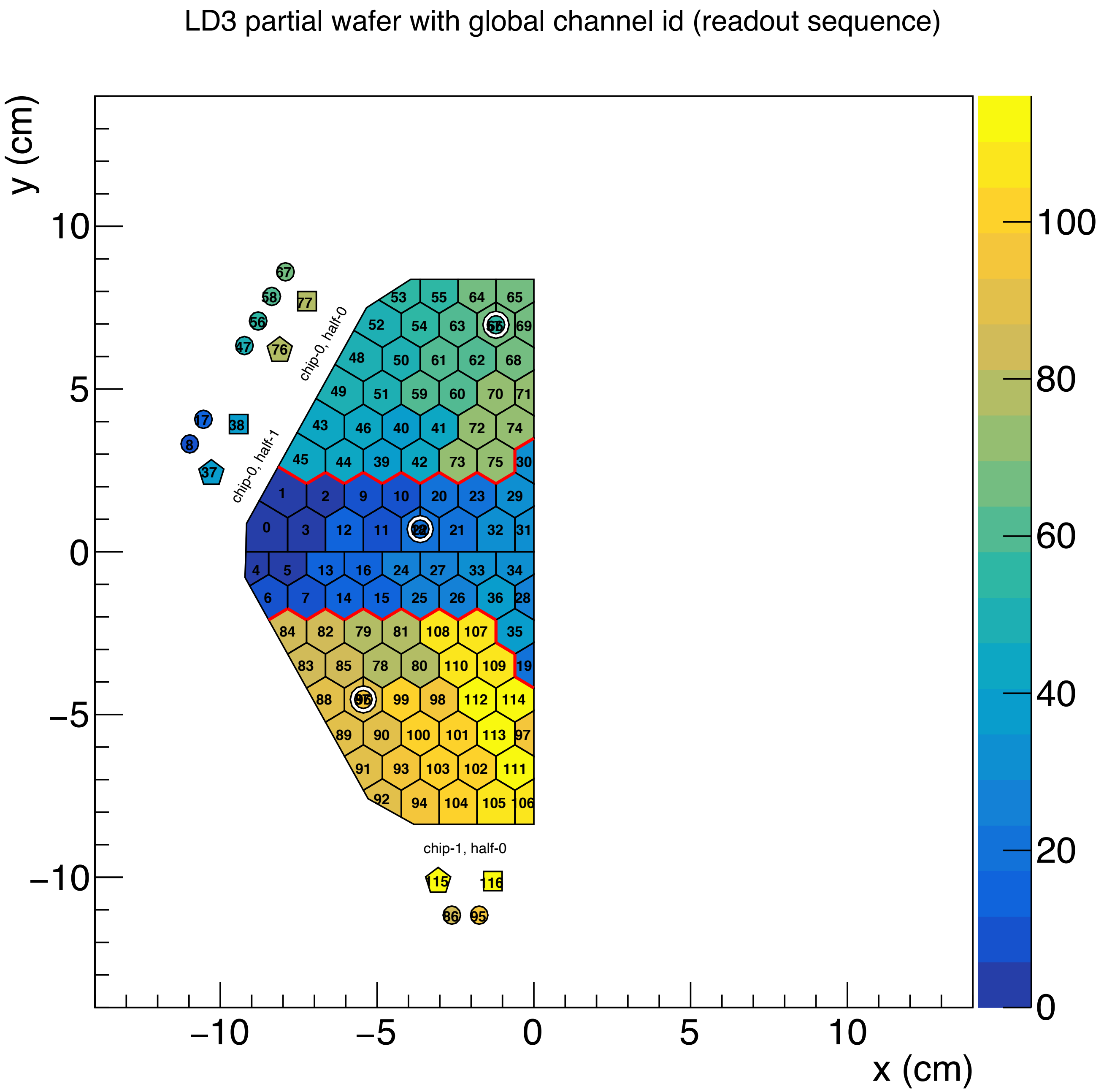
HD wafer with global channel id (readout sequence)

