SπRITROOT

Genie Jhang, Yassid Ayyad and Jung Woo Lee
for the SπRIT collaboration
2015. 6. 5

Big picture

- FairSoft
  - All the necessary packages collected to run FairRoot
  - Designed to be installed on both Linux and OS X
  - Included packages:
    - ✳ gtest, gsl, boost, Pythia6, Pythia8, HepMC, GEANT3, GEANT4, XRootD, Pluto, ROOT, VGM, VMC, Millepede, ZeroMQ, Protocol Buffers, nanomsg
  - RAVE, CLHEP, and GENFIT2 packages added for SπRITROOT (customized version)

- FairRoot
  - A framework containing base classes for running simulation, reconstruction and analysis

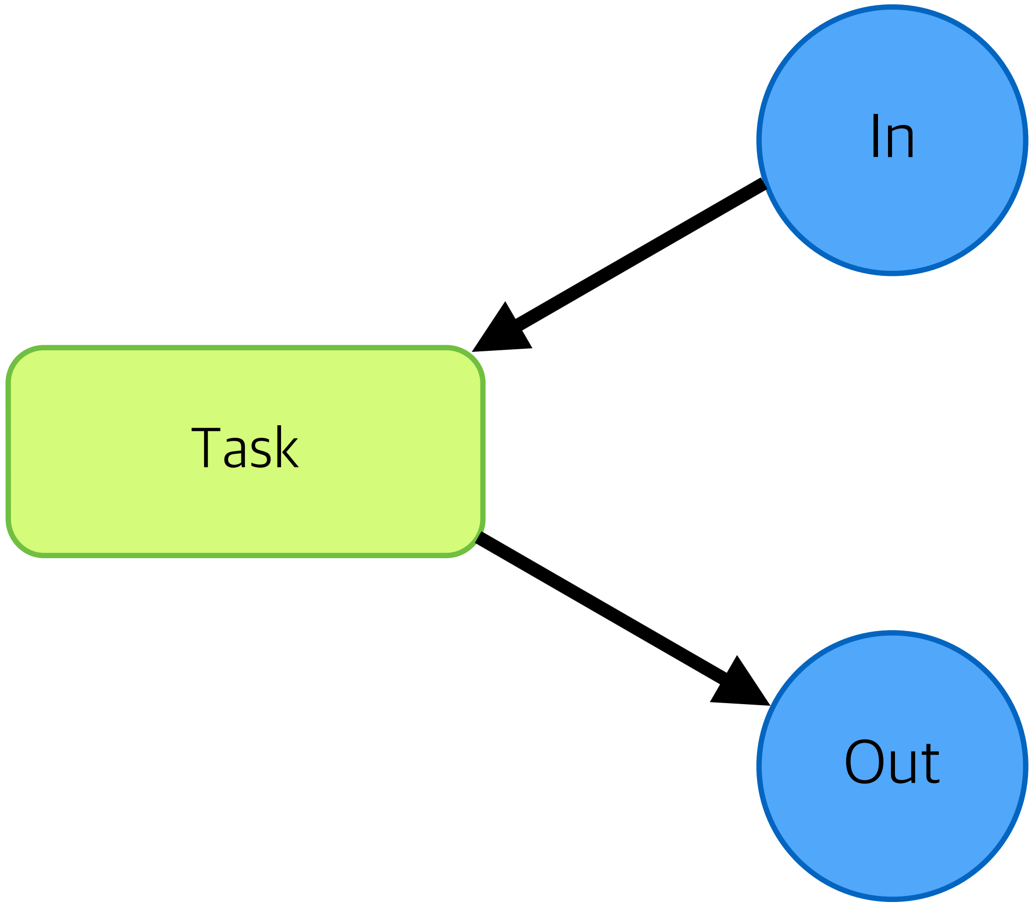
- SπRITROOT
  - A framework containing specific modules for SπRIT experiment on top of FairRoot
  - Composed of task-based modules, TGeo geometry and steering macro
  - SπRITROOT is written by following the structure of FOPIROOT[1)]

---

1) M. Ball et al., Technical Design Study for the PANDA Time Projection Chamber, http://arxiv.org/abs/1207.0013

Task-based Module

- Easy to turn on and off.
- Easy to debug and maintain.

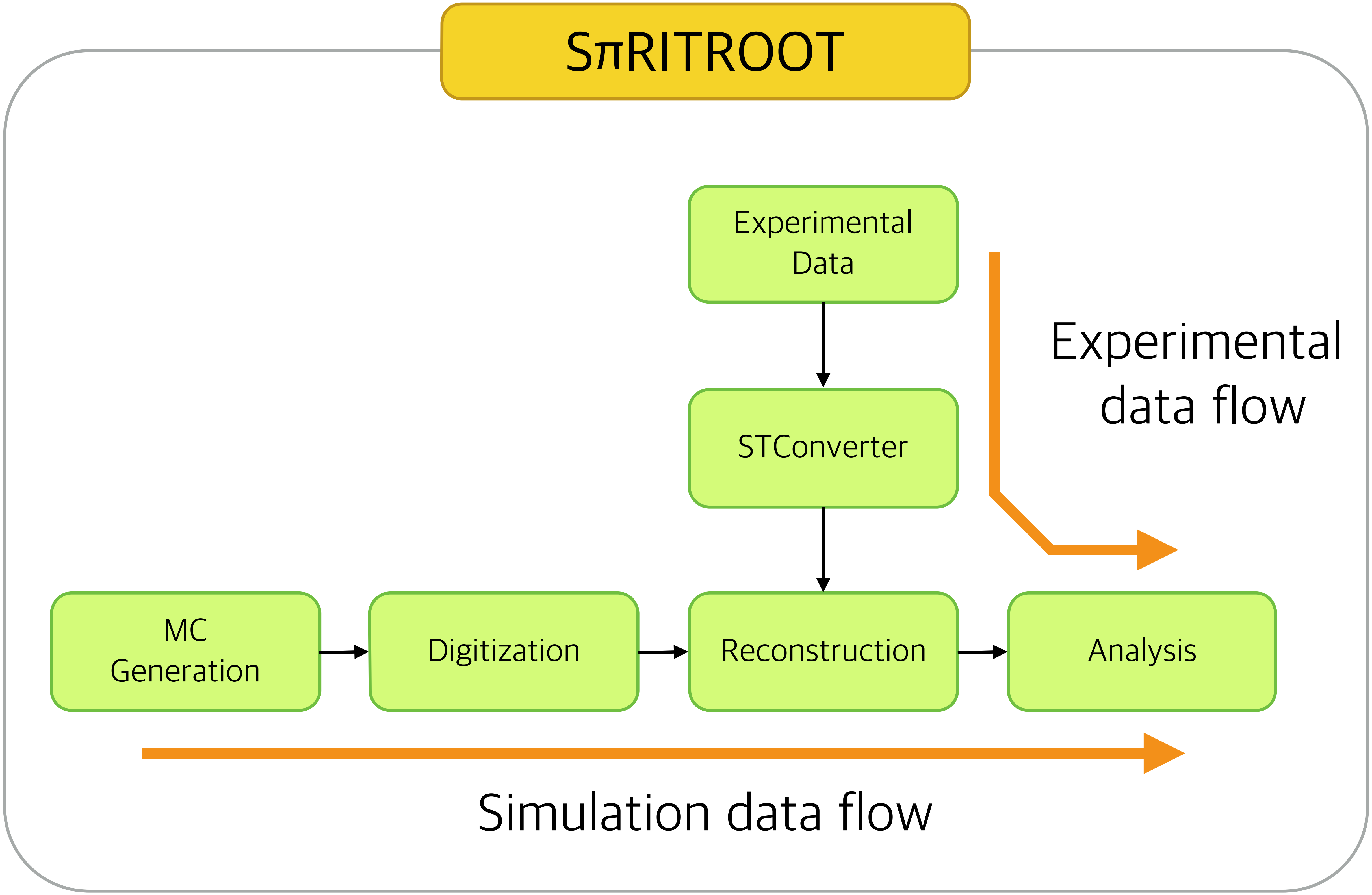


- ASCII
- GRAW
- ROOT
- An object on memory (TClonesArray*)

- ROOT
- An object on memory (TClonesArray)

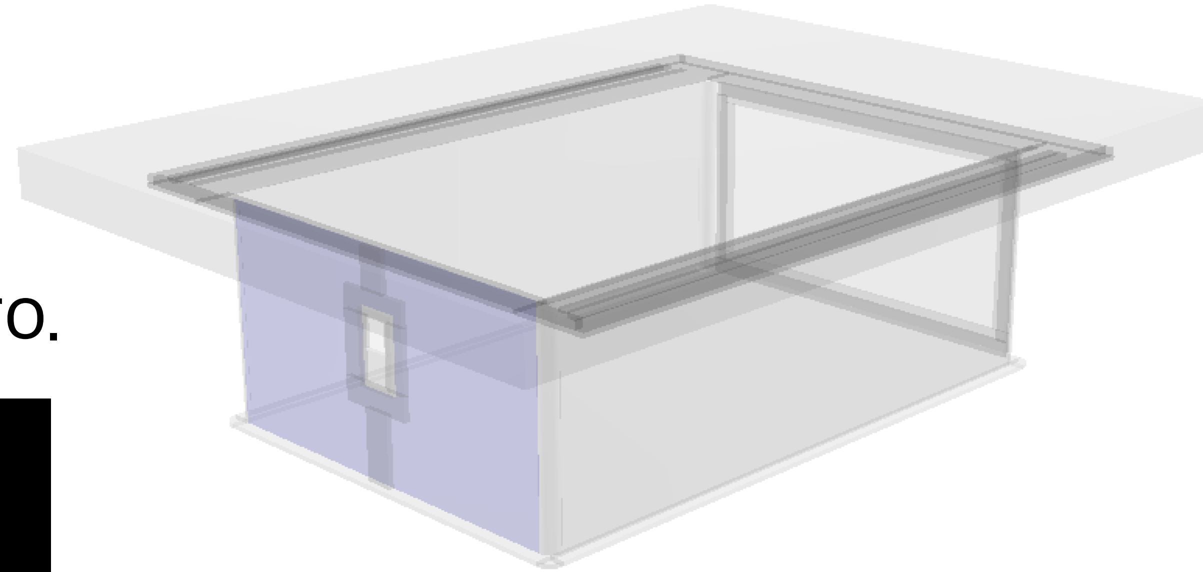
* TClonesArray is a container class provided in ROOT which can be stored in ROOT file.

Schematics of SπRITROOT



MC Generation

- Detector geometry used in MC is written with TGeo classes in ROOT.
  - https://github.com/SpiRIT-Collaboration/SPiRITROOT/blob/develop/SPiRIT/geometry/geomSPiRIT.C



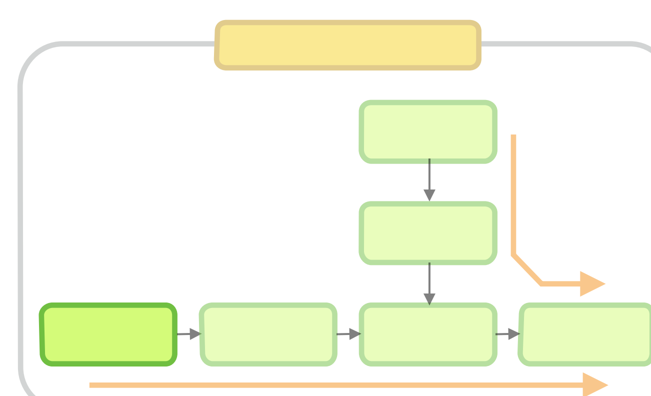
- GEANT4 is used to generate MC data.
- Program runs with a ROOT simple macro.

```
// -----   Create geometry   ----------------------------------------------
FairModule* cave = new FairCave("CAVE");
cave -> SetGeometryFileName("cave_vacuum.geo");
FairDetector* spirit = new STDetector("STDetector", kTRUE);
spirit -> SetGeometryFileName("geomSPiRIT.root");
// -----   Create and set magnetic field   --------------------------------
FairConstField *fMagField = new FairConstField();
fMagField -> SetField(0., 5., 0.); // in kG
fMagField -> SetFieldRegion(-90.275,90.2752,-95.55/2,95.55/2,-104.82/2,104.82/2);
```

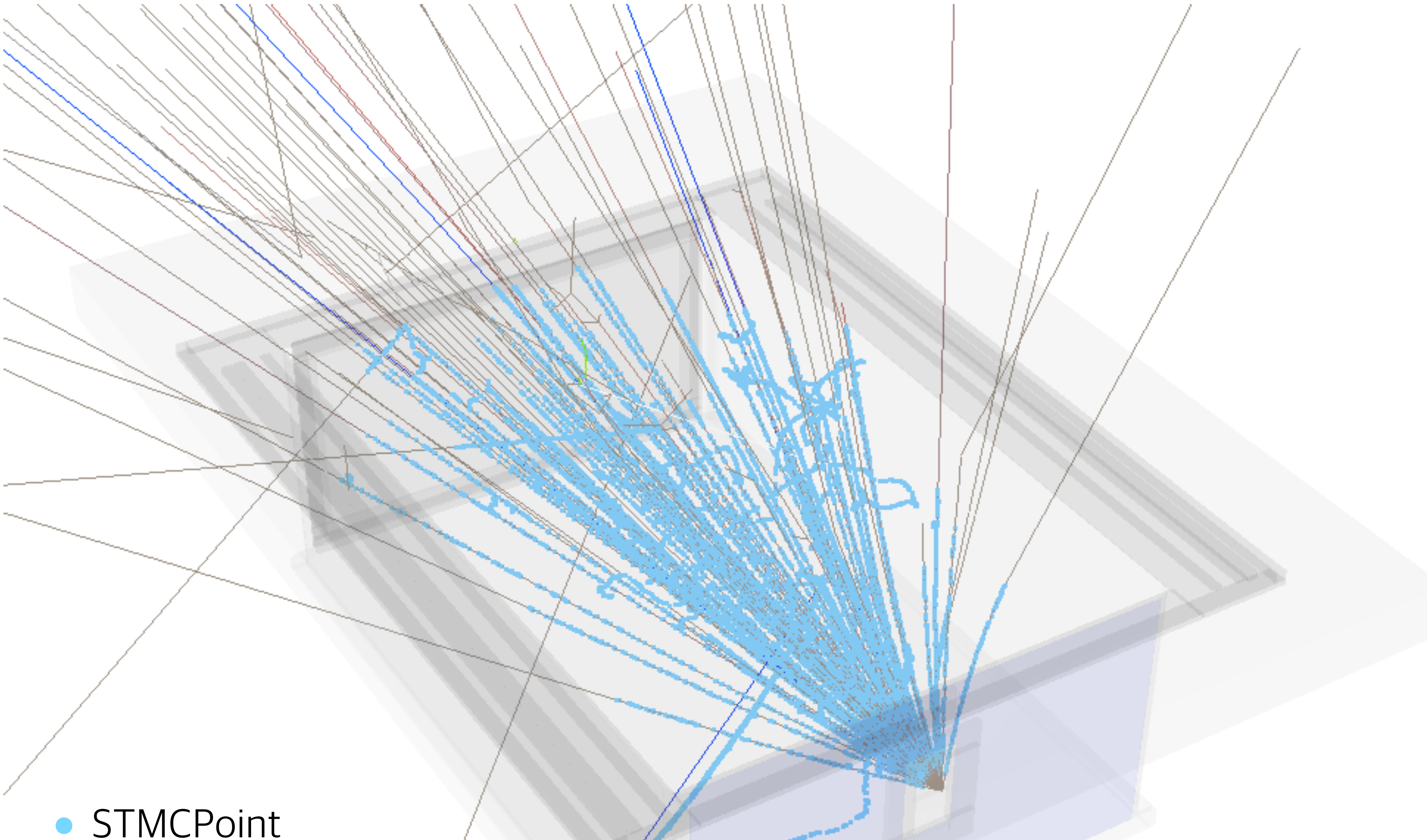
```
// -----   Create PrimaryGenerator   -----------------------------------------
STSimpleEventGenerator* gen = new STSimpleEventGenerator("../input/GEN_singleTrack.sgen");
gen -> SetPrimaryVertex(0, -21.33, -3.52);

FairPrimaryGenerator* primGen = new FairPrimaryGenerator();
primGen->AddGenerator(gen);
```

