Backward endcap EMC digitization in PandaROOT

Guang Zhao (*zhaog@ihep.ac.cn*)[1], Oliver Noll[2], Shengsen Sun[1], Luigi Capozza[2]

[1] Institute of High Energy Physics (Beijing)

[2] Helmholtz-Institute Mainz

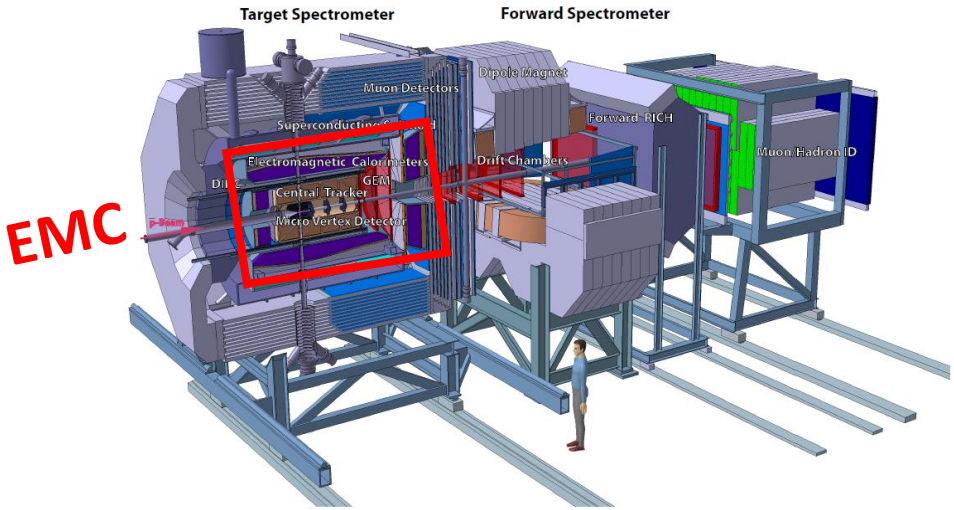
Panda Collaboration Meeting 19/2

25/06/2019 @GSI

Outline

- **Introduction**
  - Overview of the digitization software
  - Development strategy
- **Implementation of the backward endcap EMC digitization**
  - Signal generator
  - Feature extraction
- **Summary and outlook**

Introduction



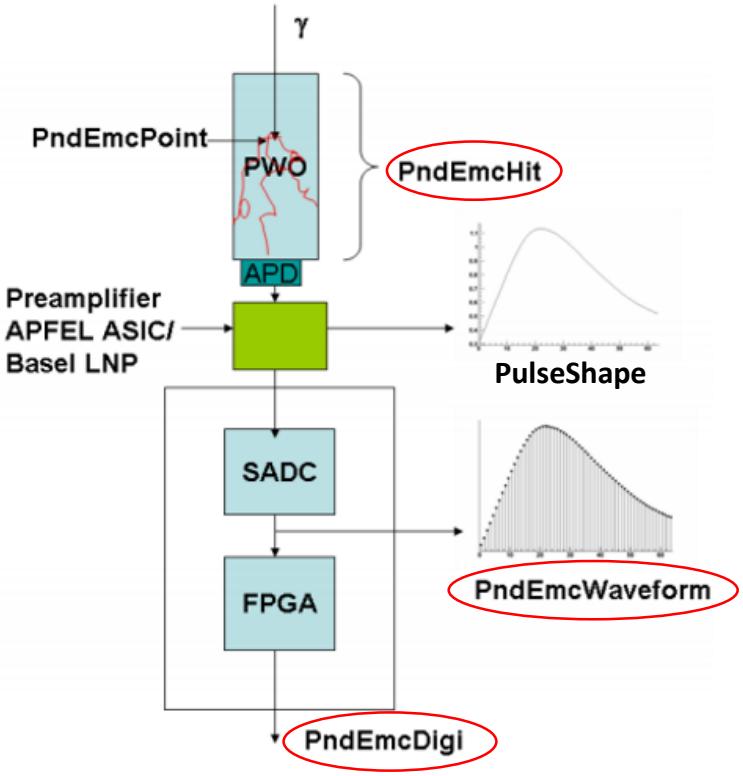
A slice of the barrel EMC

- **The EMC detector:**
  - Main target detector
  - Single crystal threshold from 3 MeV to >10 GeV
  - Working temperature: -25 C (*4 photon yields to room temperature)
- **Simulation:**
  - In order to detect photons in such a wide energy range, need detailed simulation such as geometry description and digitization, etc
- **Main work:**
  - Geometry description for barrel EMC (published in dec18)
  - Digitization implementation for the backward endcap EMC (this talk)

The backward endcap digitization

- **The latest signal processing work for the backward endcap EMC by Oliver Noll (refer his talk tomorrow)**
  - Signal generator: software for simulating 2-gain APFEL pulses, including noise, which can very well reproduce the real data
  - Feature extraction: pulse recognition and analysis, implemented as firmware, also exist as software
- **Digitization in simulation need to be updated according to these new hardware design**
- **The software implementation should be efficient and fully compatible with the PandaROOT framework**

Digitization process in PandaRoot



Signal Generator (SG)

- **Create analog waveform**
- **Add noises and digitize the waveform**

Feature Extraction (FE)

- **Hit detection**
- **Amplitude/time extraction**
- **Pile-up recovery**

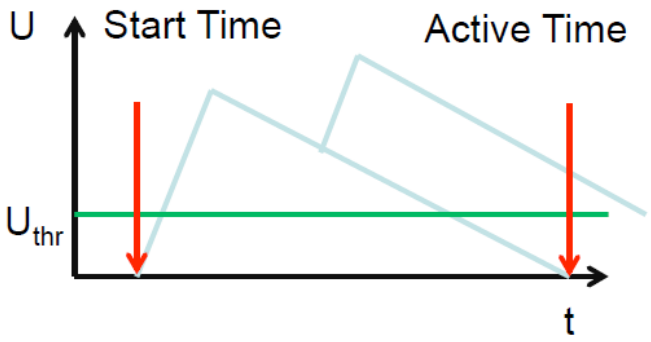
2 major components in digitization

Besides, the software should also support the time-based simulation ⇨

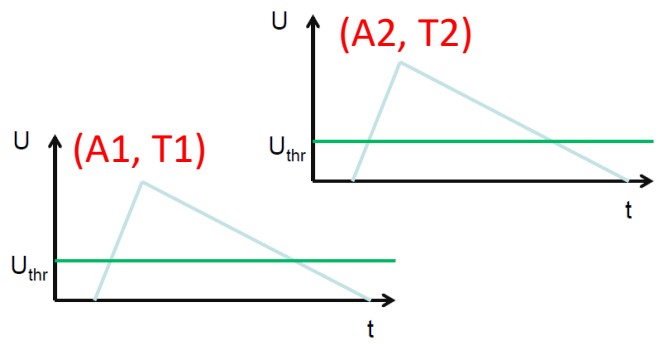
The time-based simulation

- **The digitization should support the time-based simulation**
  - Because
    - Panda readout is trigger-less
    - For barrel/backward endcap, a single crystal rate up to 100 kHz lead to 1% pile-up probability
  - Need to handle
    - Add up multiple hits in SADC as part of signal generator
    - Separate pile-up waveforms as part of feature extraction

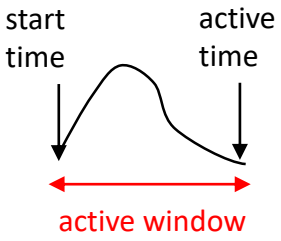
Add up hits



Extract amplitudes/times



The time-based simulation (II)



Wfs in Event 1

Wfs in Event 2

...