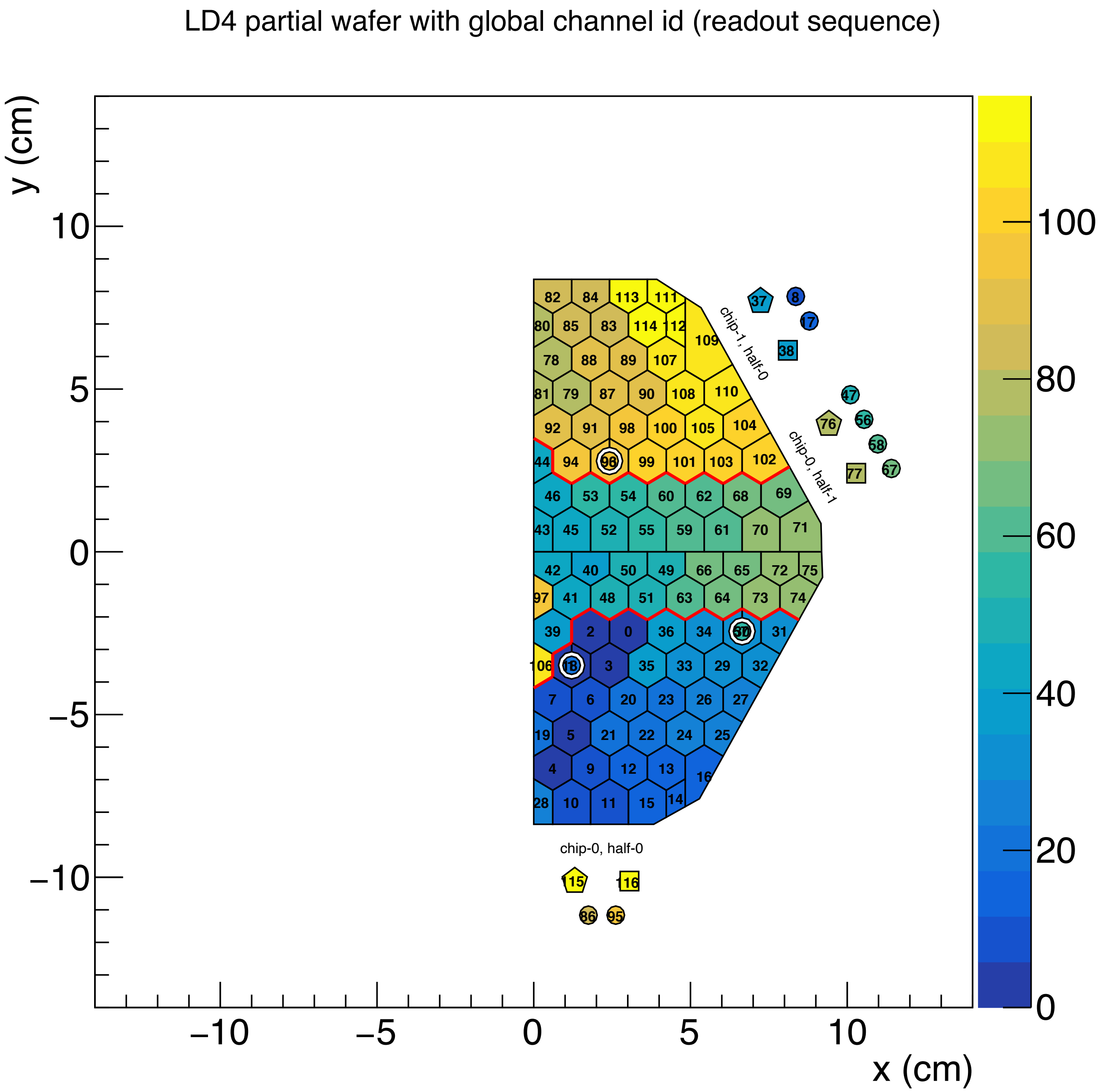


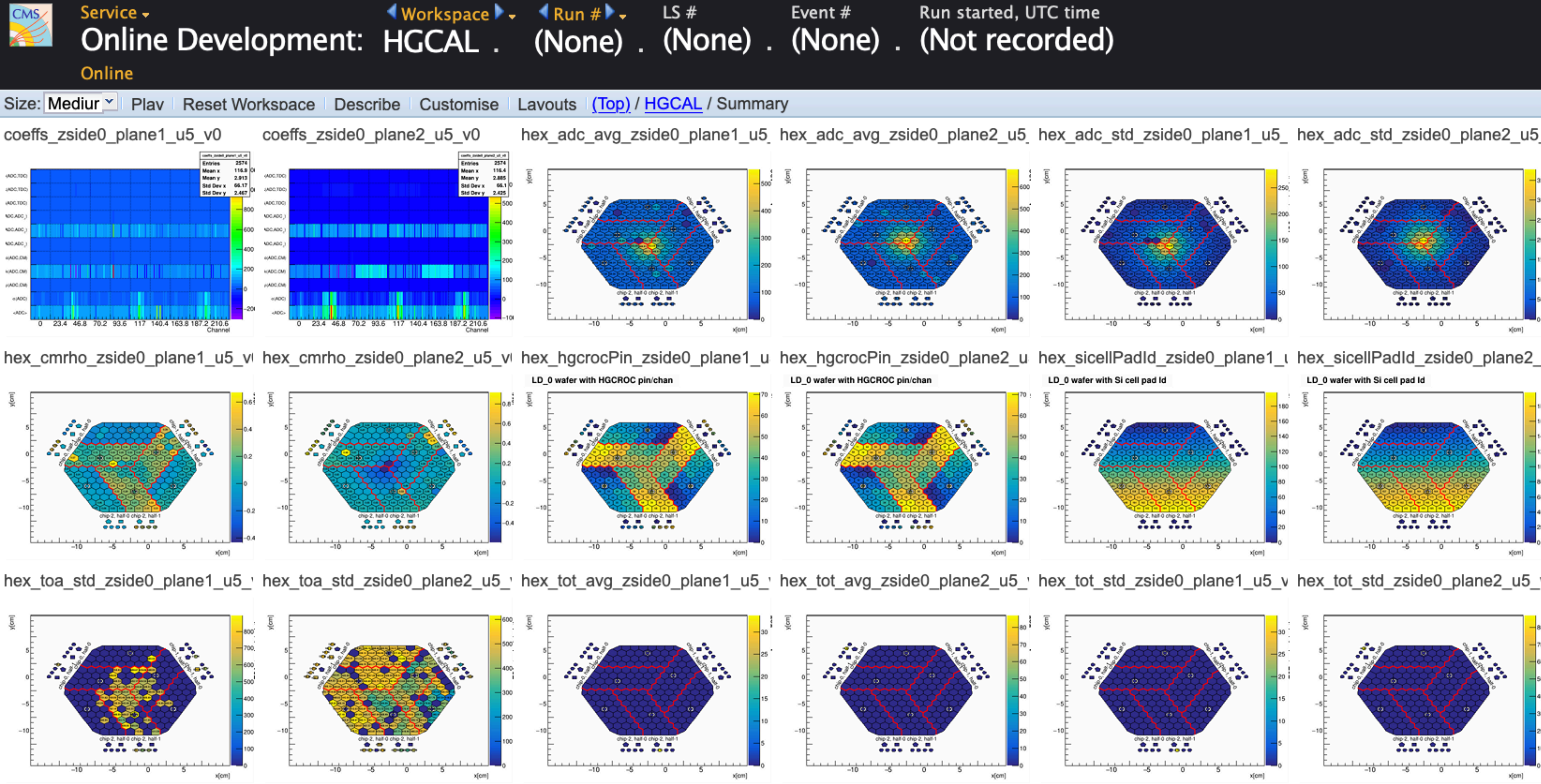
LD wafer with global channel id (readout sequence)