Output → STMCPoint

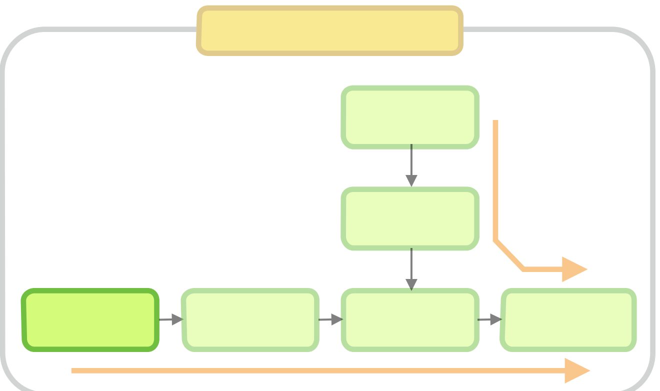
```
// -----   Create simulation run   ----------------------------------------
FairRunSim* run = new FairRunSim();
run -> SetName("TGeant4");            // Transport engine
run -> SetOutputFile(outFile);        // Output file
run -> SetMaterials("media.geo");
run -> AddModule(cave);
run -> AddModule(spirit);
run -> SetField(fMagField);
run -> SetGenerator(primGen);
run -> Init();
run -> Run(gen -> GetNEvents());
```