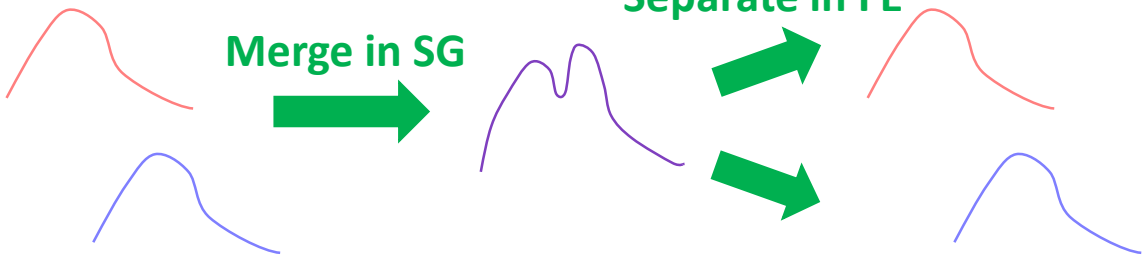
Sort by start time and fill into the buffer

Buffer (shared between events):

Waveforms pile up in the same detector element

Merge in SG

Separate in FE



Writeout data objects whose active time < event time in the buffer to TClonesArray for each event

Time-based TClonesArray:

The strategy

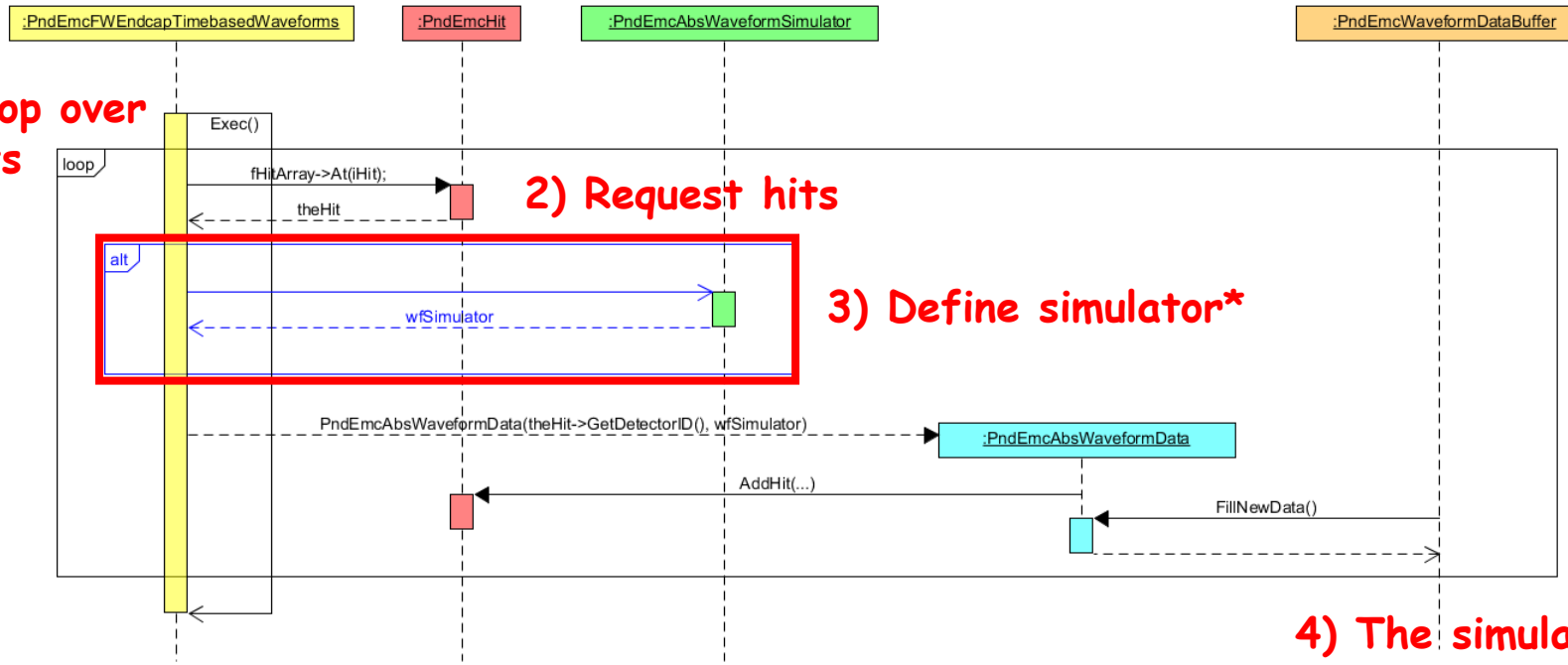
- **Major requirements for the digitization package**
  - Main function: signal generator + feature extraction
  - Support time-based simulation
- **Compare the two existing digitization algorithms in PandaROOT**

	The default package	The forward package
Use range	All EMC	Only FW Endcap
Time-based simulation	Support	Support
Multi-gain waveform	No	Yes
Scalability	OK	Easier

- **Decide to implement a standalone backward endcap digitization package based on the forward package.**
- **In the end, consider to integrate with other EMC modules**

Signal generator

The hits to waveforms process



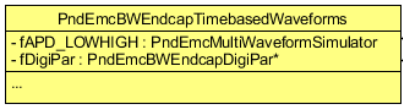
1) Loop over events

2) Request hits

3) Define simulator*

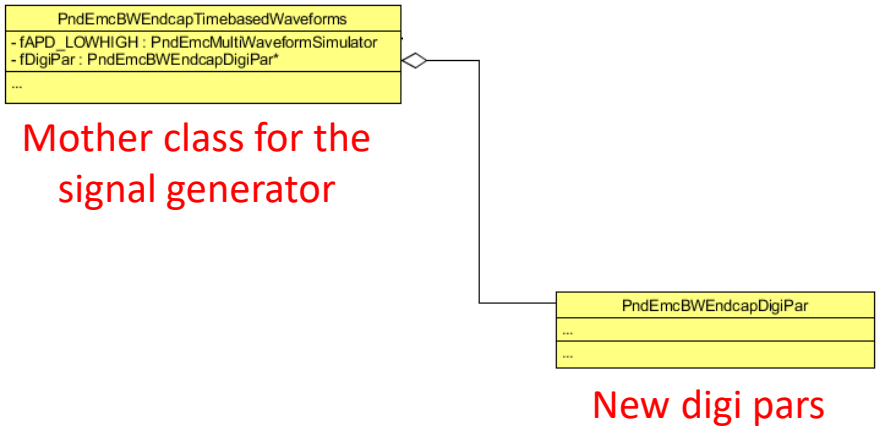
4) The simulator handles waveform generation and writeout to file using interface methods