DQM histograms for partial wafers



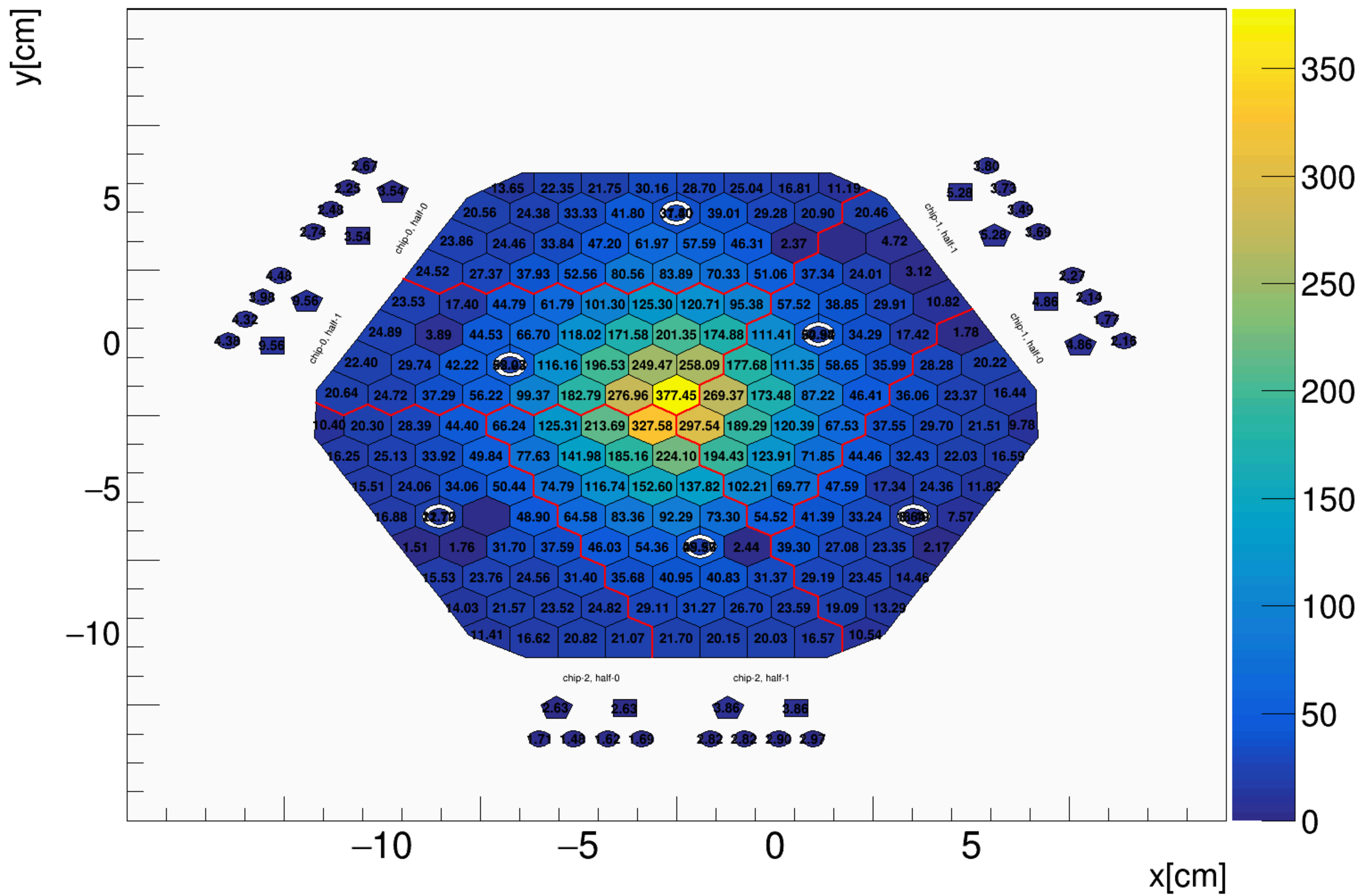


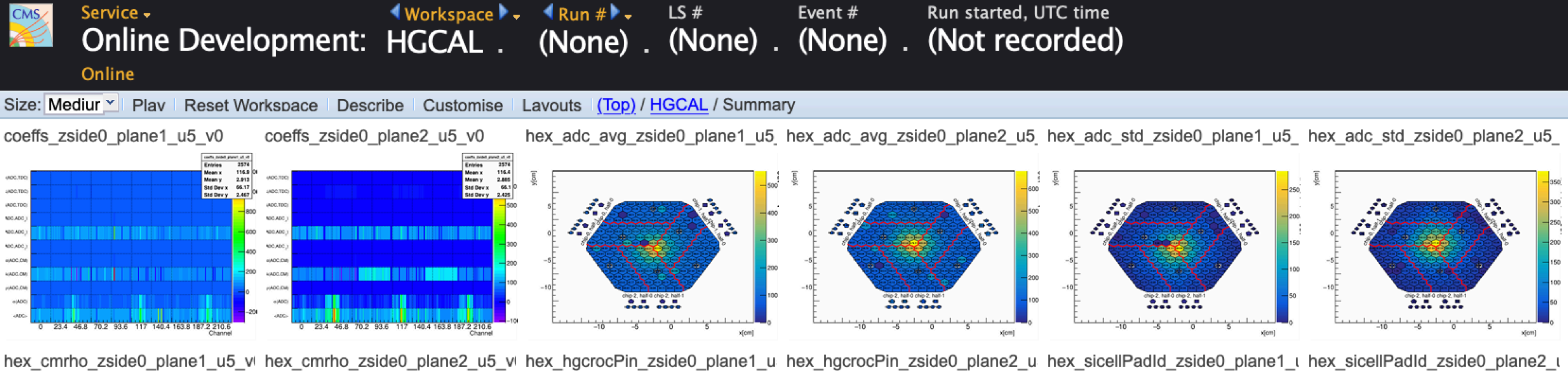


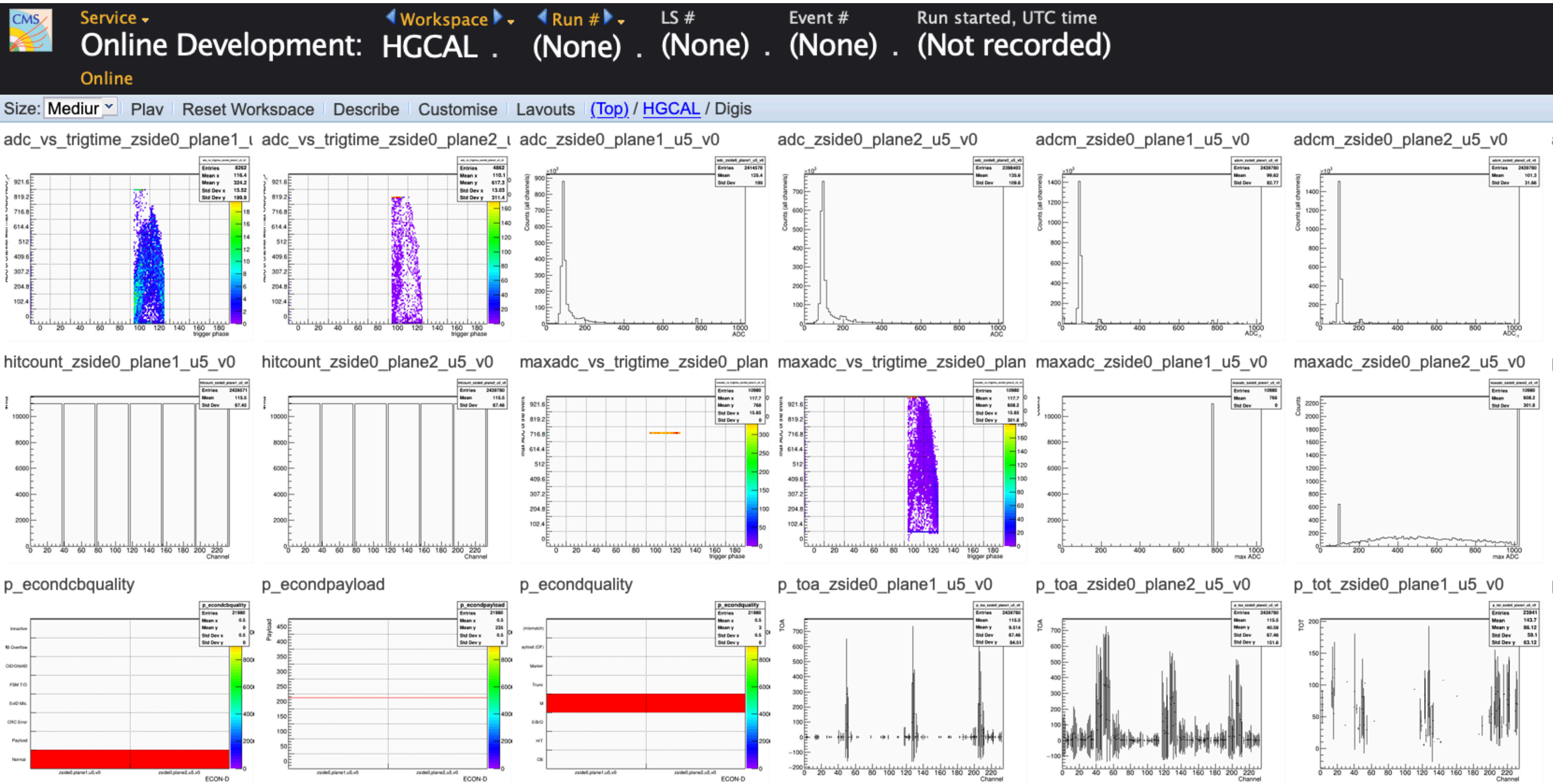
DQM GUI during the test beam activity

Service ▾ | Workspace ▾ | Run # ▾ | LS # | Event # | Run started, UTC time

Online Development: HGCAL . (None) . (None) . (None) . (Not recorded)




Service
Online Development: HGCAL
Online
Workspace
Run #
LS #
Event #
Run started, UTC time
(None)
(None)
(None)
(Not recorded)
Size: Mediur
Play
Reset Workspace
Describe
Customise
Layouts
(Top) / HGCAL / Summary
coeffs_zside0_plane1_u5_v0
coeffs_zside0_plane2_u5_v0
hex_adc_avg_zside0_plane1_u5
hex_adc_avg_zside0_plane2_u5
hex_adc_std_zside0_plane1_u5
hex_adc_std_zside0_plane2_u5
hex_cmrho_zside0_plane1_u5_v
hex_cmrho_zside0_plane2_u5_v
hex_hgcrocPin_zside0_plane1_u
hex_hgcrocPin_zside0_plane2_u
hex_sicellPadId_zside0_plane1_
hex_sicellPadId_zside0_plane2_


Service
Online Development: HGCAL
Online
Workspace
Run #
LS #
Event #
Run started, UTC time
(None)
(None)
(None)
(Not recorded)
Size: Mediur
Play
Reset Workspace
Describe
Customise
Layouts
(Top) / HGCAL / Digis
adc_vs_trigtime_zside0_plane1_
adc_vs_trigtime_zside0_plane2_