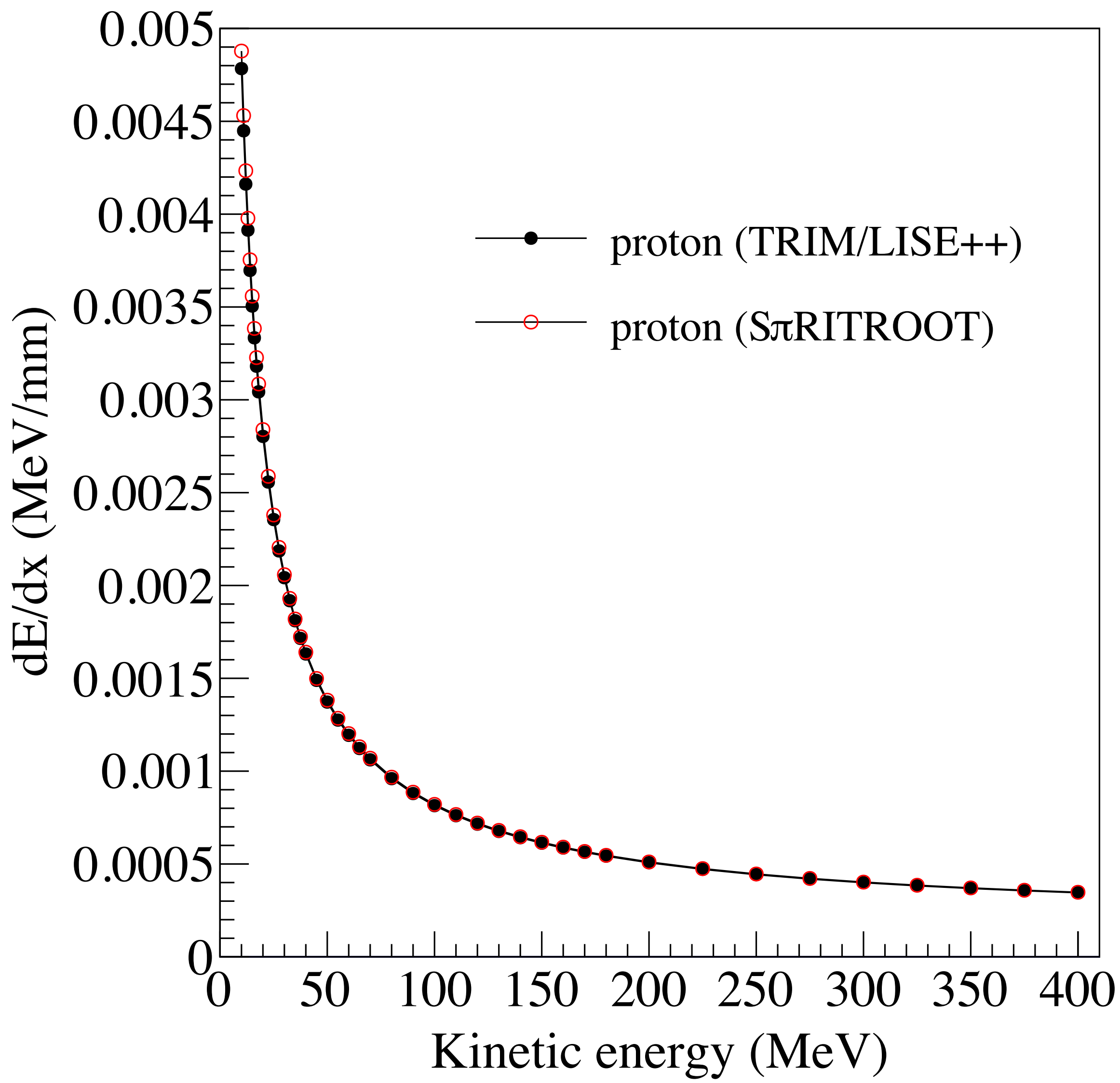
MC Generation

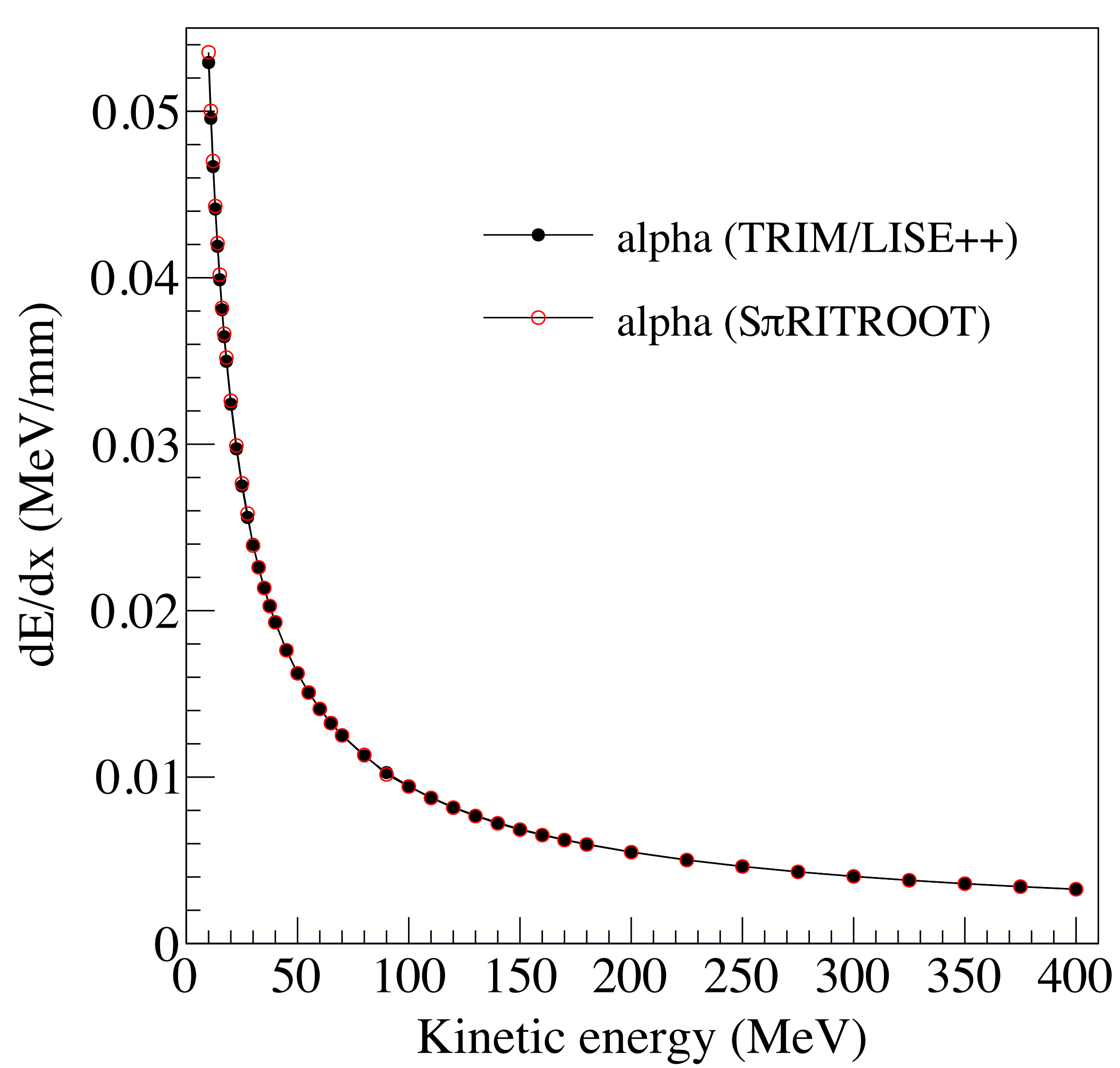


- STMCPoint
  - position vector
  - energy loss
  - hit time



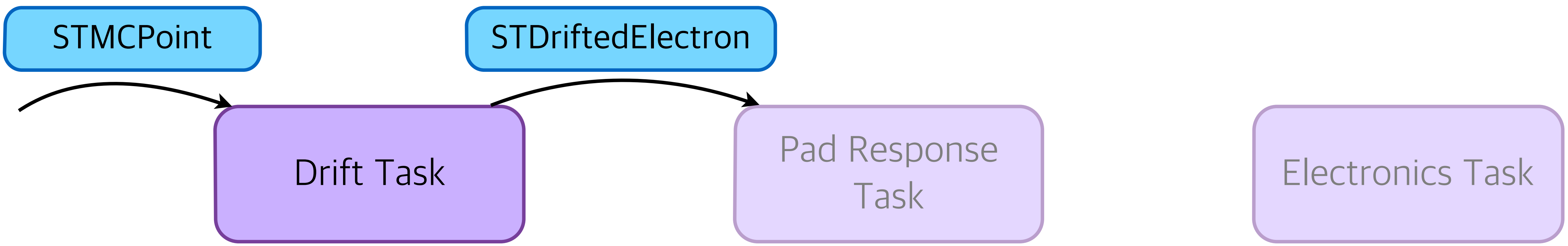
Energy loss

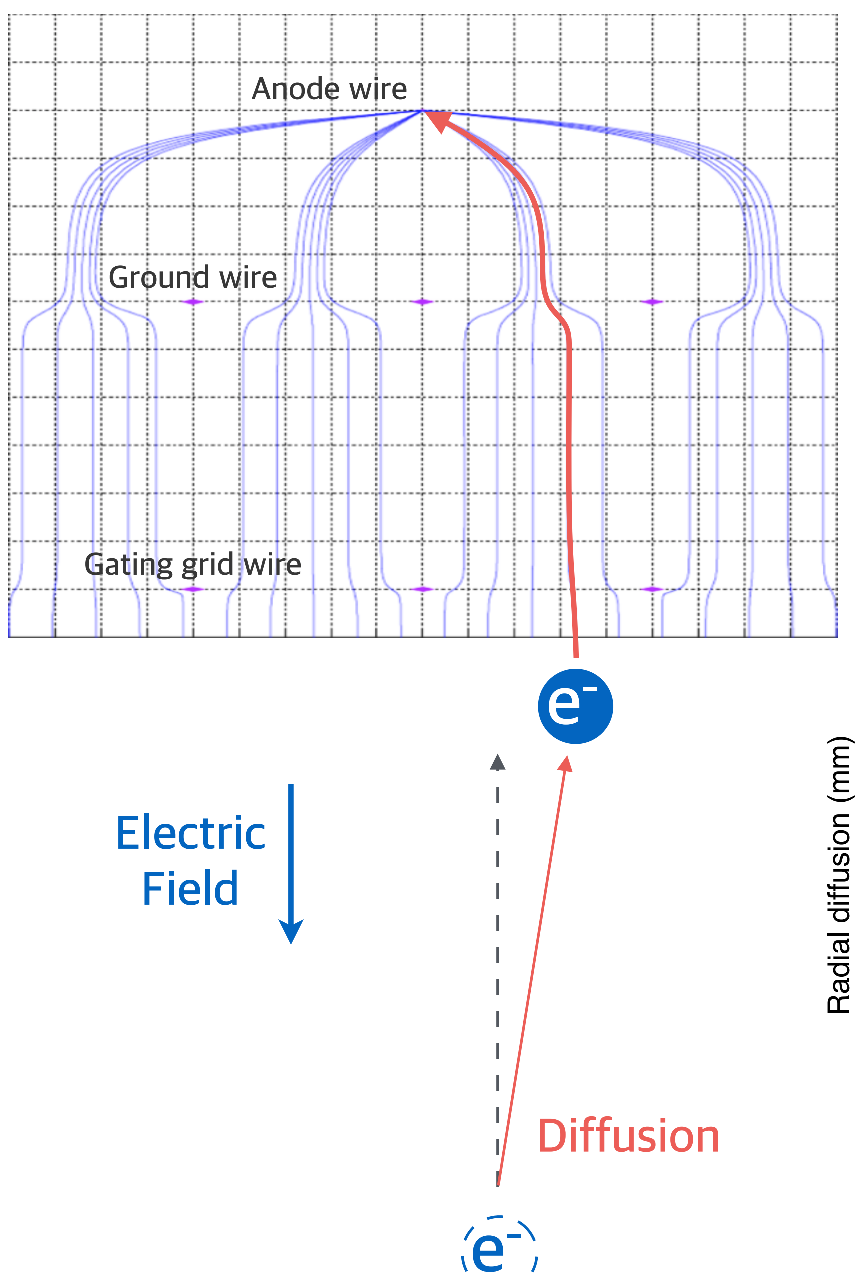




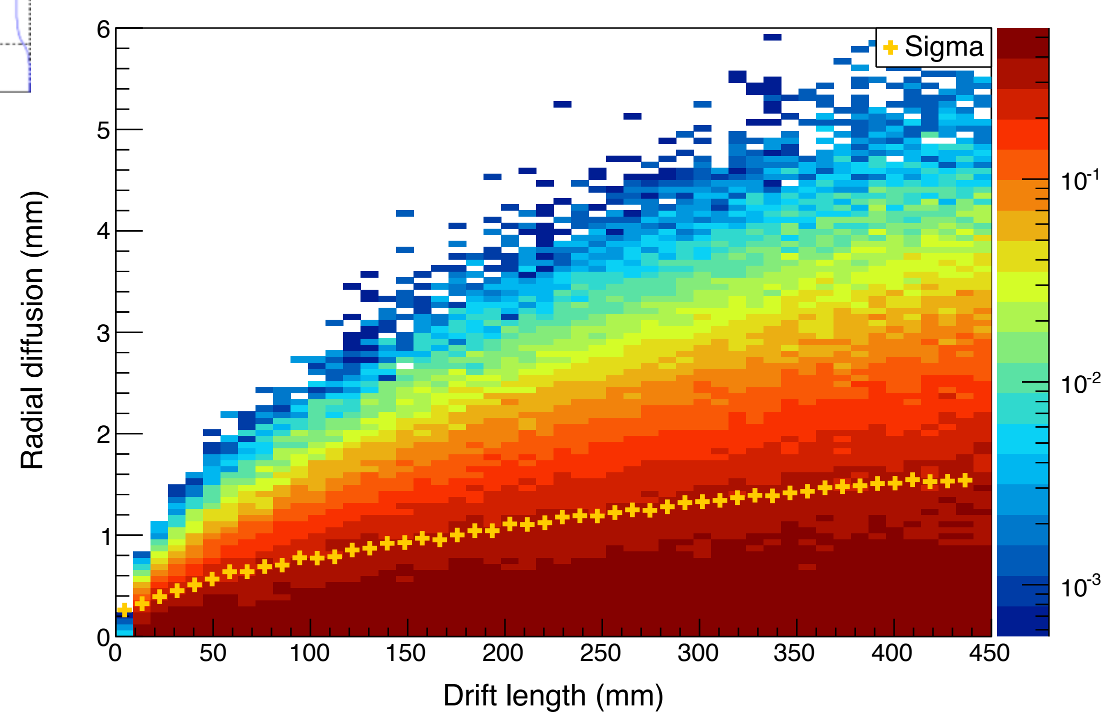
- In both cases GEANT4 result well matches to TRIM/LISE++ result.
- Maximum % difference of dE/dx – proton: 1.97 % , alpha: 1.16 %

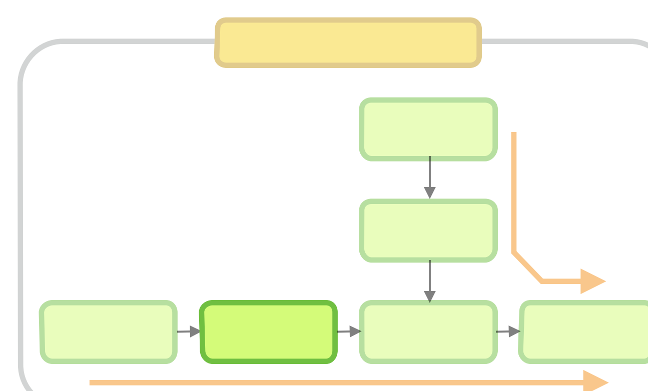
Digitization



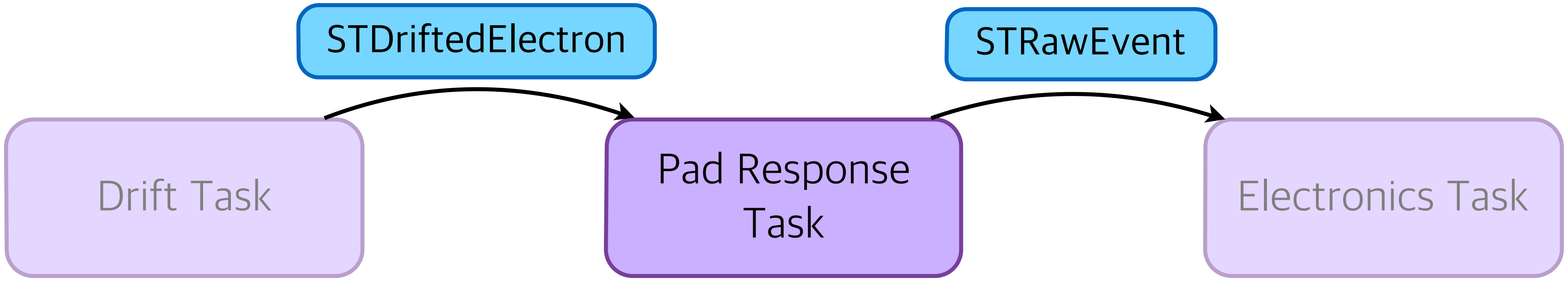


- Using the diffusion constants calculated with Garfield, calculate the drift time corresponding to the distance from ground wire to the hit point.
- Determine the nearest anode wire to apply the pad response function.



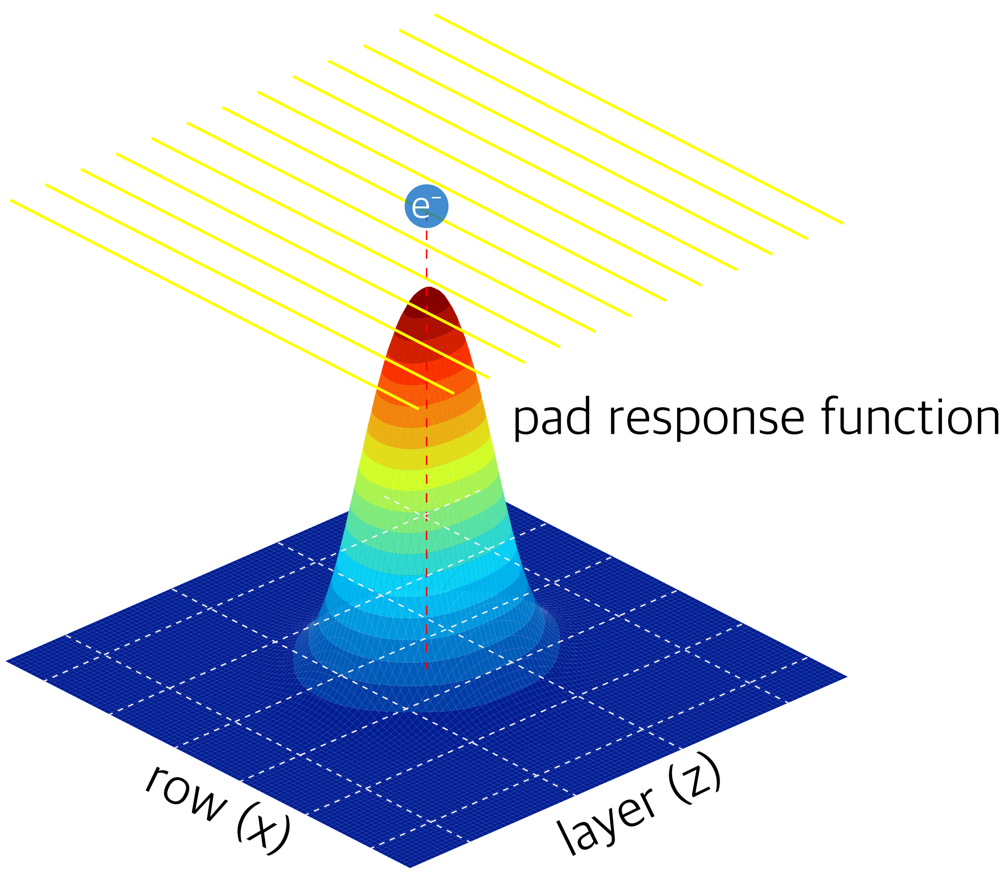


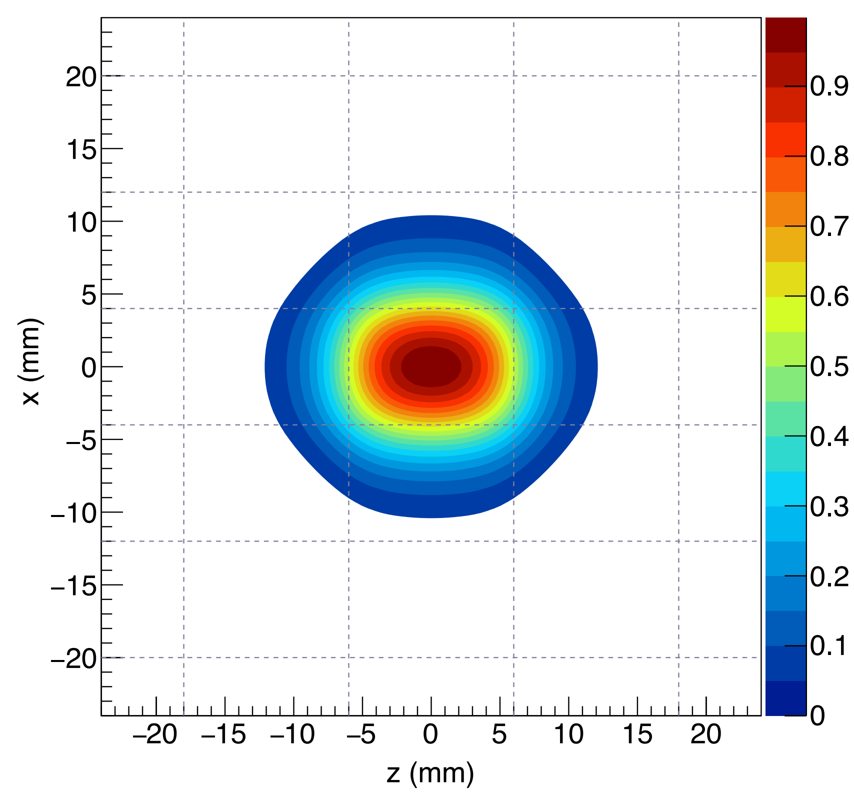
Digitization

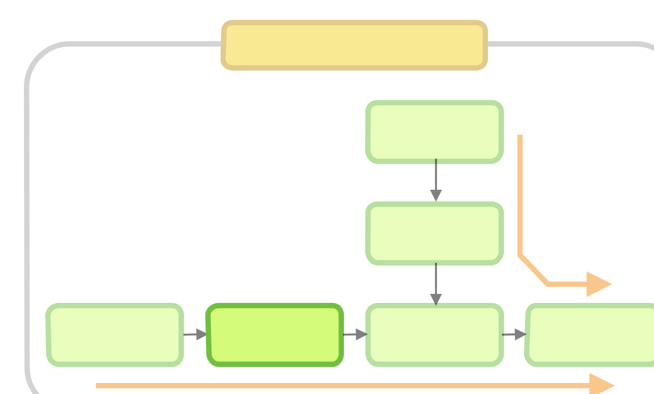


- Pad response function describes the induced charge by the avalanche electrons. The function is calculated using Gatti distribution.

$$P(\lambda) = \frac{K_1}{K_2\sqrt{K_3}}\left[\arctan\sqrt{K_3}\tanh\left(K_2\left(\lambda+\frac{w}{2h}\right)\right) - \arctan\sqrt{K_3}\tanh\left(K_2\left(\lambda-\frac{w}{2h}\right)\right)\right]^{1)}$$