The main class



Define a new main class for the signal generator

The digitization parameters



Mother class for the signal generator

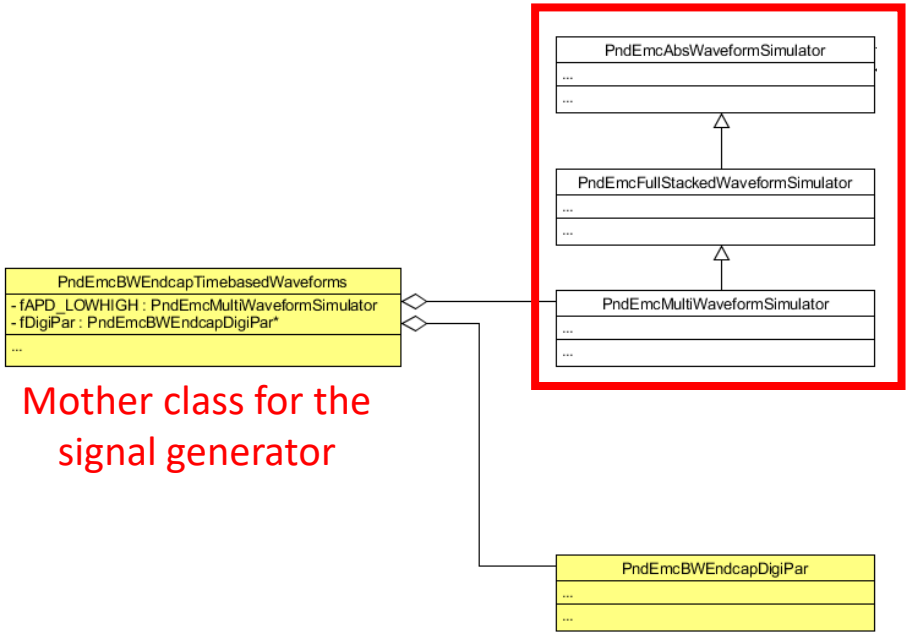
New digi pars

PndEmcBWEndcapDigiPar

+ GetTimeBeforeHit() : Double_t
+ GetTimeAfterHit() : Double_t
+ GetWfCutOffEnergy() : Double_t
+ GetSampleRate() : Double_t
+ GetNBits() : Int_t
+ GetPulseshapeTau() : Double_t
+ GetPulsesapeN() : Double_t
+ GetEnergyRangeHigh() : Double_t
+ GetEnergyRangeLow() : Double_t
+ GetGHLGRatio() : Double_t
+ GetADCNoiseHigh() : Double_t
+ GetADCNoiseLow() : Double_t
+ GetFENoiseHigh() : Double_t
+ GetFENoiseLow() : Double_t
+ GetPssigmaHigh() : Double_t
+ GetPsSigmaLow() : Double_t
+ GetGesSigmaHigh() : Double_t
+ GetGesSigmaLow() : Double_t
+ GetFIRCoeffFile() : TString
+ GetTmaxTaps() : Double_t
...

Define a new class to store all parameters for signal generation and feature extraction

The simulator



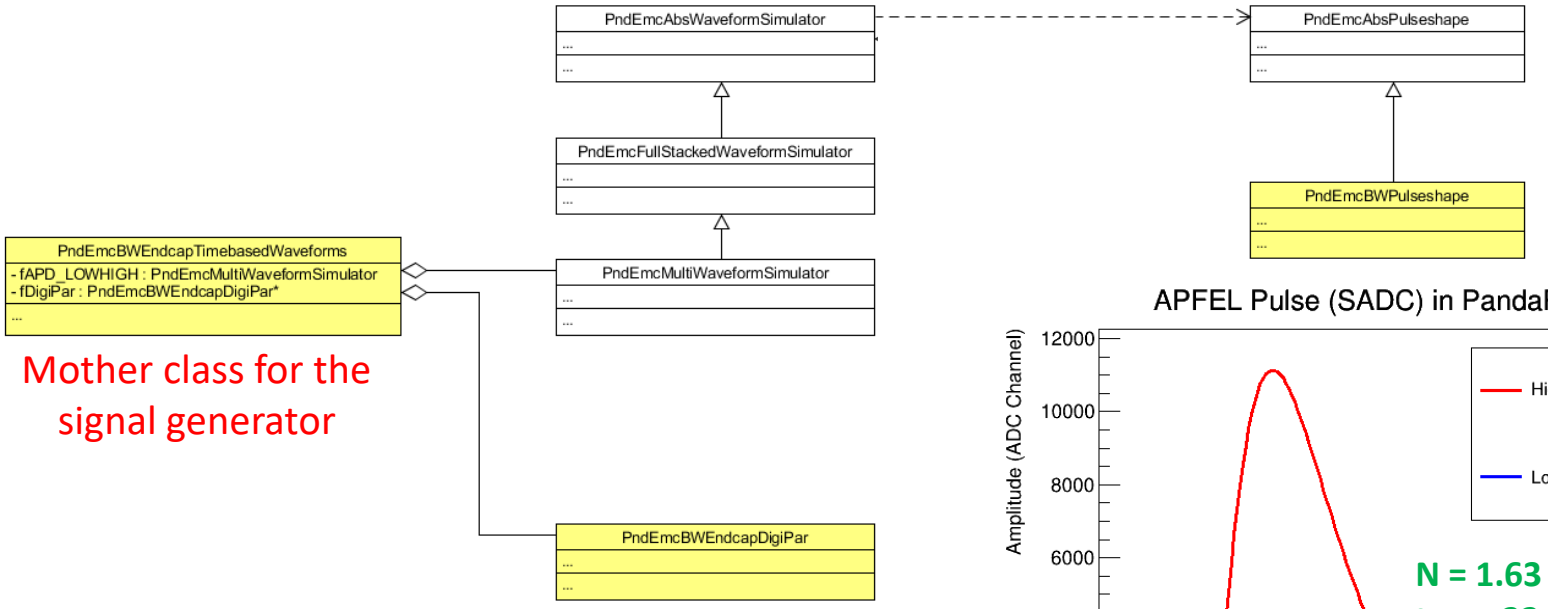
Mother class for the signal generator

New digi pars

- This is the key component
- A simulator is behaved as an interface with
  - **Input:** PndEmcHit
  - **Output:** PndEmcWaveform
  - **Interface method:** Simulate()
    - The interface implementation depend on different EMC modules

The pulse shape

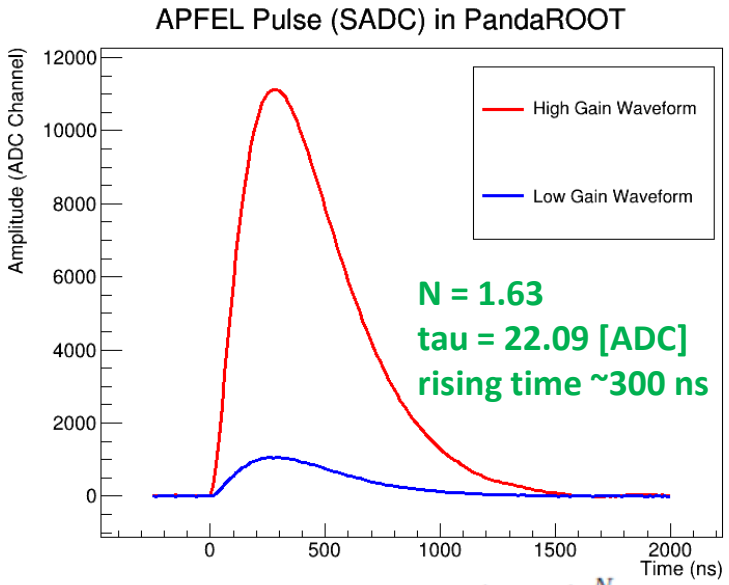
The simulator first produces two gain ideal waveforms