adc_zside0_plane1_u5_v0
adc_zside0_plane2_u5_v0
adcm_zside0_plane1_u5_v0
adcm_zside0_plane2_u5_v0
hitcount_zside0_plane1_u5_v0
hitcount_zside0_plane2_u5_v0
maxadc_vs_trigtime_zside0_plan
maxadc_vs_trigtime_zside0_plan
maxadc_zside0_plane1_u5_v0
maxadc_zside0_plane2_u5_v0
p_econdcbquality
p_econdpayload
p_econdquality
p_toa_zside0_plane1_u5_v0
p_toa_zside0_plane2_u5_v0
p_tot_zside0_plane1_u5_v0

Summary

Contributions

- Involved in the HGCAL raw data handling tasks with a realistic data processing chain established in a CMSSW branch in the past year. (RAW → DIGI → RECO → DQM / Nano)
- Implemented level-0 calibration algorithms in Alpaka modules for heterogeneous computing. GPU already shows an order of magnitude faster for the expected HGCAL multiplicity in the preliminary study.
- Initiated HGCAL DQM with polygonal DQM monitor elements implemented. A DQM GUI was built for the 2023 beam test activities.

Next steps of the raw data handling group

- More calibration algorithms for local reconstruction will be studied.
- Scale up from the wafer-level test to cassette-level test.

Acknowledgement

Advisors

- Kai-Feng Chen and Stathes Paganis

HGCAL Calibration & Heterogeneous computing