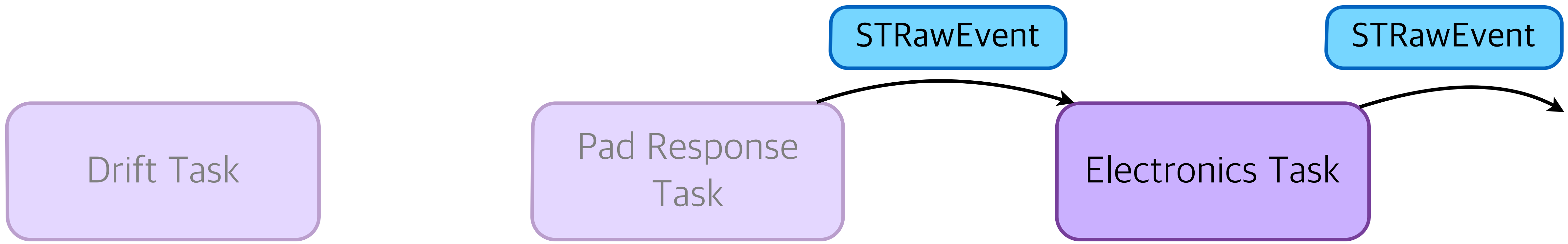






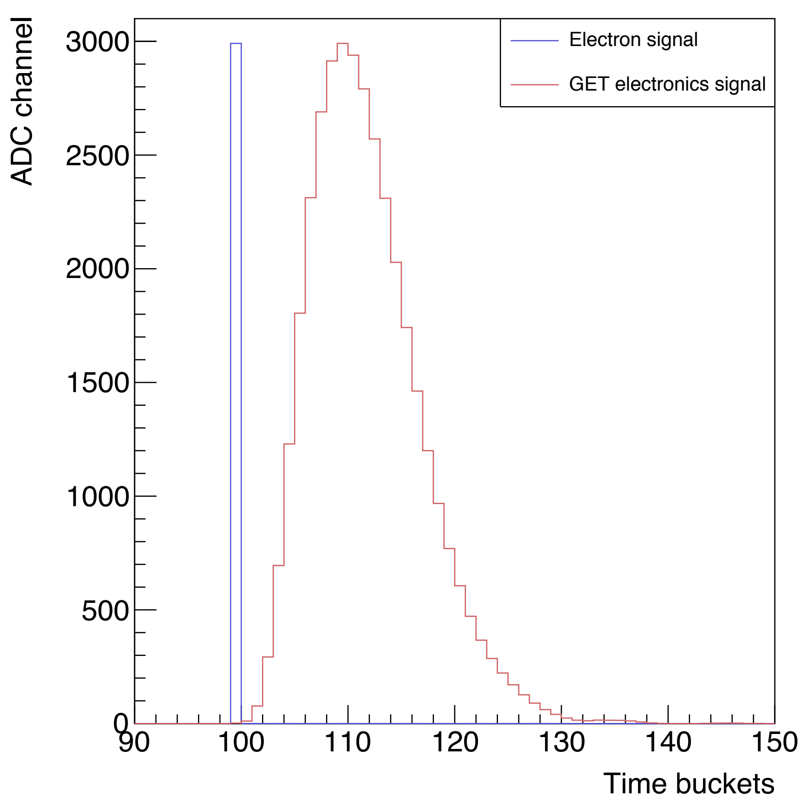
1) NIM **A270** (1998) 602

Digitization

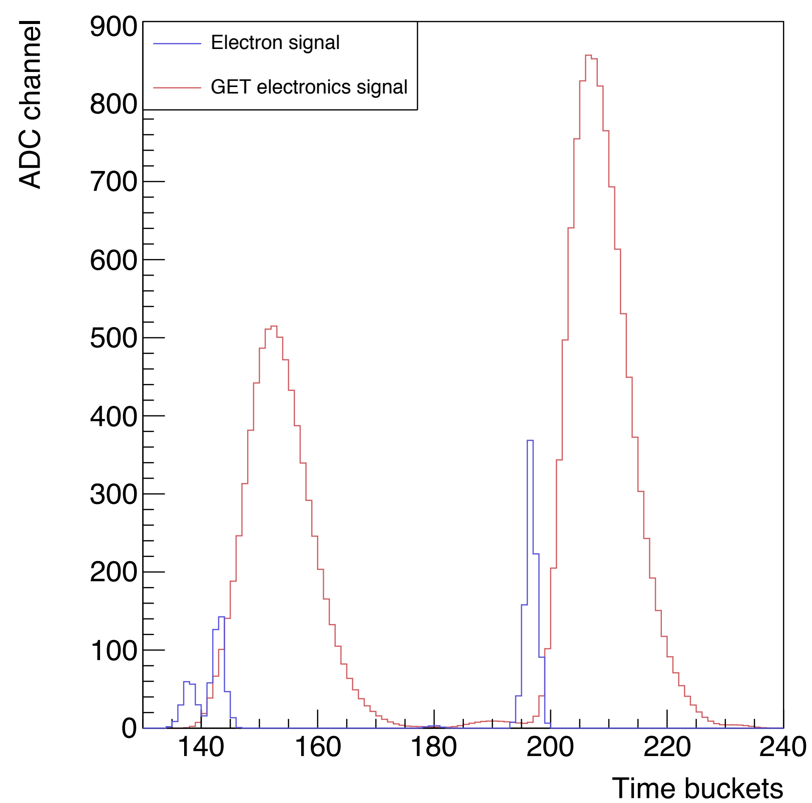


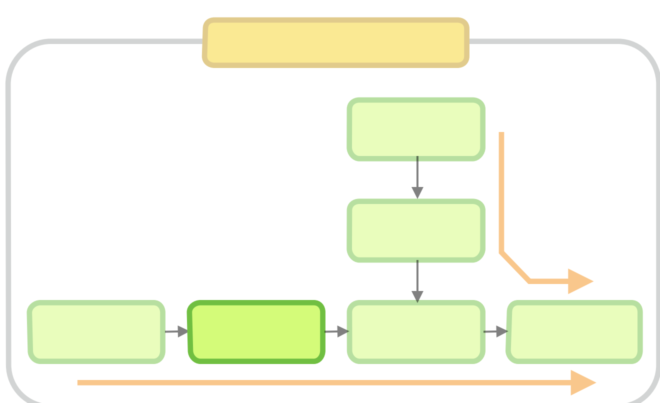
- Average pulse shape is obtained from HIMAC pulser data.
- Pulse height is set to be the same as the number of amplified electrons.

HIMAC pulser data

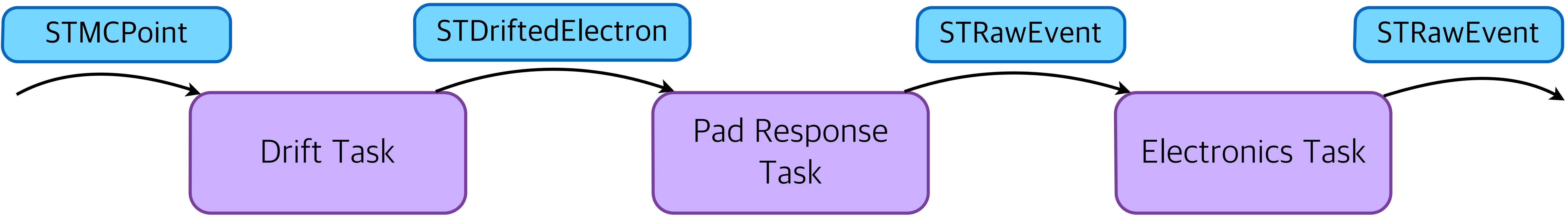


Simulation result



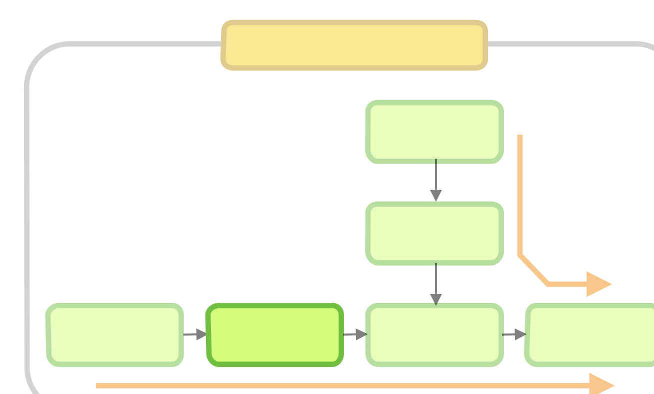


Digitization

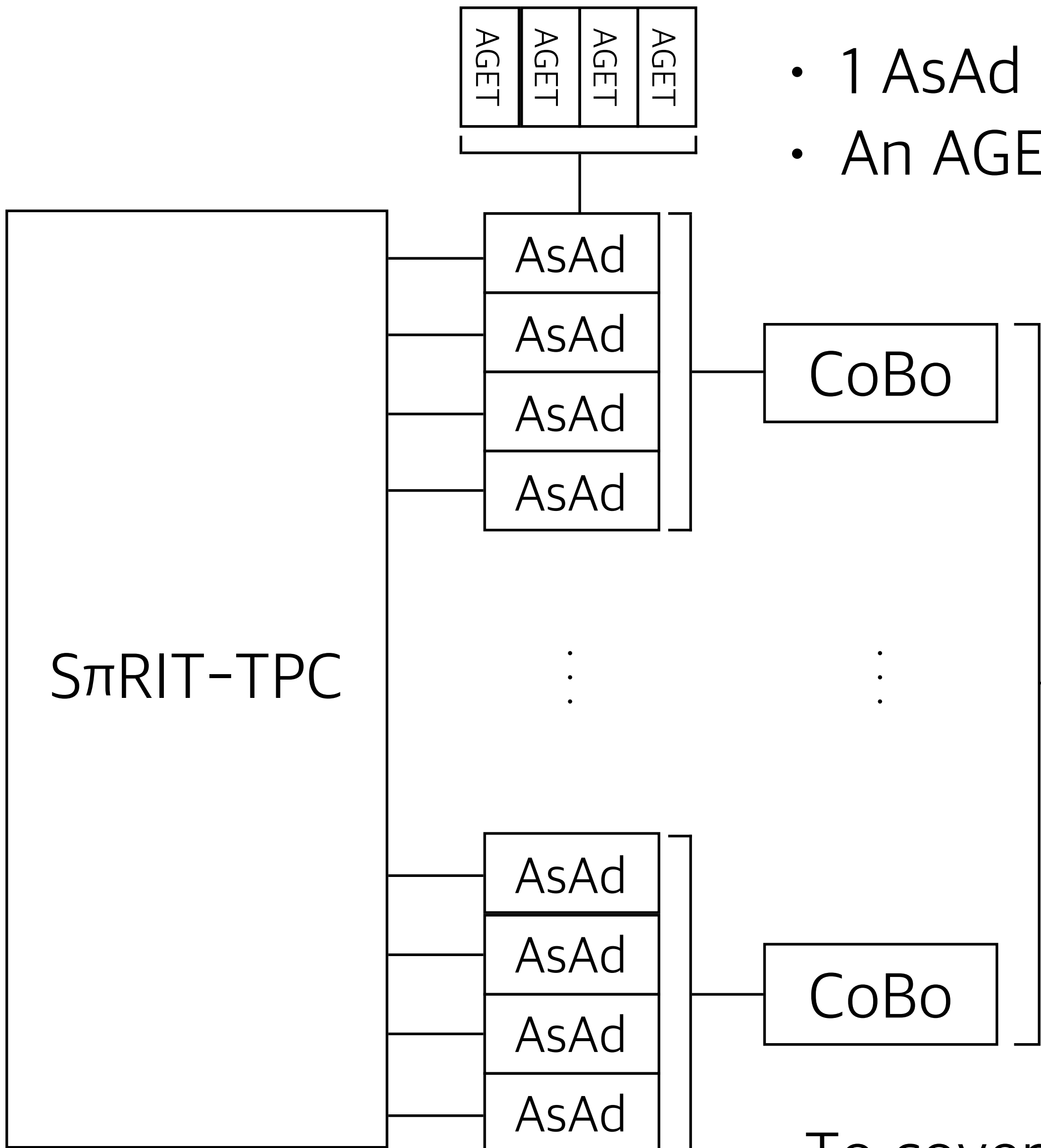


- Code

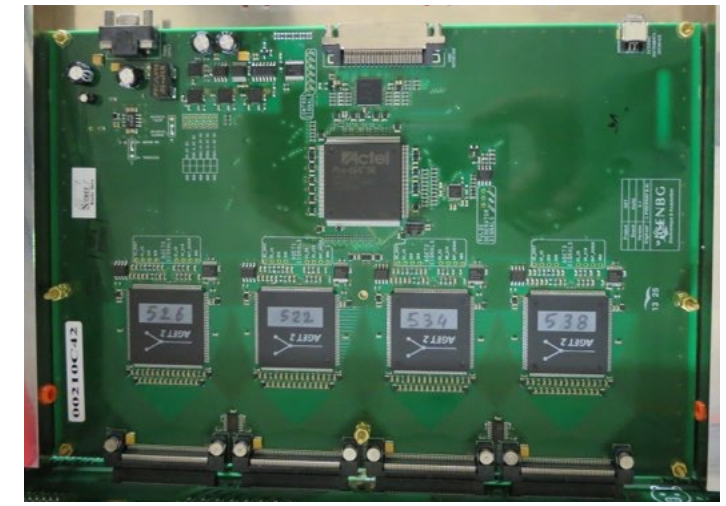
```
STDriftTask* drift = new STDriftTask();
             drift -> SetInputPersistance(kTRUE);

STPadResponseTask* padResponse = new STPadResponseTask();
                   padResponse -> SetInputPersistance(kTRUE);
                   padResponse -> AssumeGausPRF();

STElectronicsTask* electronics = new STElectronicsTask();
                   electronics -> SetInputPersistance(kTRUE);


FairRunAna* fRun = new FairRunAna();
fRun -> SetInputFile(mcFile.Data());
fRun -> SetOutputFile(digiFile.Data());
fRun -> AddTask(drift);
fRun -> AddTask(padResponse);
fRun -> AddTask(electronics);
fRun -> Init();
fRun -> Run(0,0);
```



Experimental Data (Setup)

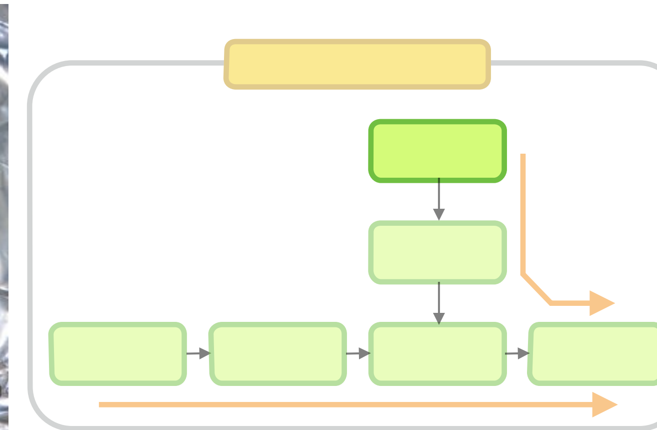


- 1 AsAd has 4 AGET chips.
- An AGET chip handles 64 channels.



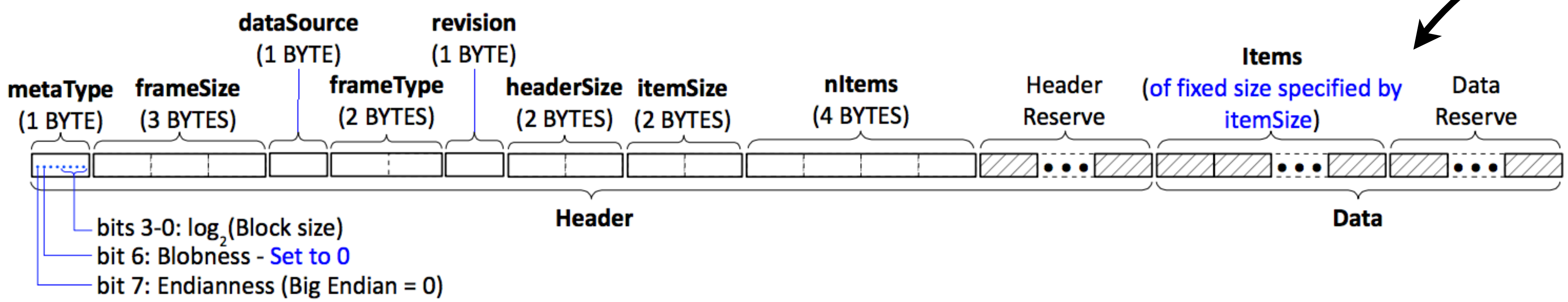
- NARVAL is used for merging data from multiple CoBos.