New pulse shape

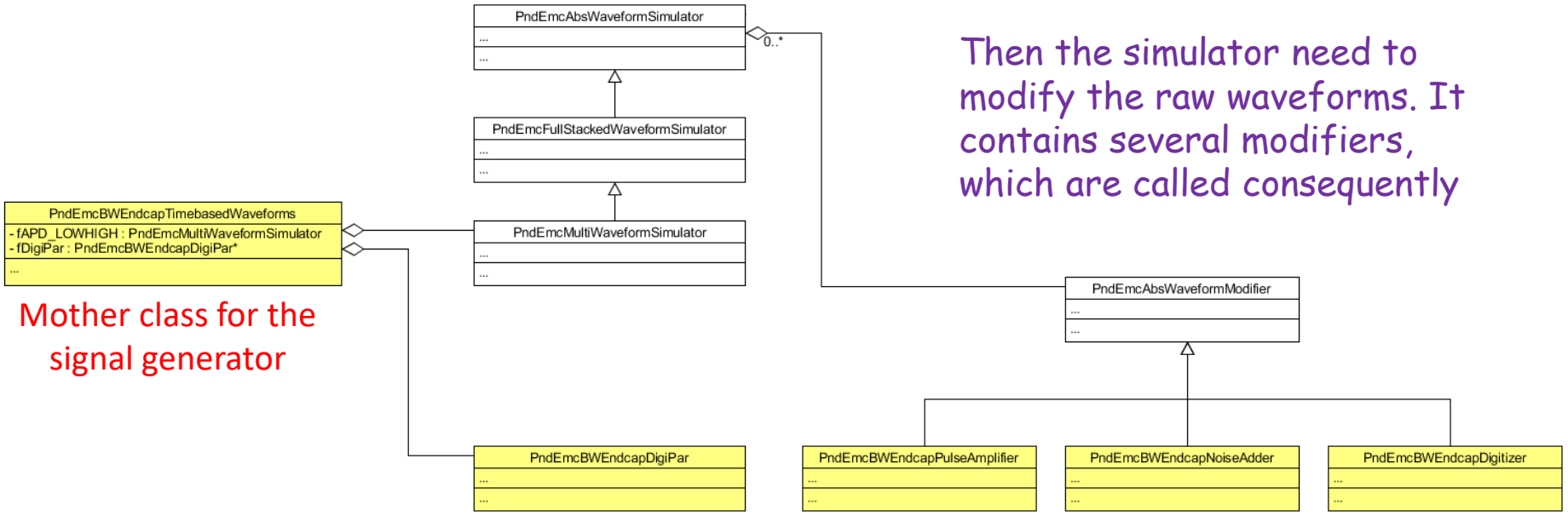
Mother class for the signal generator

New digi pars



$$f(x) = -A \cdot e^{\frac{-N(x-\delta)}{\tau}} \cdot \left(\frac{x-\delta}{\tau}\right)^{N}$$

The modifiers



Then the simulator need to modify the raw waveforms. It contains several modifiers, which are called consequently