- Andre Bocci, Andre Govinda Stahl Leiton, Eric Cano, Geliang Liu, Huilin Qu, Izaak Neutelings, Jeremi Niedziela, Pedro Silva, Yulun Miao

HGCAL DQM

- Arnaud Steen, Andre David, Chris Seez, Dimitra Tsionou, Eiko Shin-Shan Yu, Huilin Qu, Javier Fernandez, Marco Rovere, Pedro Silva, Pruthvi Suryadevara, Yulun Miao


19. Jun. 2023
HGCAL workshop
BBQ event @ CERN Prevessin
Thank you!


Backup

Energy information

- Energy path
  - conversion to charge (A→q)
  - subtract the baseline stochastic noise: pedestal (P)
  - subtract fluctuations of common mode noise (q_{CM})
  - subtract the leakage from the previous bunch ($q_{0,-1}$)

- Conversion from A to charge is made using the information
  - ADC/TOT mode: extracted from the TcTp flags of the raw data
  - LSB and offset (OFF) set by configuration (4 constants per ROC per run)

$$q_0 = (LSB + 1/2) \cdot A + OFF$$

- Corrected charge measurement

$$\begin{aligned} q &= (q_0 - P) + \beta \cdot (q_{CM} - P_{CM,0}) + \kappa \cdot \left[(q_{0,-1} - P) + \beta \cdot (q_{CM} - P_{CM,0}) \right] \\ &= q_0 - (1+\kappa) \cdot P - \kappa \cdot q_{0,-1} + \beta \cdot (1+\kappa) \cdot (q_{CM} - P_{CM,0}) \end{aligned}$$