- To cover 108 x 112 = 12,096 pads, we use
  - 48 AsAd boards
  - 12 Cobos



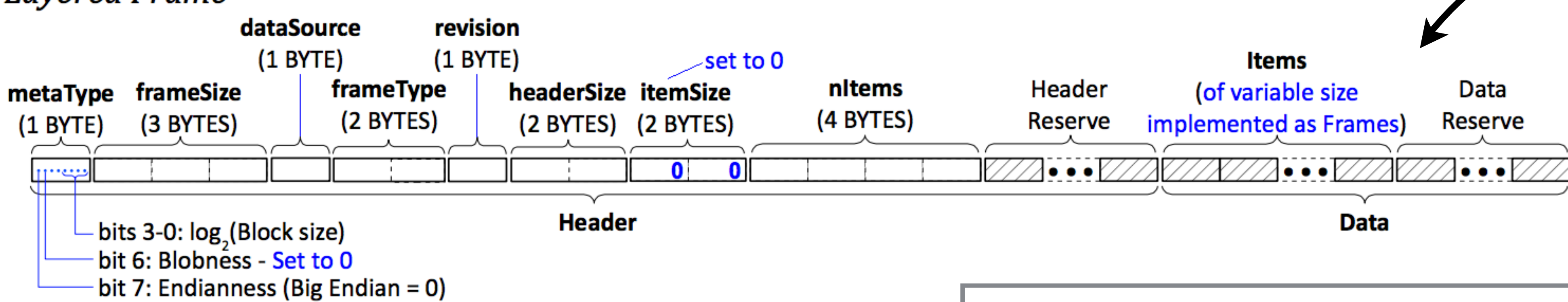
Experimental Data (GRAW file, binary)

Basic Frame



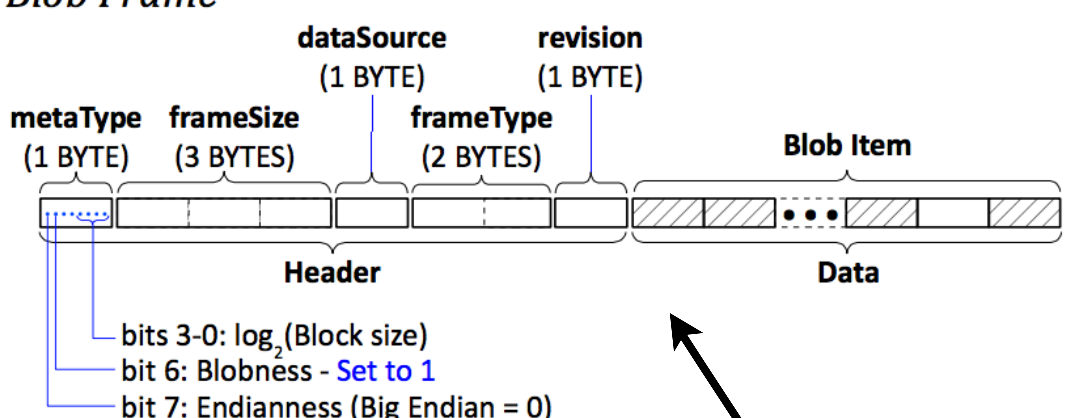
One AsAd data

Layered Frame

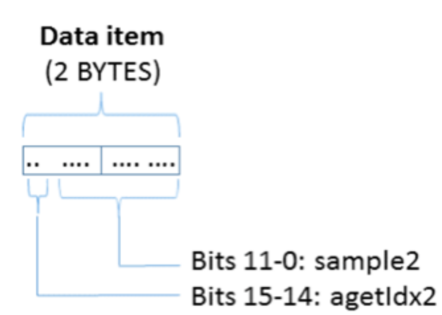


Multiple AsAds merged data

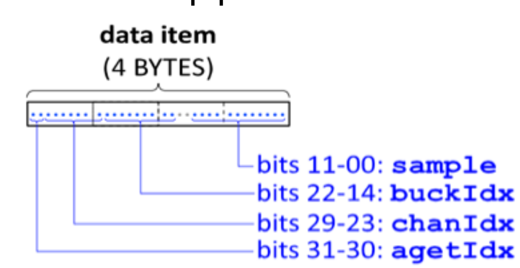
Blob Frame



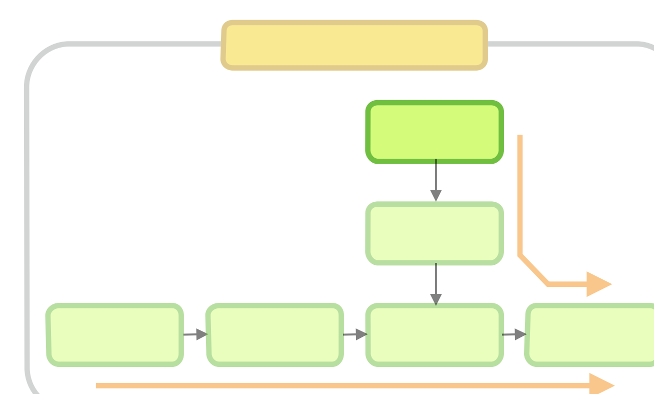
- Full readout mode



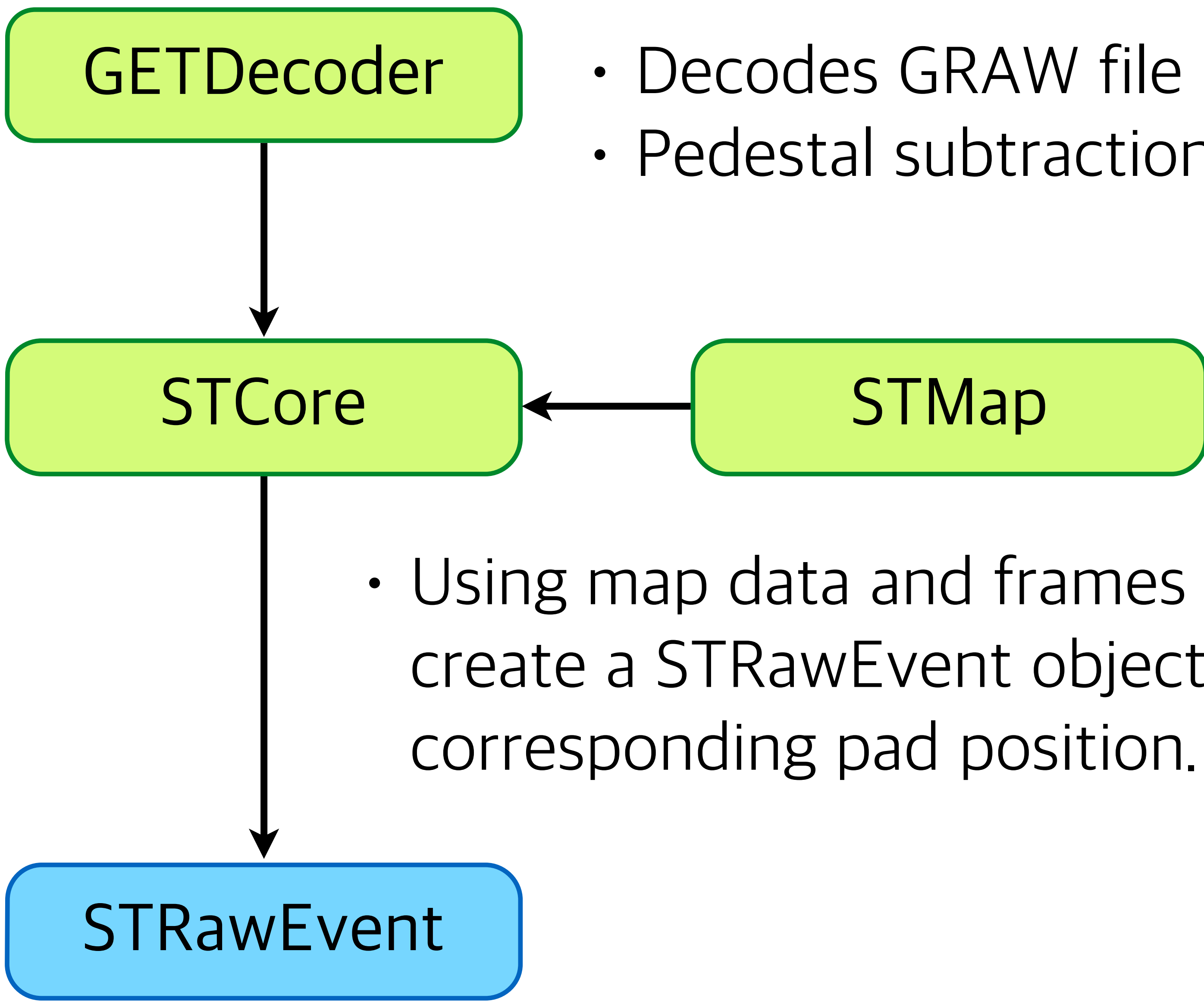
- Zero-suppresion mode



Only for additional information



STConverter

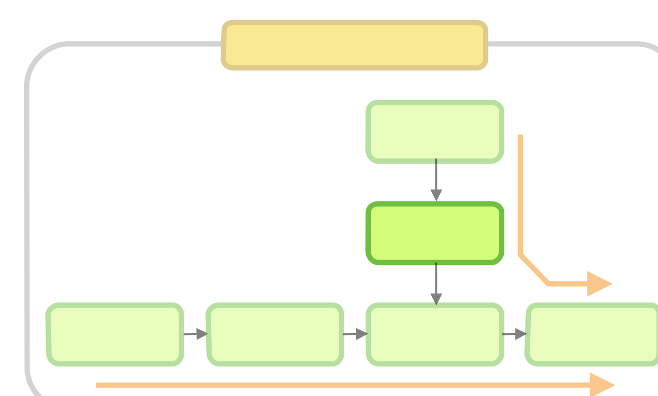


- Decodes GRAW file into a C++ object
- Pedestal subtraction using the nearest FPN channels

- Using map data and frames from GETDecoder, create a STRawEvent object all channels mapped corresponding pad position.

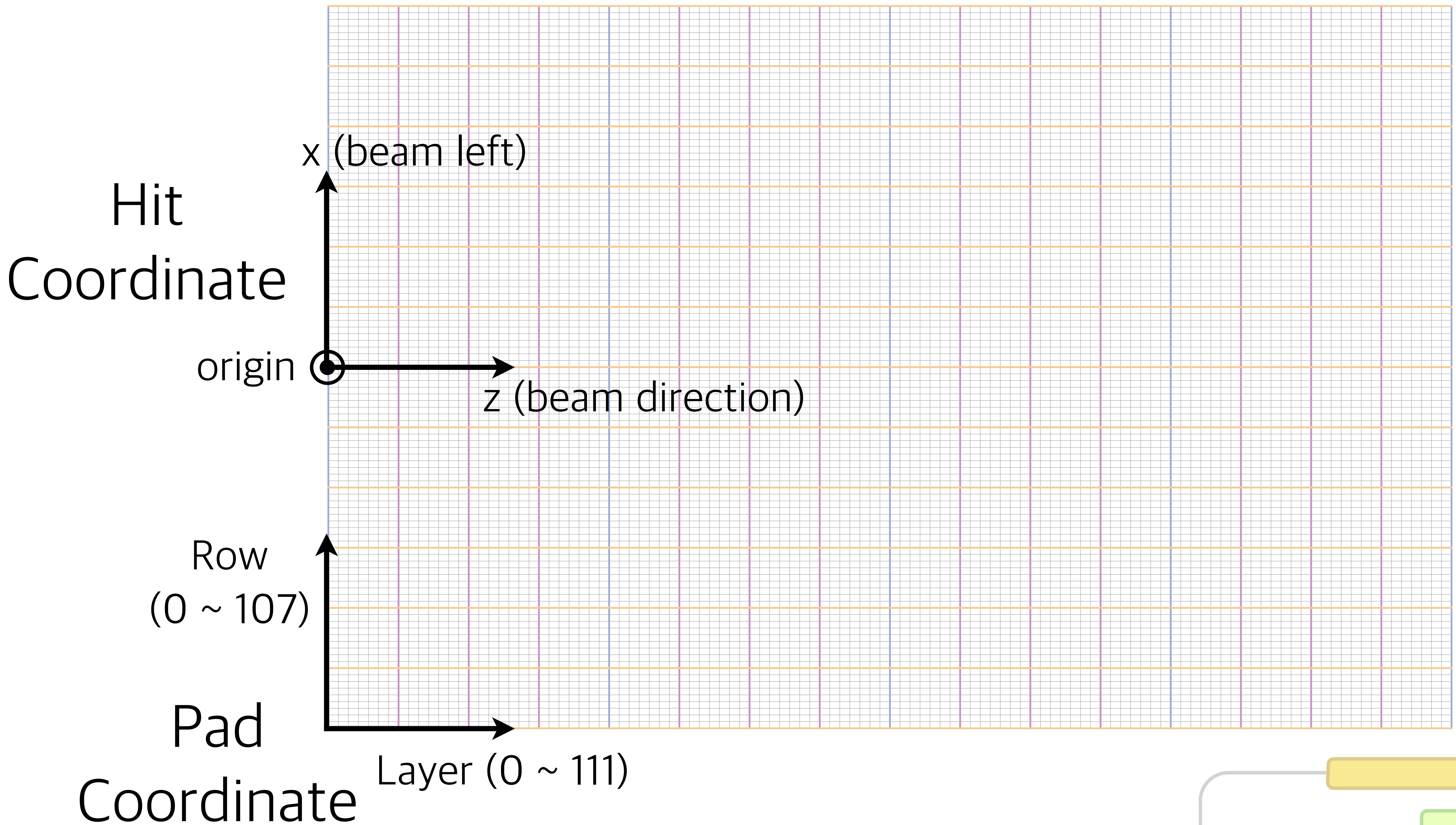
- Code

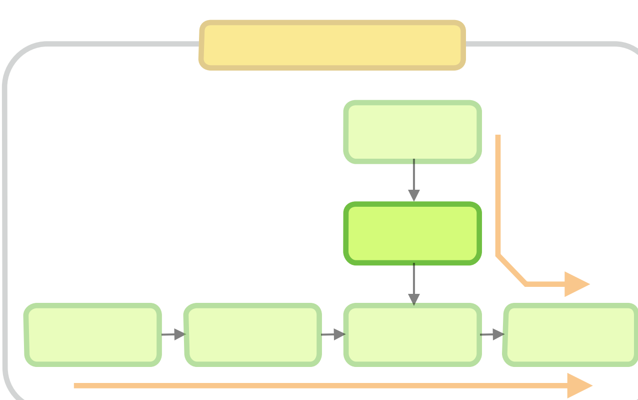
```
STDecoderTask *decoderTask = new STDecoderTask();
decoderTask -> AddData("SETGRAWFILE.graw");
decoderTask -> SetFPNPedestal(50);
decoderTask -> SetNumTbs(512);
decoderTask -> SetPersistence();
run -> AddTask(decoderTask);
```



STConverter

- Coordinates convention (Top view)