Mother class for the signal generator

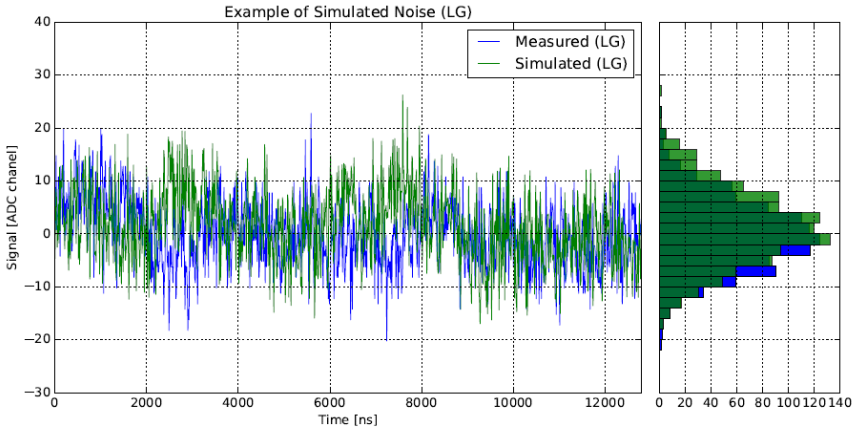
New digi pars

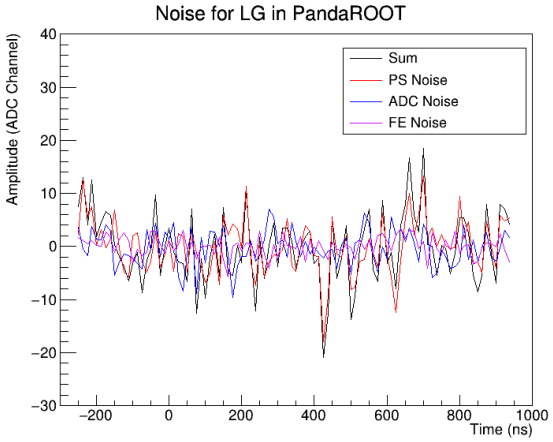
New noise/digitization on waveforms

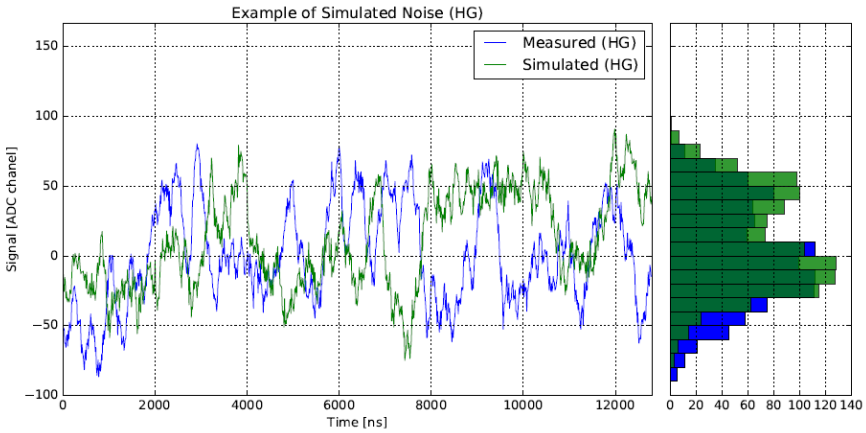
The noises

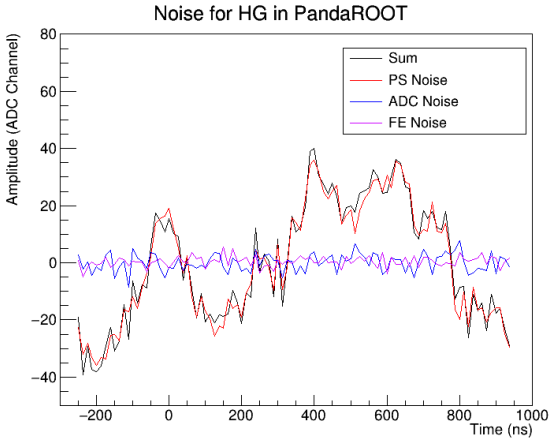
Data and sim by Oliver

Sim in PandaROOT



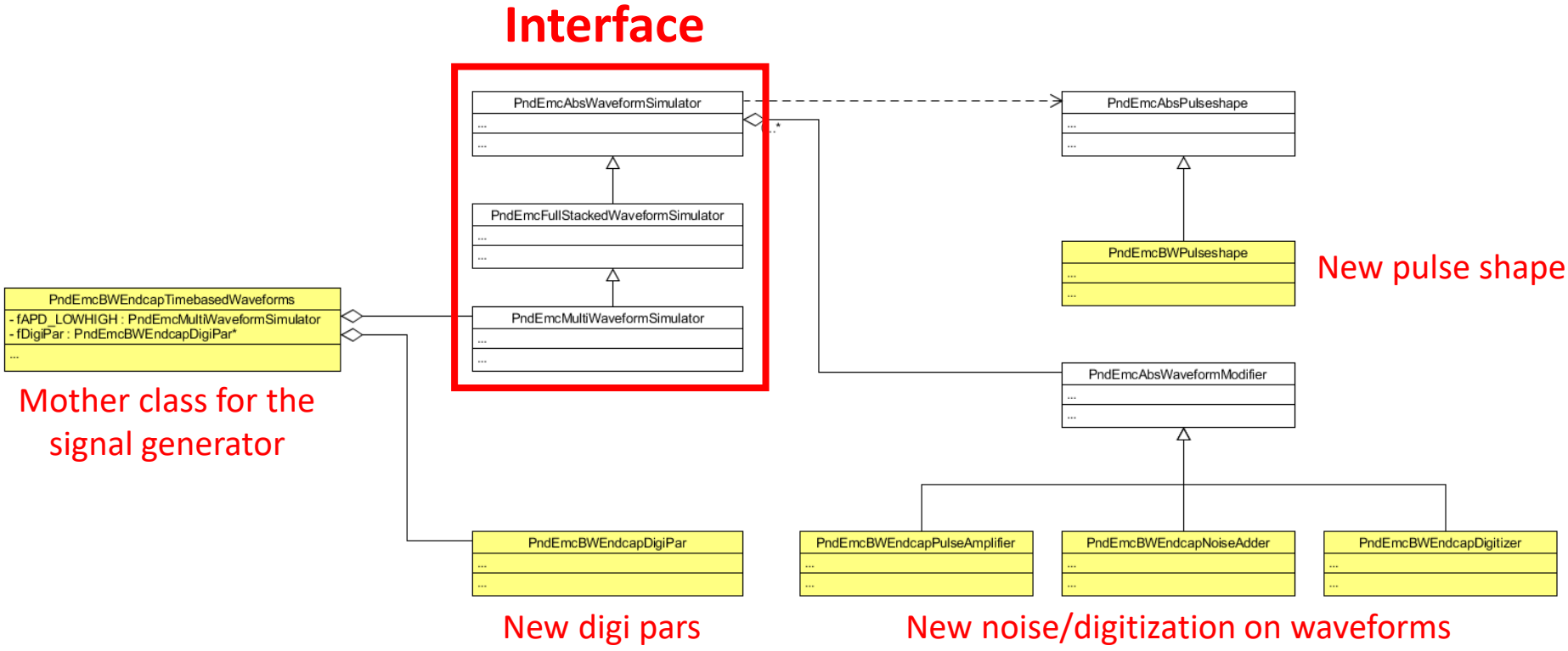






- **Noises are studied with a FFT analysis using beam test data**
- **Simulation can well reproduce the real noise floor**