Source: https://docs.google.com/document/d/1fSYI8ftHjVG0DwDt1xpZcBMKixobyeQnKA4T9hScEqg/edit#heading=h.10ts2c4n5i7j

Level-0 calibrations: main operations

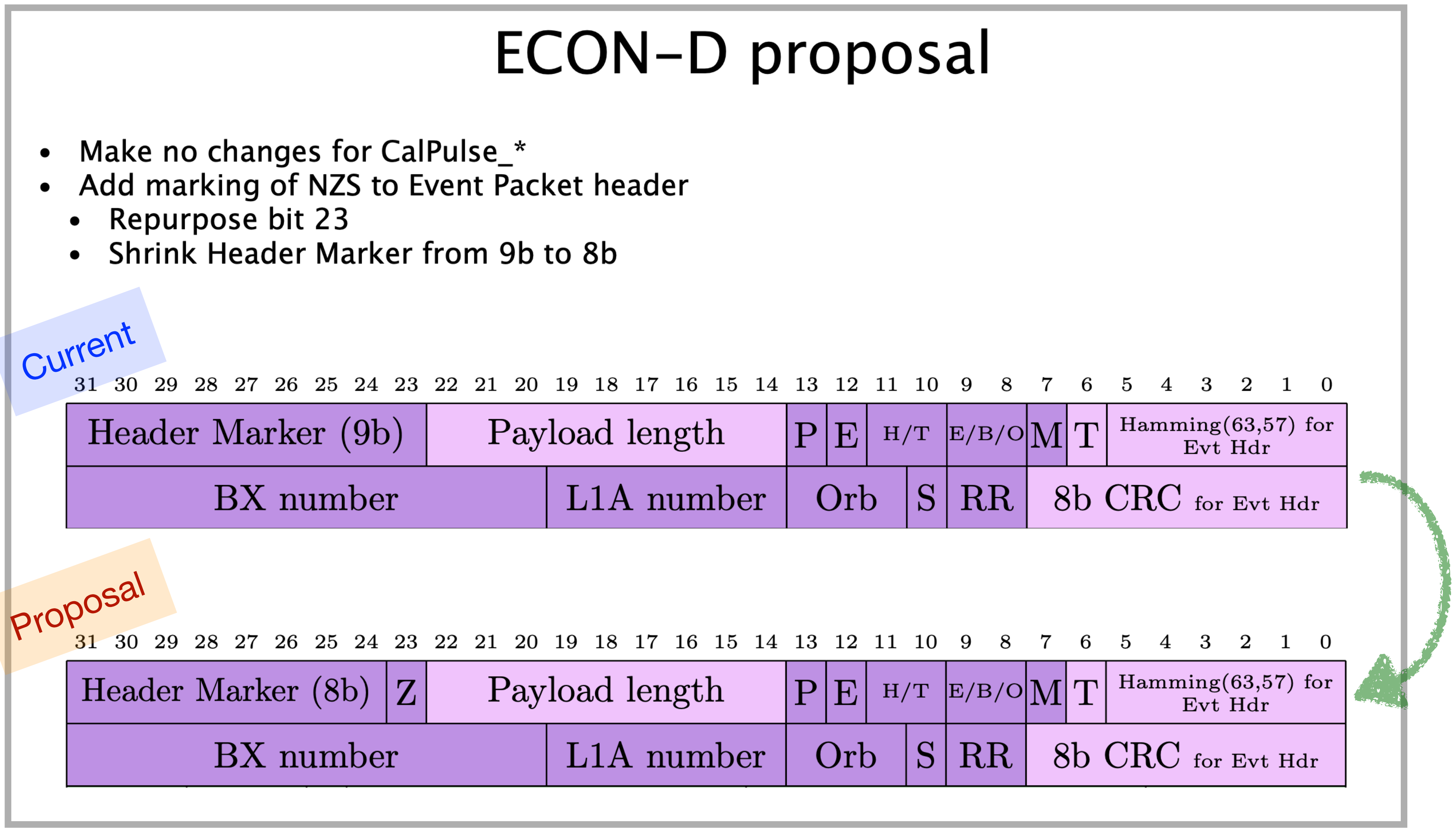
	Target	Dataset	Frequency	Notes
relative calibration	3/15% uncertainty in CE-E/CE-H cells	standard L-1 triggers	few times a year	MIP at 10 ADC counts ZS threshold 0.5 MIP
pedestal	0.3 LSB uncertainty in mean	unsuppressed readout in standard L-1 triggers	see discussion in text	
charge non-linearity	2%	charge injection data	infrequent	
ToA time slew		charge injection data	infrequent	
SiPM non-linearity	10%??	LED data	commissioning and startup	single p.e. peak
TDC non-linearity	15 ps	random-clock events	infrequent	see Ref. [3]
time zero-offset	15 ps	standard L-1 triggers	every run	