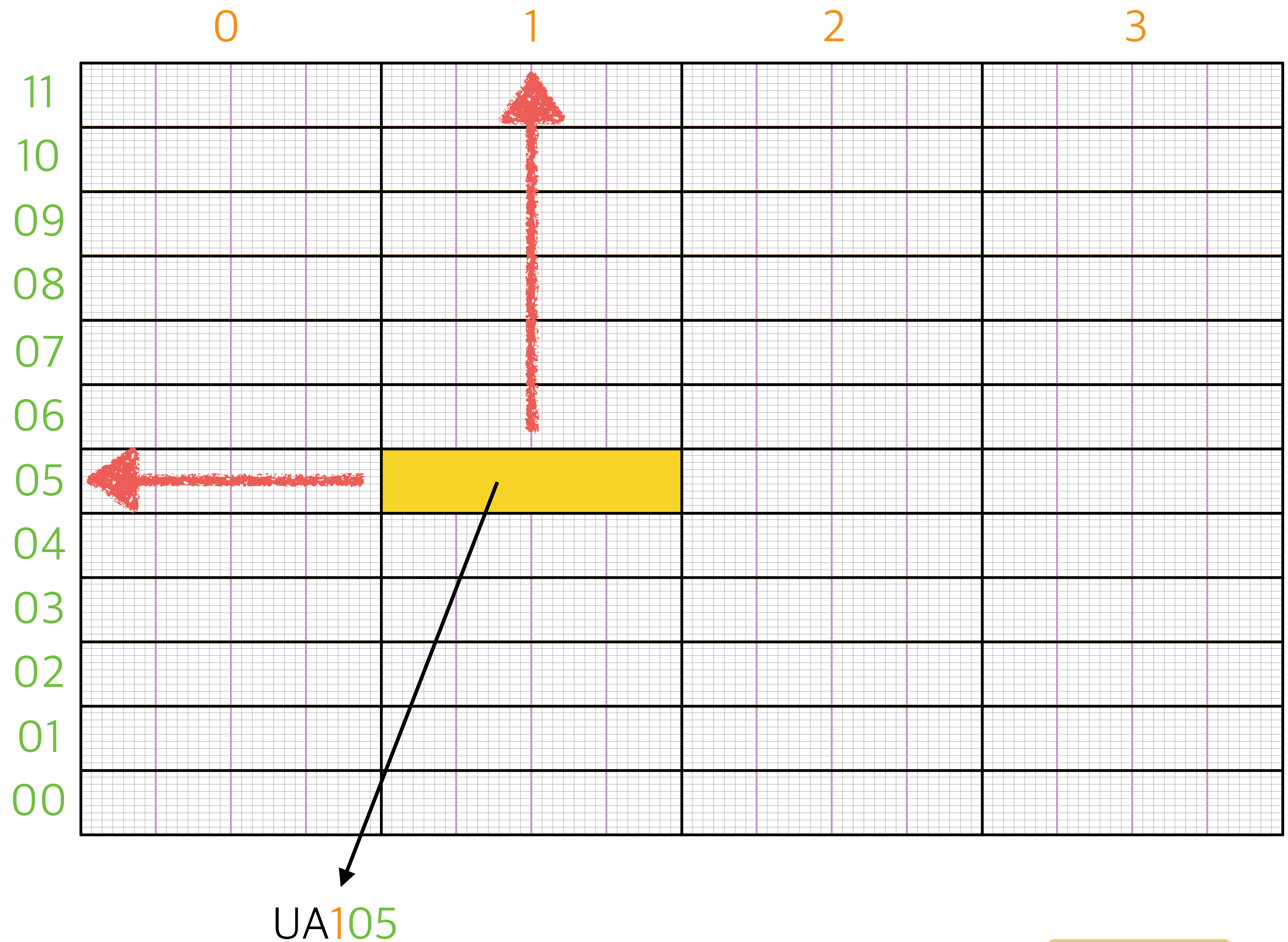




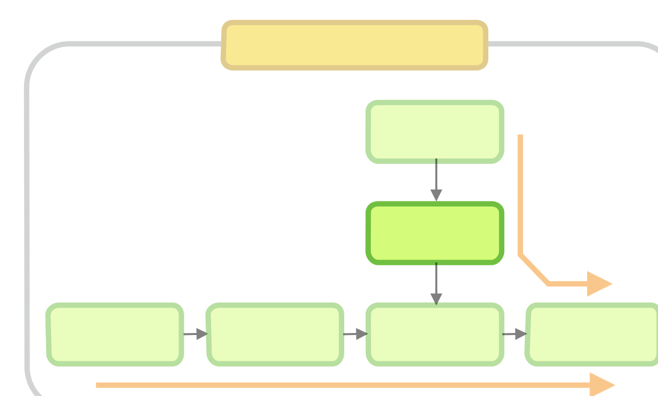
STConverter

- AsAd mapping

# UAIdx	CoBoIdx	AsAdIdx
000	0	0
001	0	1
002	0	2
003	0	3
004	1	0
005	1	1
006	1	2
007	1	3
008	2	0
009	2	1
010	2	2
011	2	3
100	3	0
101	3	1
102	3	2
103	3	3
104	4	0
105	4	1
106	4	2
107	4	3
108	5	0
109	5	1
110	5	2
111	5	3
200	6	0
201	6	1
202	6	2
203	6	3
204	7	0
205	7	1
206	7	2
207	7	3
208	8	0
209	8	1
210	8	2
211	8	3
300	9	0
301	9	1
302	9	2
303	9	3
304	10	0
305	10	1
306	10	2
307	10	3
308	11	0
309	11	1
310	11	2
311	11	3



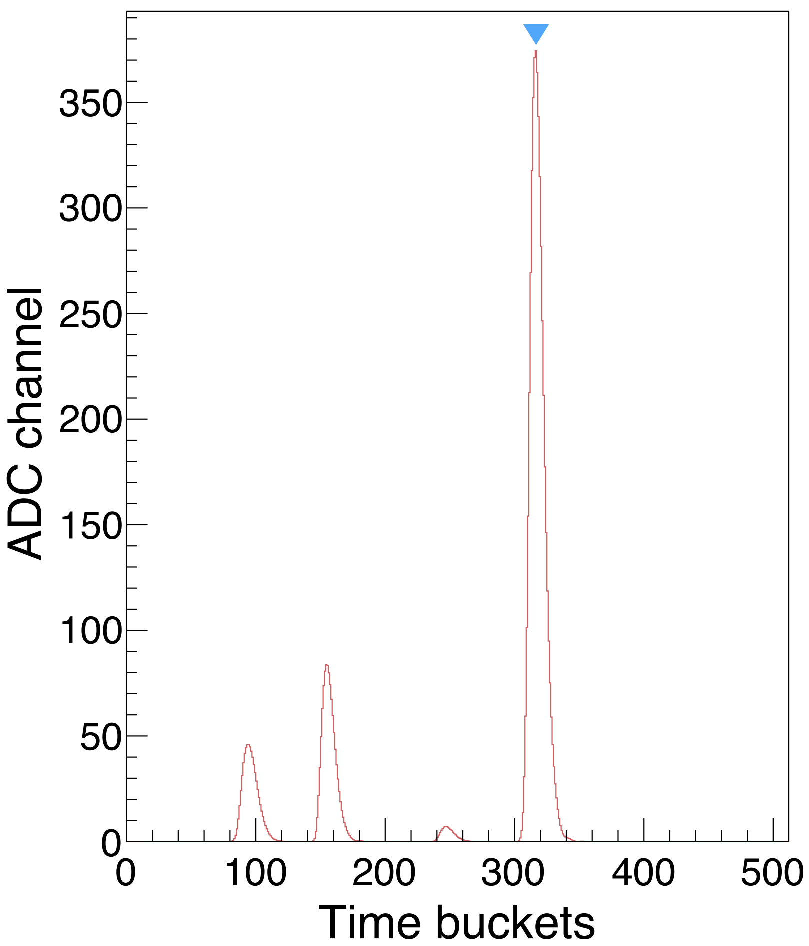
- Represented with 3 digit number.
  - 1st digit: Layer direction number
  - 2nd and 3rd digits: Row direction number



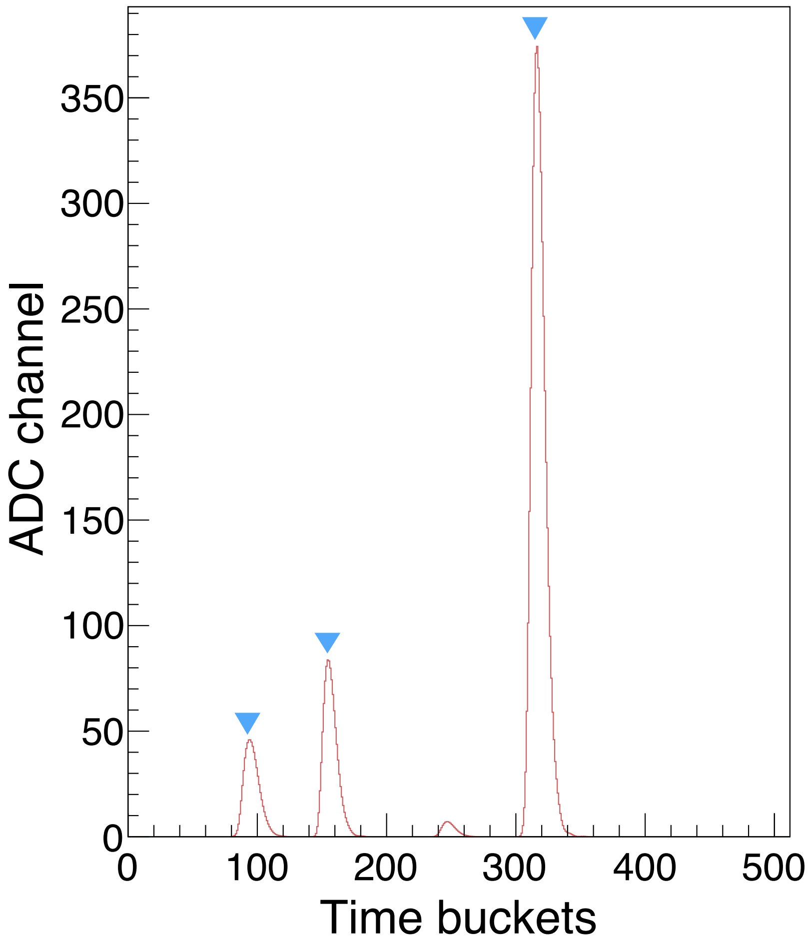
Reconstruction

- Pulse shape analysis
  - Average pulse shape is obtained from HIMAC pulser data.
  - For all cases, charge is recorded by the pulse height.

PSAMode 0



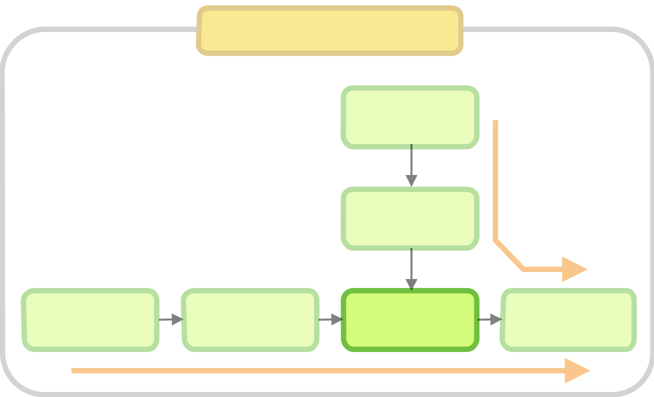
PSAMode 1



- Two simple methods for test
  - PSAMode 0 finds the highest pulse only in each pad
  - PSAMode 1 searches all the peaks in each pad.

- Code

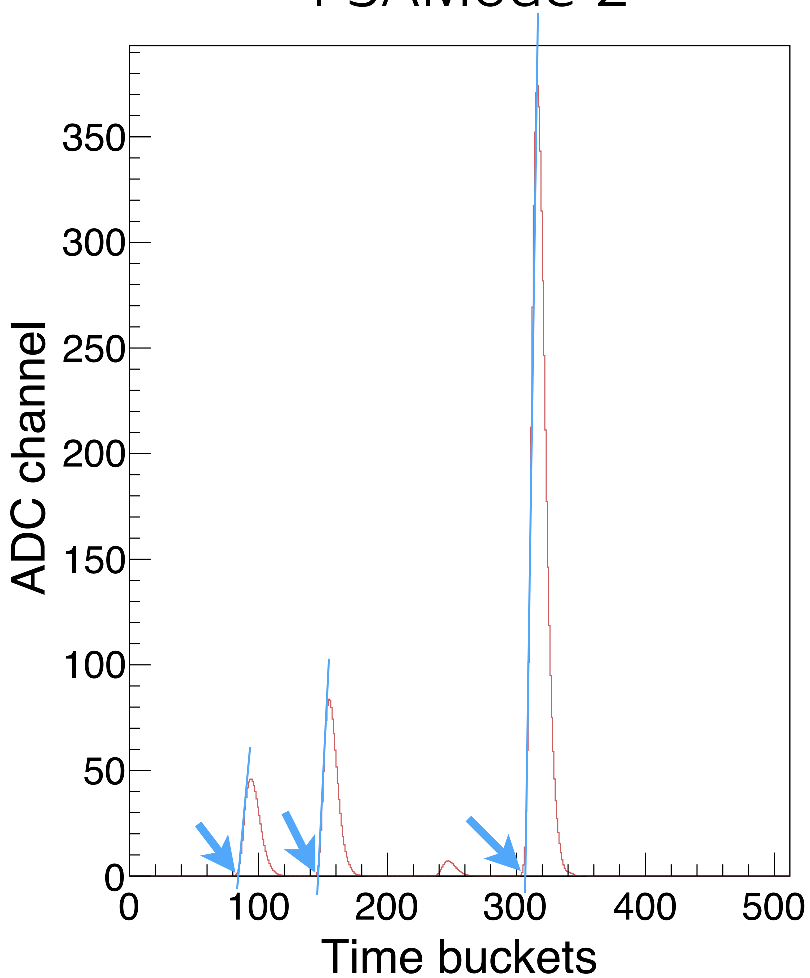
```
STPSATask *psaTask = new STPSATask();
psaTask -> SetPersistence();
psaTask -> SetPSAMode(0);
psaTask -> SetThreshold(40);
run -> AddTask(psaTask);
```



Reconstruction

- Pulse shape analysis
  - Average pulse shape is obtained from HIMAC pulser data.
  - For all cases, charge is recorded by the pulse height.

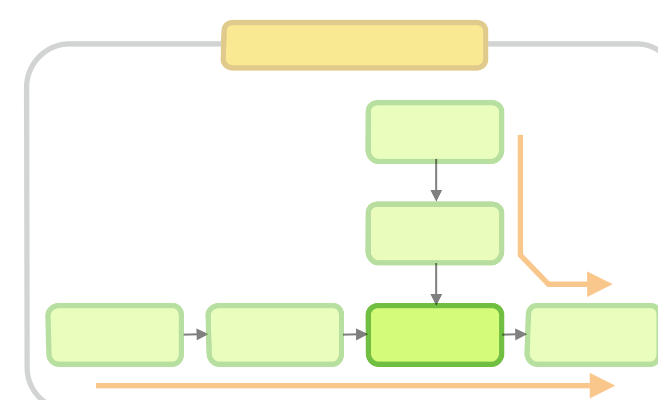
PSAMode 2



- PSAMode2
  - searches every peak in every pad
  - collects (tb, ADC) points between 5 % to 95 % of each peak
  - fits those points with least-square-fit to obtain hit time for each peak
  - from the peak pad, it calculates center of charge with neighboring pads (This will be separated to Clustering Task later.)