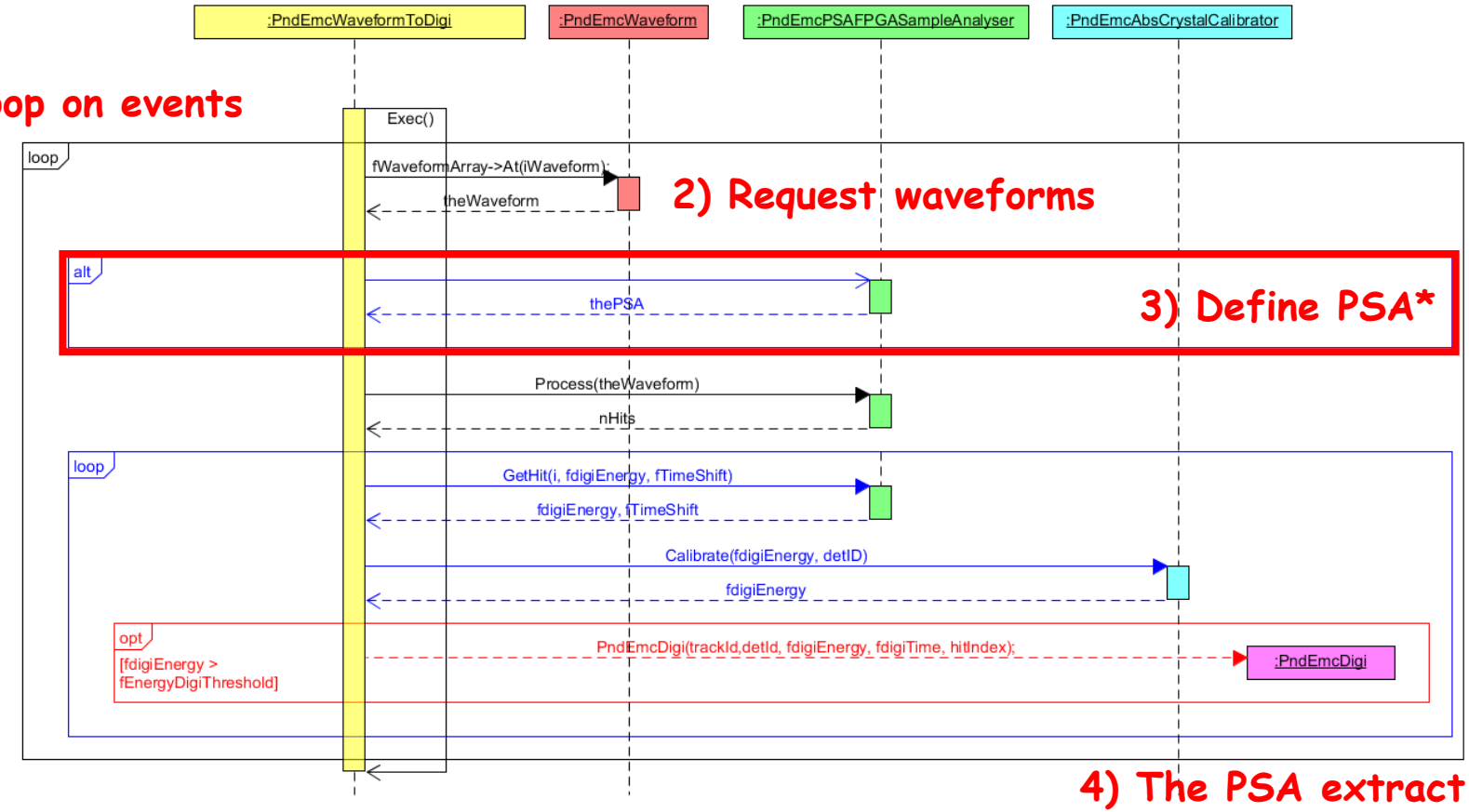
Class diagram for signal generator



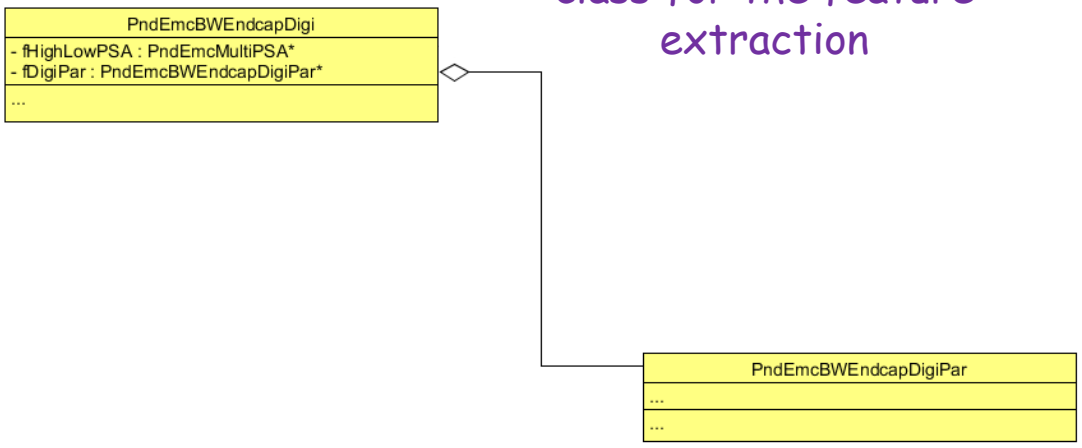
Newly added for BW

Feature extraction

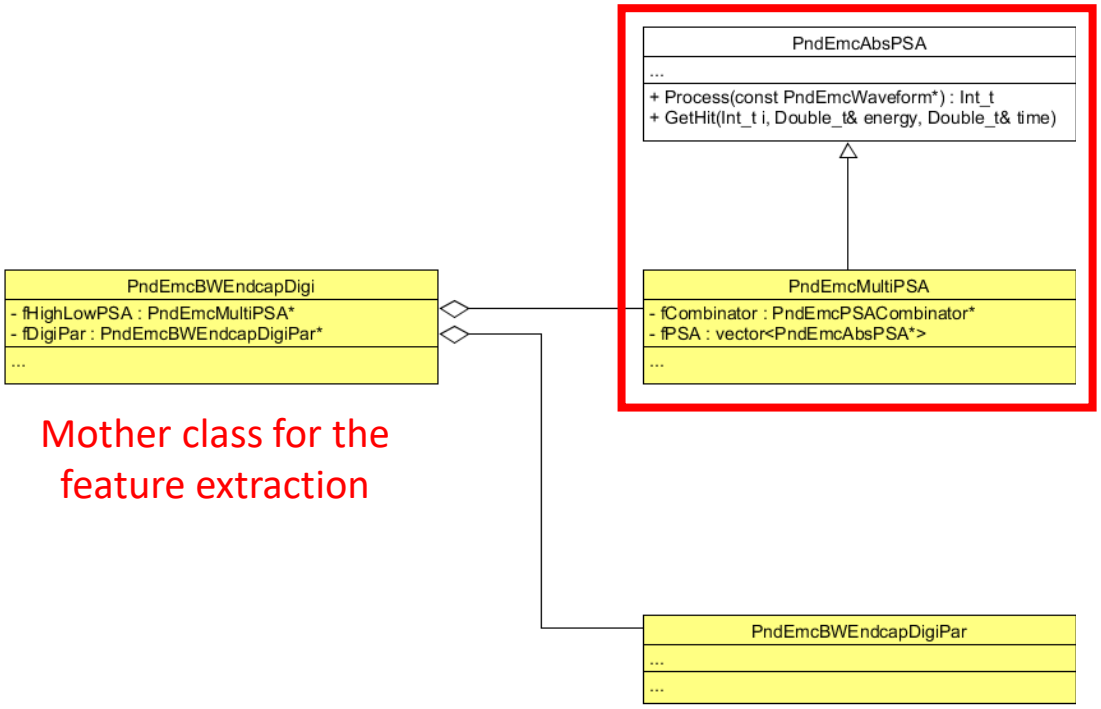
The waveforms to digis process



The main class



The PSA



Mother class for the feature extraction

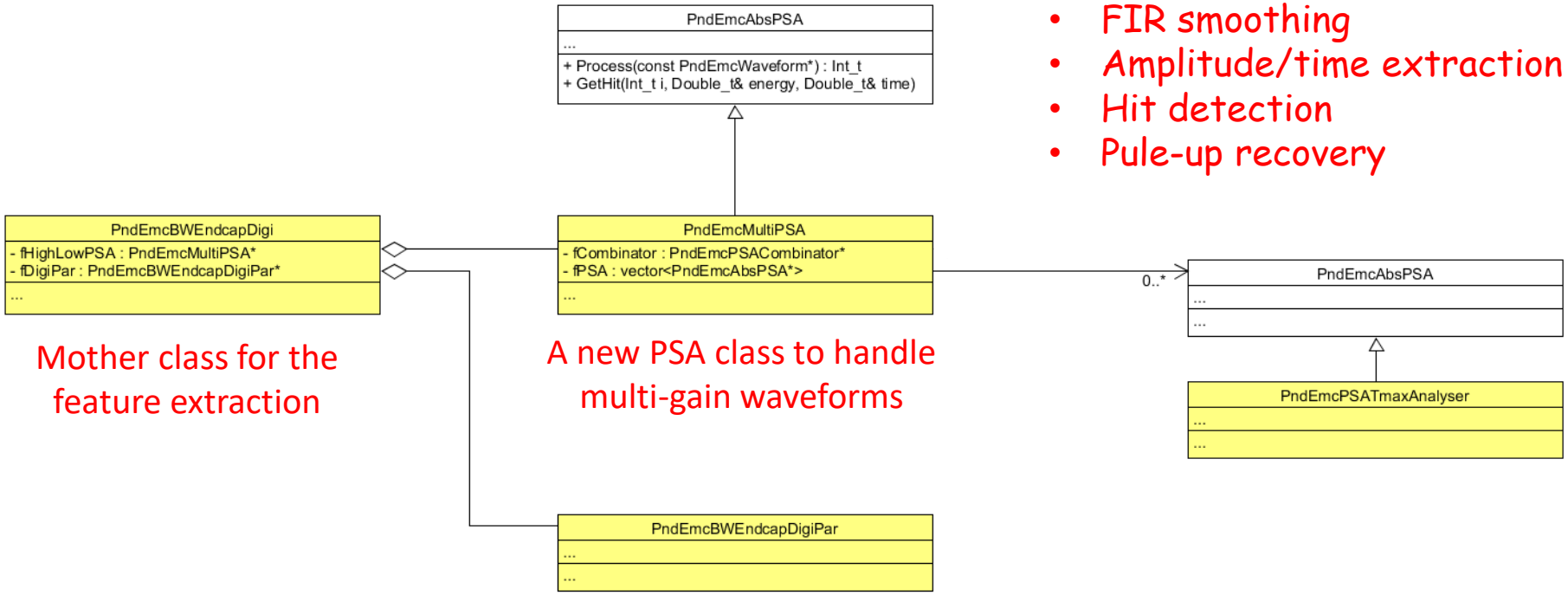
Digi pars (the same to the signal generator)