Source: https://gitlab.cern.ch/tdr/notes/DN-20-002

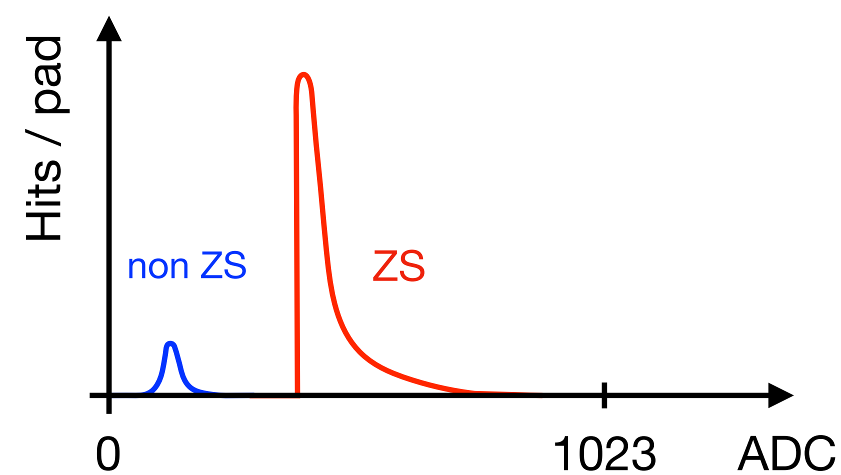
- DPG is in the process of updating DN-20-002 to reflect latest discussions
- Pedestal and time zero-offset will be the most frequent
- Need special S-Link data for non-linearity of charge and SiPM and ToA time slew

Pedestal runs: usage of unsupressed events

- Need to choose events without zero suppression for evaluating pedestal
  - Standard level-1 trigger → operations are greatly eased if a flag is in ECON-D header
  - If there is no flag → need a special algorithm to look for unsuppressed events



Source: first page of from slides here



TcTp

- Tc and Tp flags
  - Added (MSB positions) to the 30 bits in order to remove some ambiguities which can occurs in the data path
  - TOT-Complete, Tc: the second 10 bits packet corresponds to TOT, not ADC. It is applicable in lines 3 and 4 of the table
  - TOT-In-Progress, Tp: A TOT occurred in a previous BX and the ADC value can be “corrupted” (saturation or undershoot). It is applicable for lines 1 and 2 and 4 of the table

	ADC (t-1)	ADC (t)	TOT	TOA	Charge collection	Data type
1	x	x		x (=0)	Q < TOA_thr AN	Normal
2	x	x		x	Q < TOT_thr AN	Normal
3	x		x	x	Q > TOT_thr AN	Normal
4		x	x	x		“Characterization”