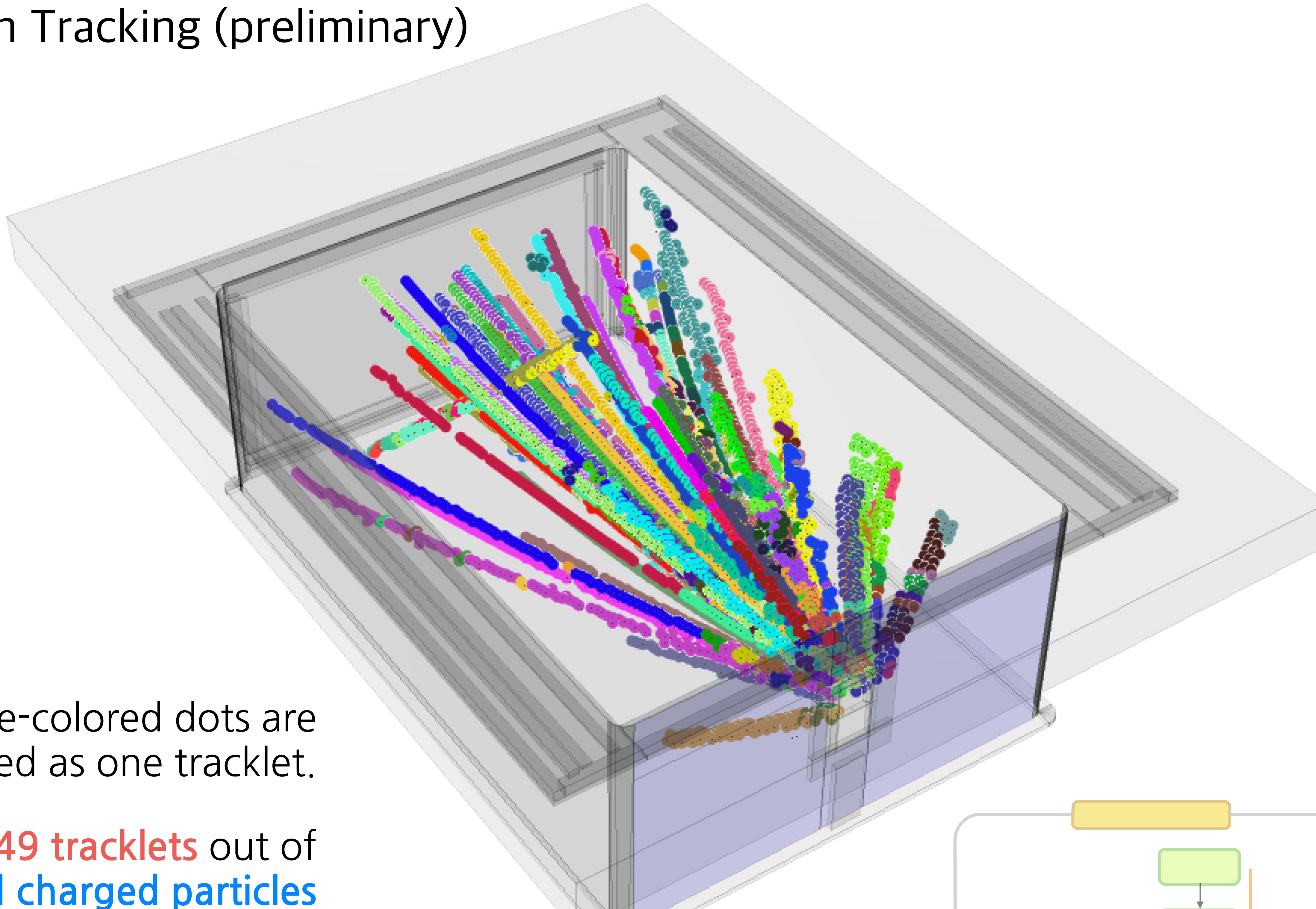
- Code

```
STPSATask *psaTask = new STPSATask();
psaTask -> SetPersistence();
psaTask -> SetPSAMode(0);
psaTask -> SetThreshold(40);
run -> AddTask(psaTask);
```



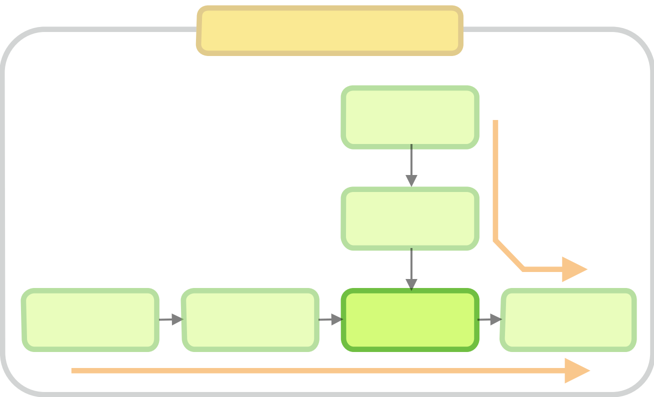
Reconstruction

- Riemann Tracking (preliminary)



Same-colored dots are recognized as one tracklet.

Found 249 tracklets out of 80 produced charged particles



Summary

- FairROOT is well structured framework for an experiment.
- SπRITROOT is written on top of FairROOT.
- All the stages (MC generation, digitization and reconstruction) are working and need to be tuned.

Download link of this slide: http://goo.gl/VQNh1G

Appendix I
Packages in FairSoft

FairSoft

- gtest
  - https://code.google.com/p/googletest/
- GSL
  - http://www.gnu.org/software/gsl/
- boost
  - http://www.boost.org
- Phythia
  - http://home.thep.lu.se/~torbjorn/Pythia.html
- HepMC
  - http://lcgapp.cern.ch/project/simu/HepMC/
- ZeroMQ
  - http://zguide.zeromq.org
- XRootD
  - http://xrootd.org
- Protocol Buffers
  - http://developers.google.com/protocol-buffers
- VGM
  - http://ivana.home.cern.ch/ivana/VGM.html
- Pluto
  - http://www-hades.gsi.de/?q=pluto
- ROOT
  - https://root.cern.ch/drupal/content/download
- VMC, GEANT3
  - https://root.cern.ch/drupal/content/vmc
- Xerces
  - http://xerces.apache.org
- GEANT4
  - http://geant4.web.cern.ch
- Millepede
  - http://www.desy.de/~blobel/mptalks.html
- nanomsg
  - http://nanomsg.org

Appendix II
Pedestal Subtraction Method

FPN channel description in datasheet

2.2 Architecture of the Fixed Pattern Noise channel (FPN channel)

A part of the SCA noise will be probably coherent between channels. To perform common mode rejection, 4 extra channels FPN (Fixed Pattern Noise) are included in the chip. The front-end part of these channels (**Fig. 7**) only includes the inverting 2x Gain stage where the inputs are connected to the input reference voltage. The F.P.N channels will be treated by the SCA exactly as the other channels. Off-line, their outputs can be subtracted from the other 64 analog channels. This pseudo-differential operation can reject the common noise due to 2x Gain and SCA i.e. clock feed-through and substrate coupling. It also improves the power supply rejection ratio (PSRR) of the chip. These channels are distributed uniformly in the chip as shown on **Fig. 30**. Their readout indexes are: 12, 23, 46 & 57.

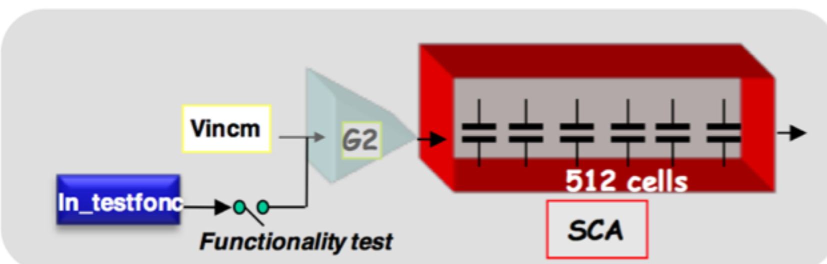


Fig. 7: Schematic of a FPN channel.

The FPN channels can be tested only by using the functionality test. In this case, the input voltage step from the In_testfonc input (pad n° 40) is applied directly to the input of the inverting 2x GAIN stage of the selected FPN channel.

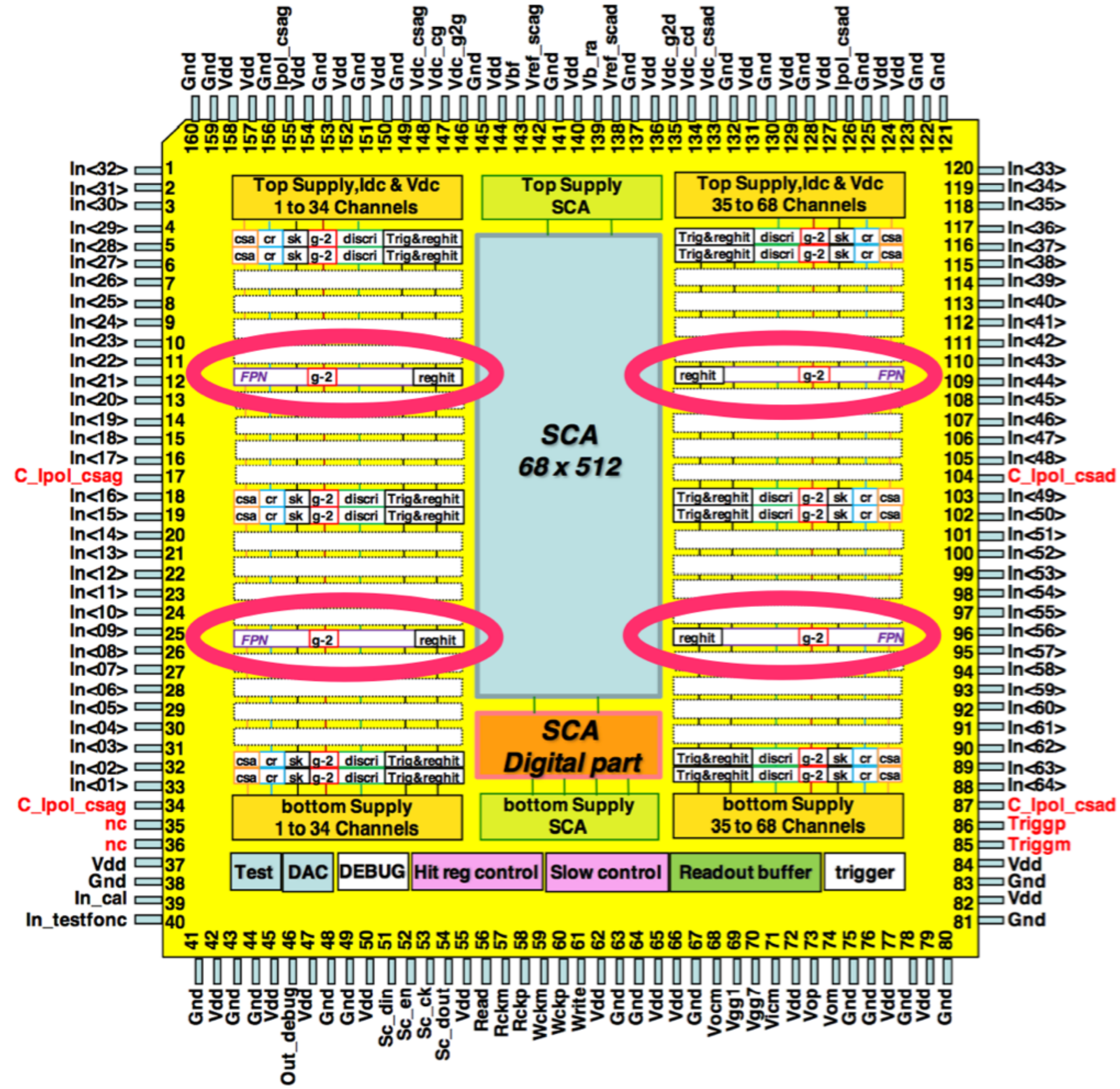
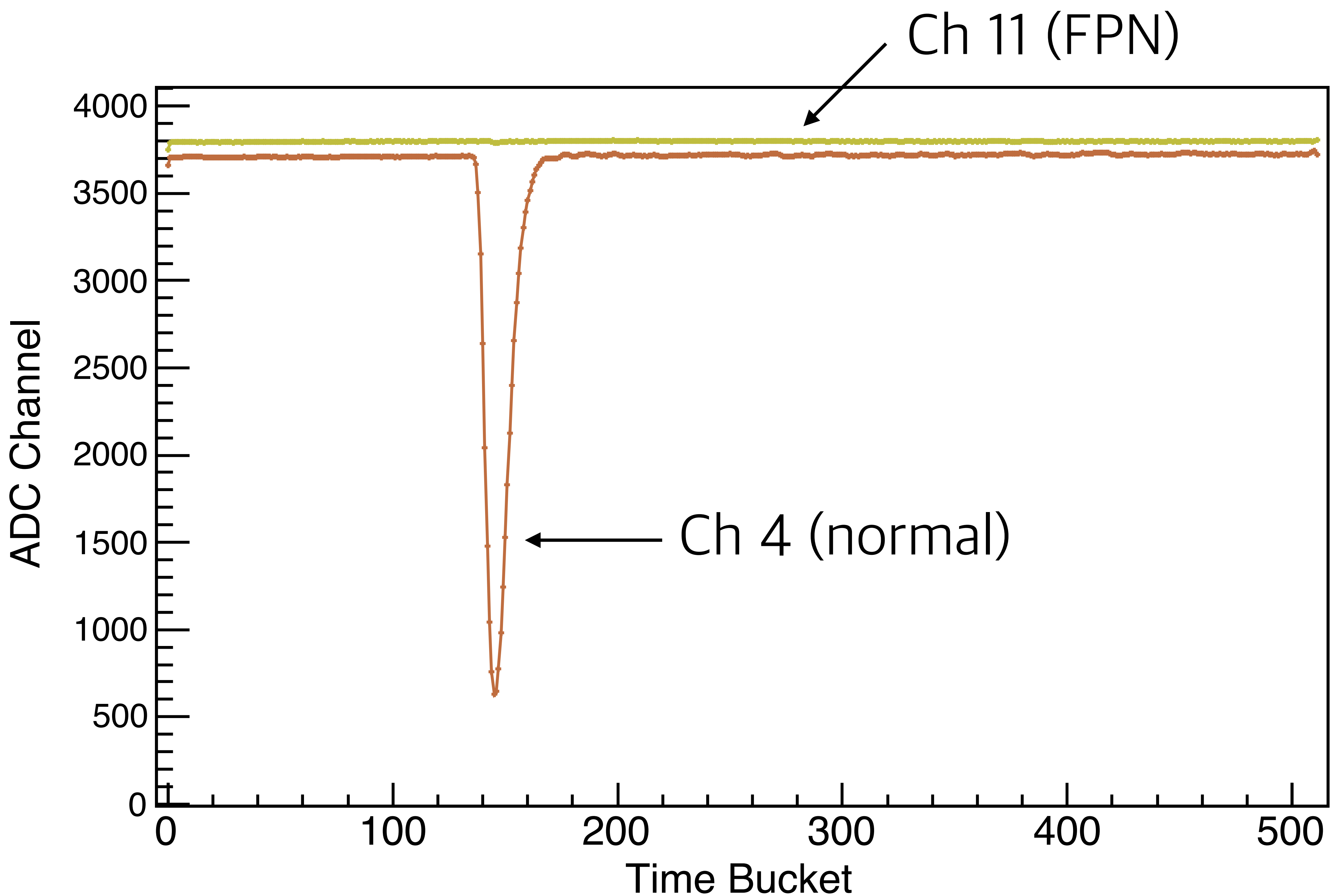
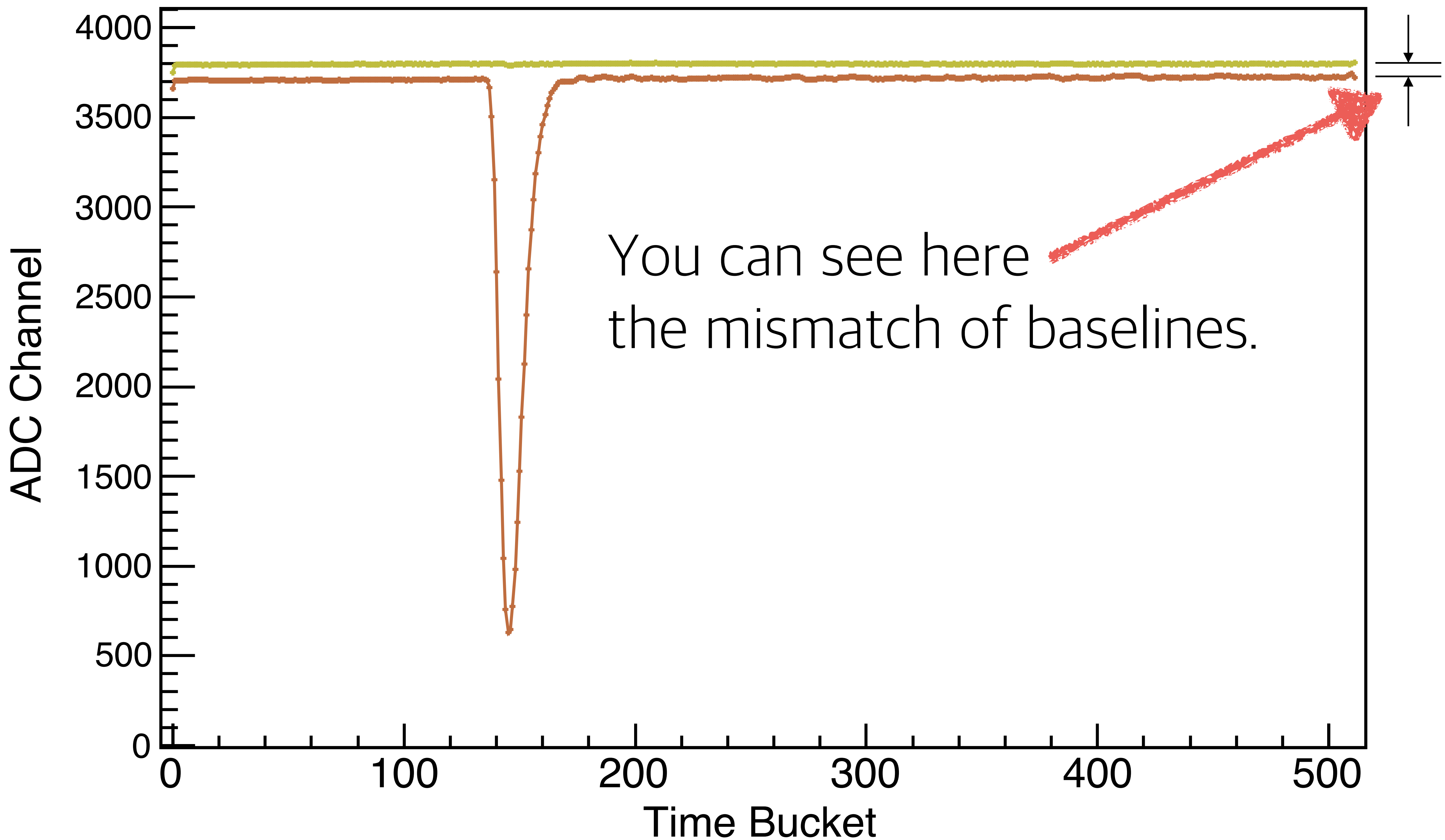