Newly added for BW

- Key component
- The PSA (pulse shape analyser) is behaved as an interface with:
  - **Input:** PndEmcWaveform
  - **Output:** Energy/time information of the input waveform
  - **Interface method:** Process()
    - The implementation depend on EMC modules
- As there are 2 gains for the backward endcap, define a daughter class PndEmcMultiPSA

The TMAX PSA

- The PndEmcMultiPSA includes 2 PndEmcPSATmaxAnalyser for each gain
- The TMAX PSA contains all the feature extraction algorithms
  - FIR smoothing
  - Amplitude/time extraction
  - Hit detection
  - Pule-up recovery



Mother class for the feature extraction

A new PSA class to handle multi-gain waveforms

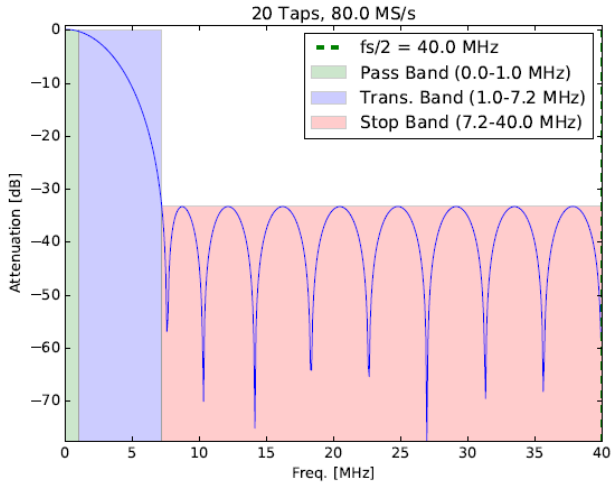
Digi pars (the same to the signal generator)

Newly added for BW

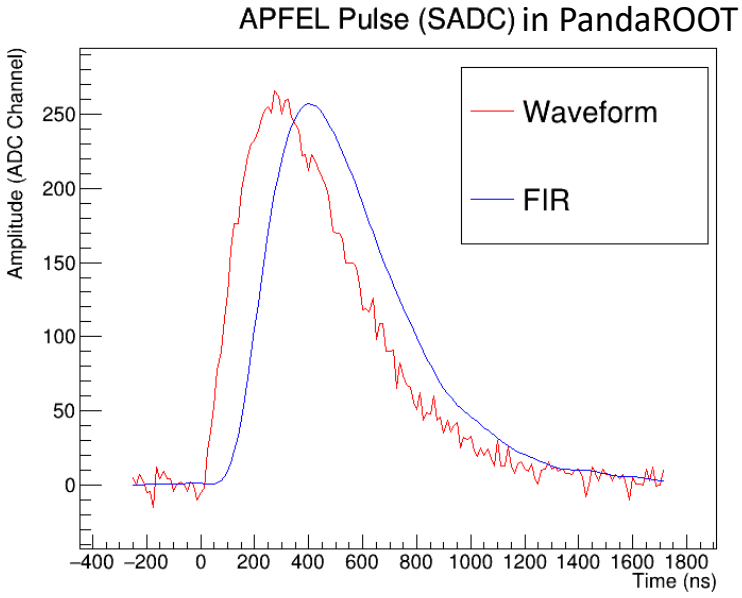
FIR filtering

PndEmcPSATmaxAnalyser
- fEnergyList : vector<Double_t> - fTimeList : vector<Double_t>
+ Process(const PndEmcWaveform*) : Int_t + GetHit(Int_t, Double_t&, Double_t&) - fir(Int_t*, Int_t) : Double_t* - hit_det(Int_t, Int_t, Int_t) : Double_t

- Transfer function suppressed HF noise (low pass)
- Z transformation of impulse response
- $H(z) = \sum_{n=0}^{N} h(n) \cdot z^{-n}$
  - $h(n)$: Filter Koeffizienten
  - $z = e^{i\omega T}$
- Each output value is weighted sum of most recent input values
- $\mathbf{out}[n] = h_0\mathbf{in}[n] + h_1\mathbf{in}[n-1] + ... + h_N\mathbf{in}[n-N]$



Low pass filter to smooth the waveform
(20 coefficients, ~10 cycle clocks latency)



Amplitude/Time path

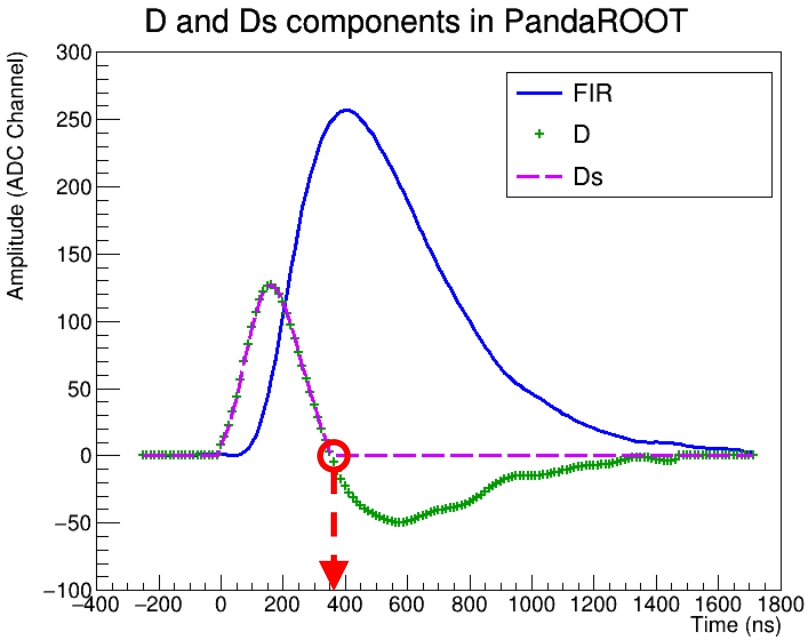
PndEmcPSATmaxAnalyser
- fEnergyList : vector<Double_t> - fTimeList : vector<Double_t>
+ Process(const PndEmcWaveform*) : Int_t + GetHit(Int_t, Double_t&, Double_t&) - fir(Int_t*, Int_t) : Double_t* - hit_det(Int_t, Int_t, Int_t) : Double_t

The TMAX filter:

Deviation: $$D[i] = T[i + r] - T[i]$$

$$D^*_{inv}[i] = -\Theta(-D[i]) \cdot D[i]$$

Falling edge cancelling: $$D_s[i] = D[i] + D^*_{inv}[i]$$

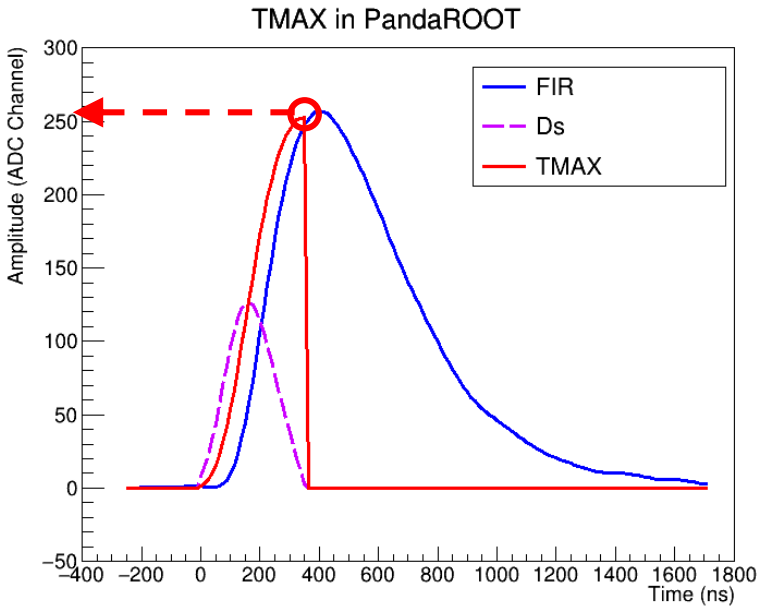


Time is determined at the transition of the derivative at zero

Amplitude/Time path (II)