Case 1 and 2

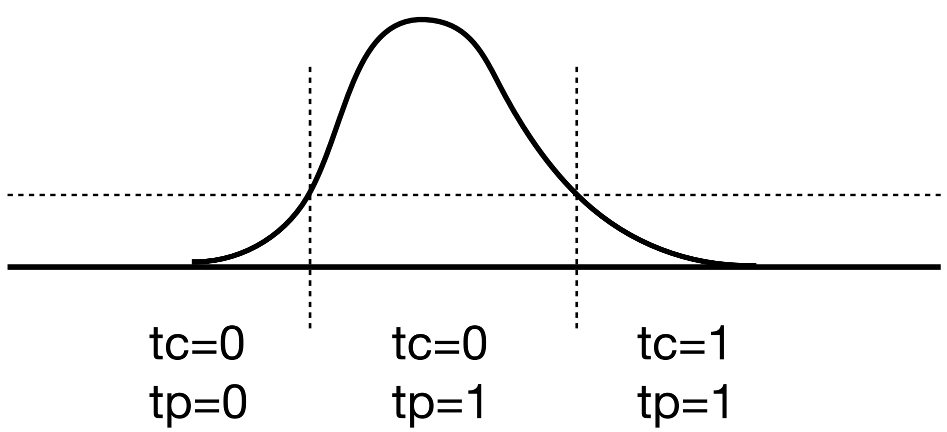
0	Tp	10b ADC-1	10b ADC	10b TOA

Case 3

1	Tp	10b ADC-1	10b TOT	10b TOA

Case 4

Tc	Tp	10b ADC	10b TOT	10b TOA

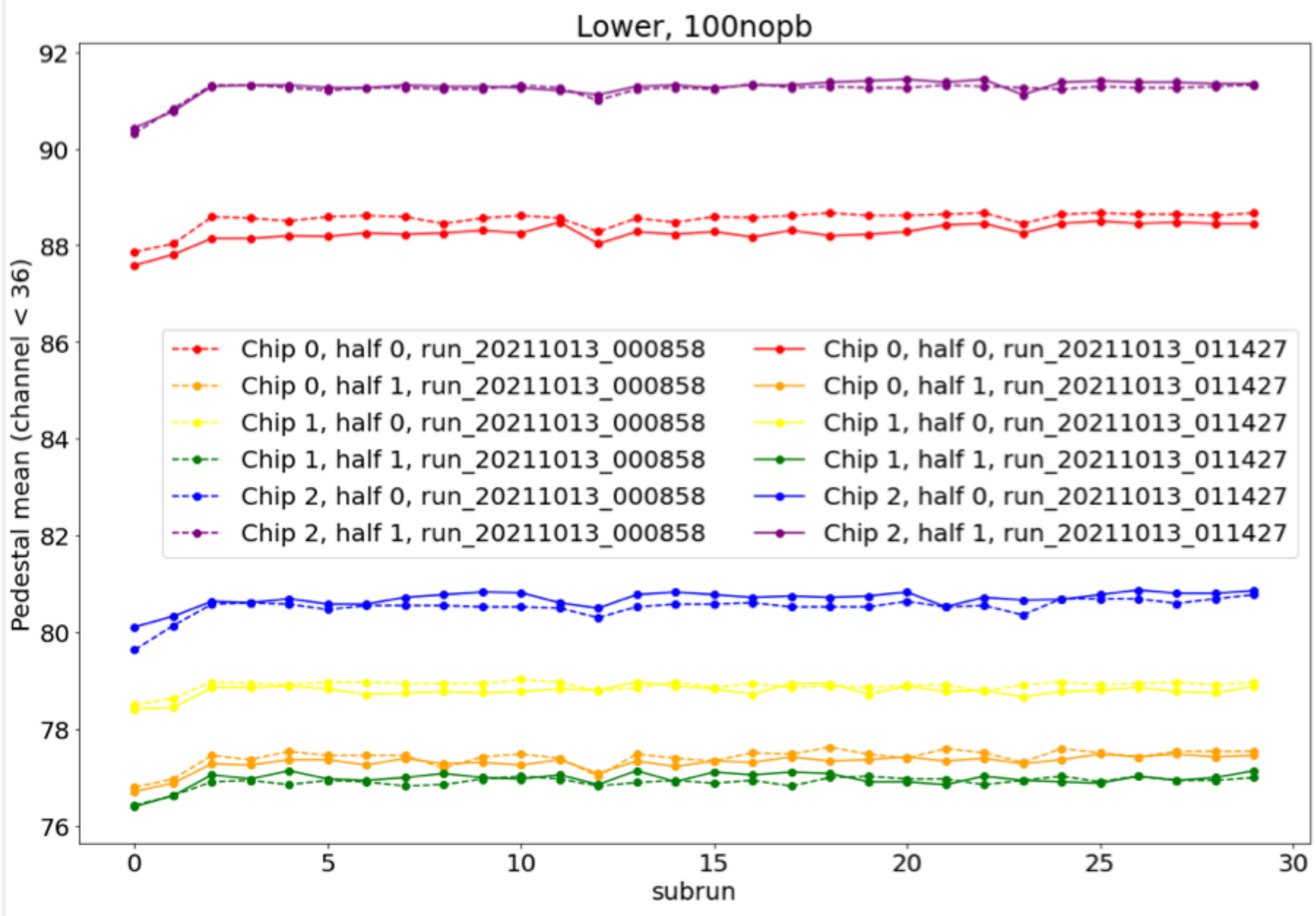


Source: p.33 in HGCROC3_Spec_Working_Document_v2.0.pdf

Glossary

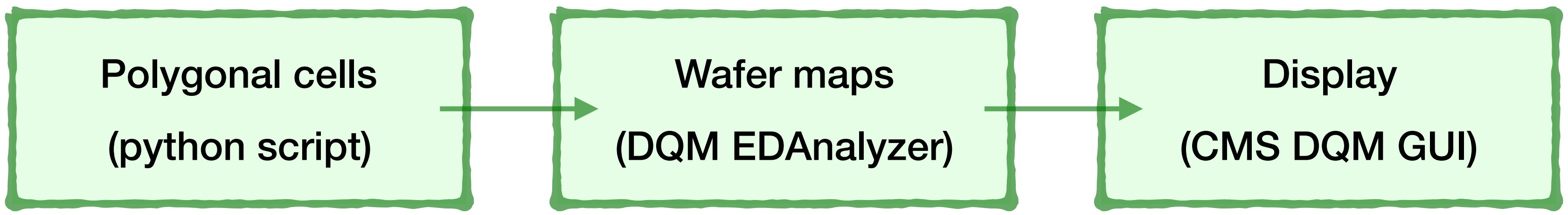
Abbreviation	Original words	Meaning
ADC	Analog-to-digital converter	Used as energy unit for digitized signal
TDC	Time-to-digital converter	Used as unit for timing information
LSB	Least significant bit	The smallest level that ADC or TDC can convert
ToT	Time over threshold	A span of time when signal is over energy threshold
ToA	Time of arrival	An instant of time when signal is over energy threshold
TcTp	ToT-complete and ToT-in-progress	Flags for three time intervals of signals (before rising, over threshold, and after declining)

Instability of pedestal



Source: https://indico.cern.ch/event/1142454/contributions/4794019/attachments/2411882/4127370/20220322_BeamTest_GeliangLiu.pdf

Displaying wafer maps



- Prepare polygonal cells using pyRoot
  - Define polygonal bins as TGraph objects in a root file
  - Tool: https://github.com/ywkao/hexagonal_histograms
- Create wafer maps in DQM EDAnalyzer
  - Monitor element of TH2Poly is necessary → PR on CMSSW is created (#41932)
  - Declare TH2Poly monitor elements and load the polygonal bins
  - Store the polygonal histograms in an output DQM root file
  - A DQM module tested in a private cmssw branch: PlaygroundDQMEDAnalyzer
- Display on a CMS DQM GUI
  - An online DQM GUI is built
    - Instructions on cms twiki, DQMGuiForUsers#How
    - Layout and rendering plugin for HGCAL system tests are set
    - Recognition of TH2Poly is implemented → PR on dqmgui_prod is created (#14)
  - Upload the DQM root file & monitor plots on the DQM GUI