Fig. 30: Internal architecture of the AGET chip.

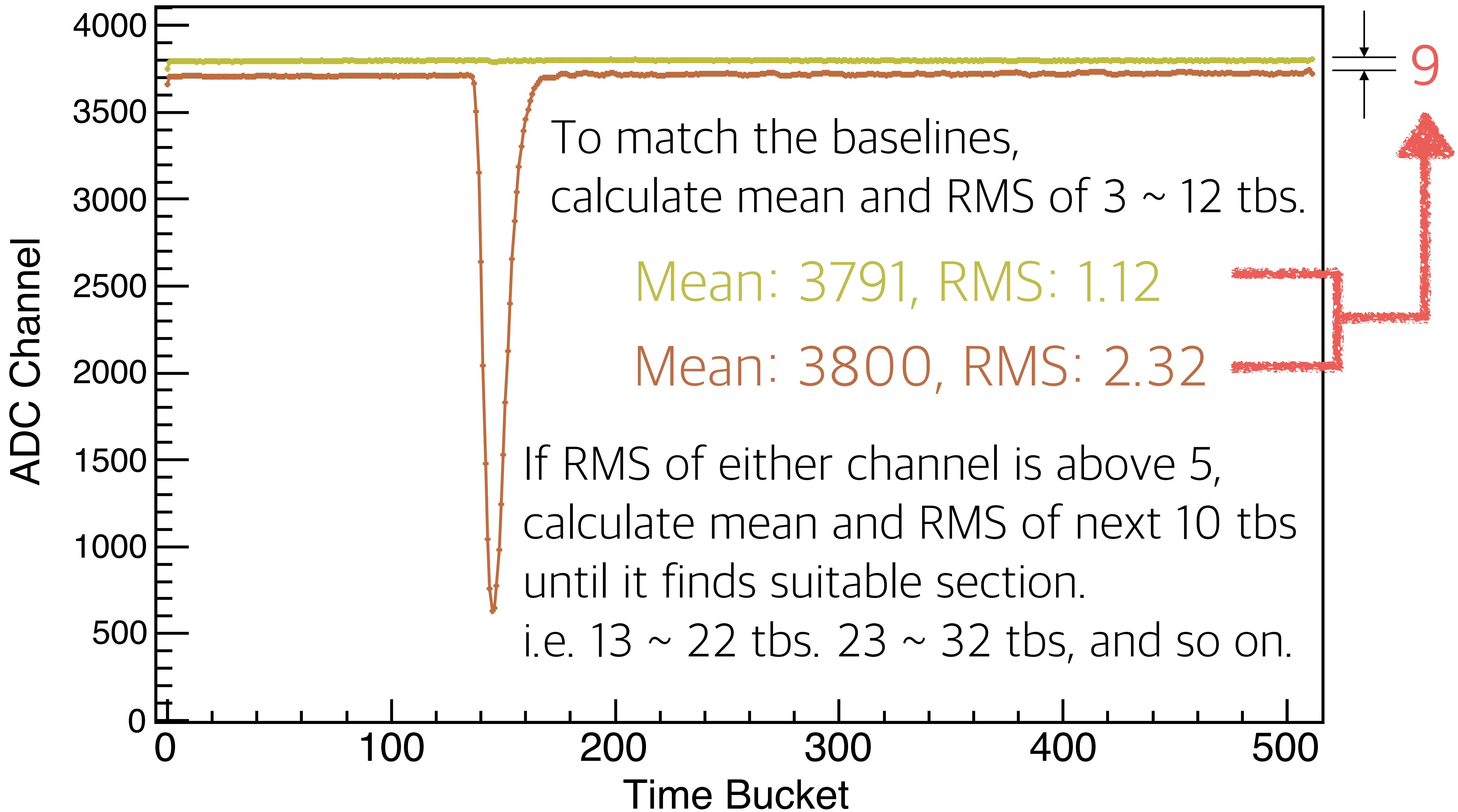
Pedestal subtraction method - 1



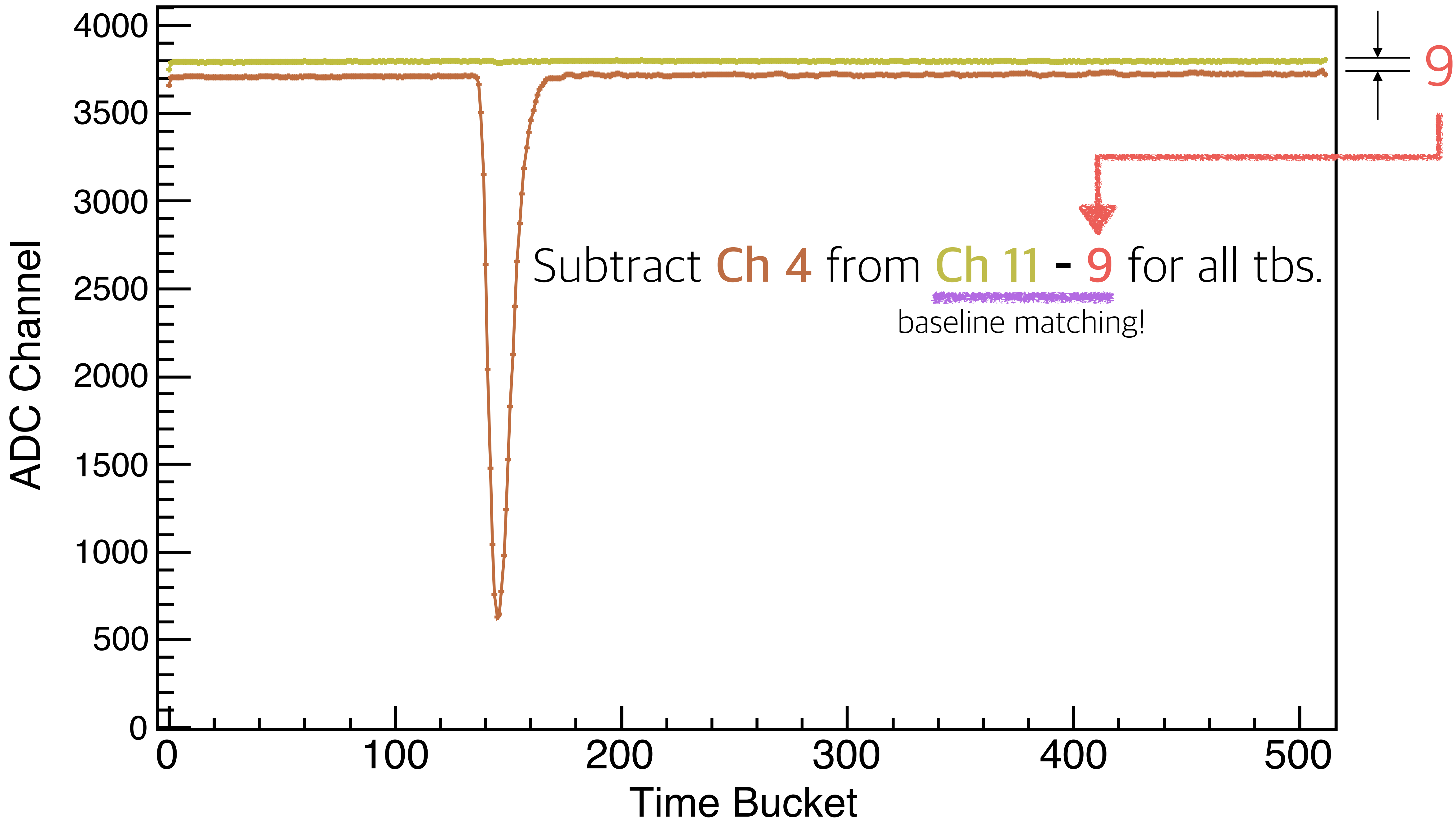
Pedestal subtraction method - 2



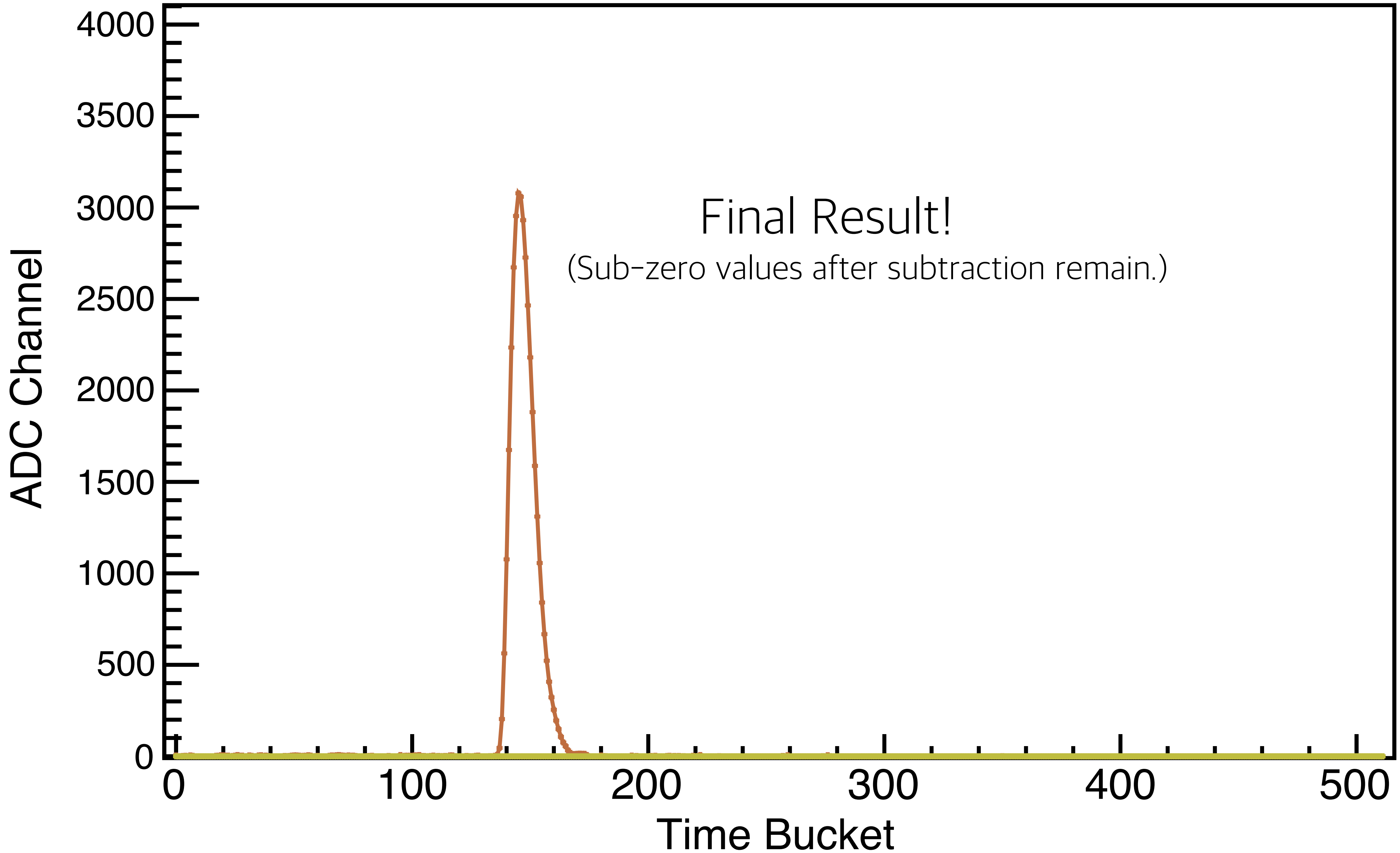
Pedestal subtraction method - 3



Pedestal subtraction method - 4



Pedestal subtraction method - 5



Pedestal subtraction method - Note

- GETDecoder automatically finds nearest FPN channel from the given channel number.
  - Ch 0 ~ 16 = FPN Ch 11
  - Ch 17 ~ 33 = FPN Ch 22
  - Ch 34 ~ 50 = FPN Ch 45
  - Ch 51 ~ 67 = FPN Ch 56

- If there's no section whose RMS is below 5,
  it returns error. (Threshold value is changeable.)