PndEmcPSATmaxAnalyser
- fEnergyList : vector<Double_t> - fTimeList : vector<Double_t>
+ Process(const PndEmcWaveform*) : Int_t + GetHit(Int_t, Double_t&, Double_t&) - fir(Int_t*, Int_t) : Double_t* - hit_det(Int_t, Int_t, Int_t) : Double_t

Extract amplitude at the TMAX peak



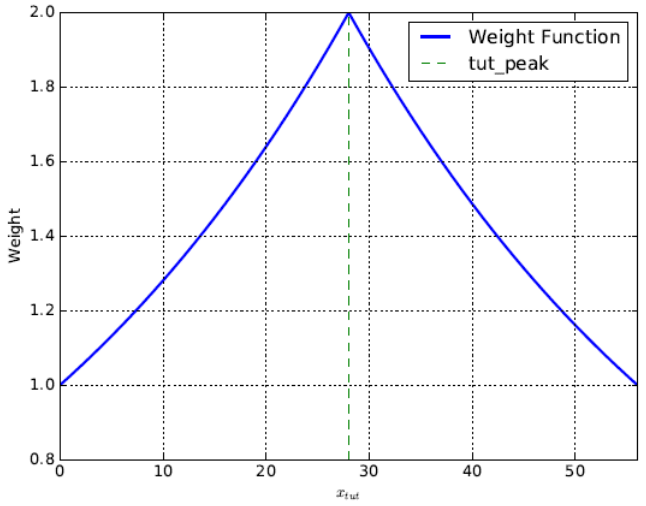
The TMAX filter:

$$D_s[i] \mapsto \begin{cases} F_{TMAX}[i] = F_{TMAX}[i-1] + \frac{D_s[i]}{r} & : D_s[i] < 0 \\ F_{TMAX}[i] = 0 & : D_s[i] = 0 \end{cases}$$

By integrating D_s, we can obtain the amplitude of the rising edge of the pulse

Hit detection

PndEmcPSATmaxAnalyser
- fEnergyList : vector<Double_t> - fTimeList : vector<Double_t>
+ Process(const PndEmcWaveform*) : Int_t + GetHit(Int_t, Double_t&, Double_t&) - fir(Int_t*, Int_t) : Double_t* - hit_det(Int_t, Int_t, Int_t) : Double_t

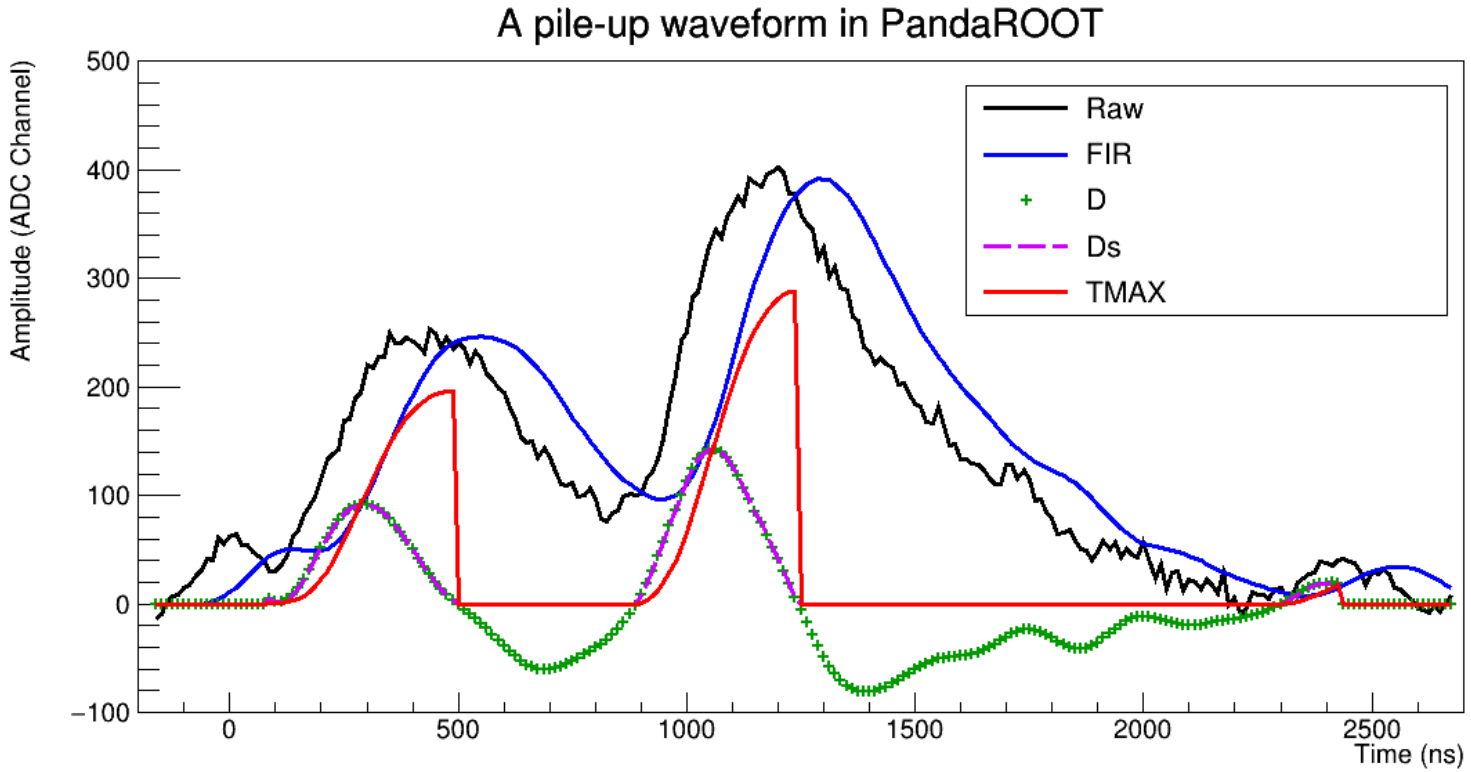


- True hits should be detected from noises
- A function to weight the hit detection with the time under threshold is derived
- The weight function is convoluted with the extraction function (TMAX), and a hit is detected when its value passes a threshold

$$x_{tut} \mapsto \begin{cases} e^{a \cdot x_{tut}} & : x_{tut} < tut_{peak} \\ hit_{val} \cdot e^{-a \cdot (x_{tut} - tut_{peak})} & : x_{tut} \geq tut_{peak} \end{cases}$$

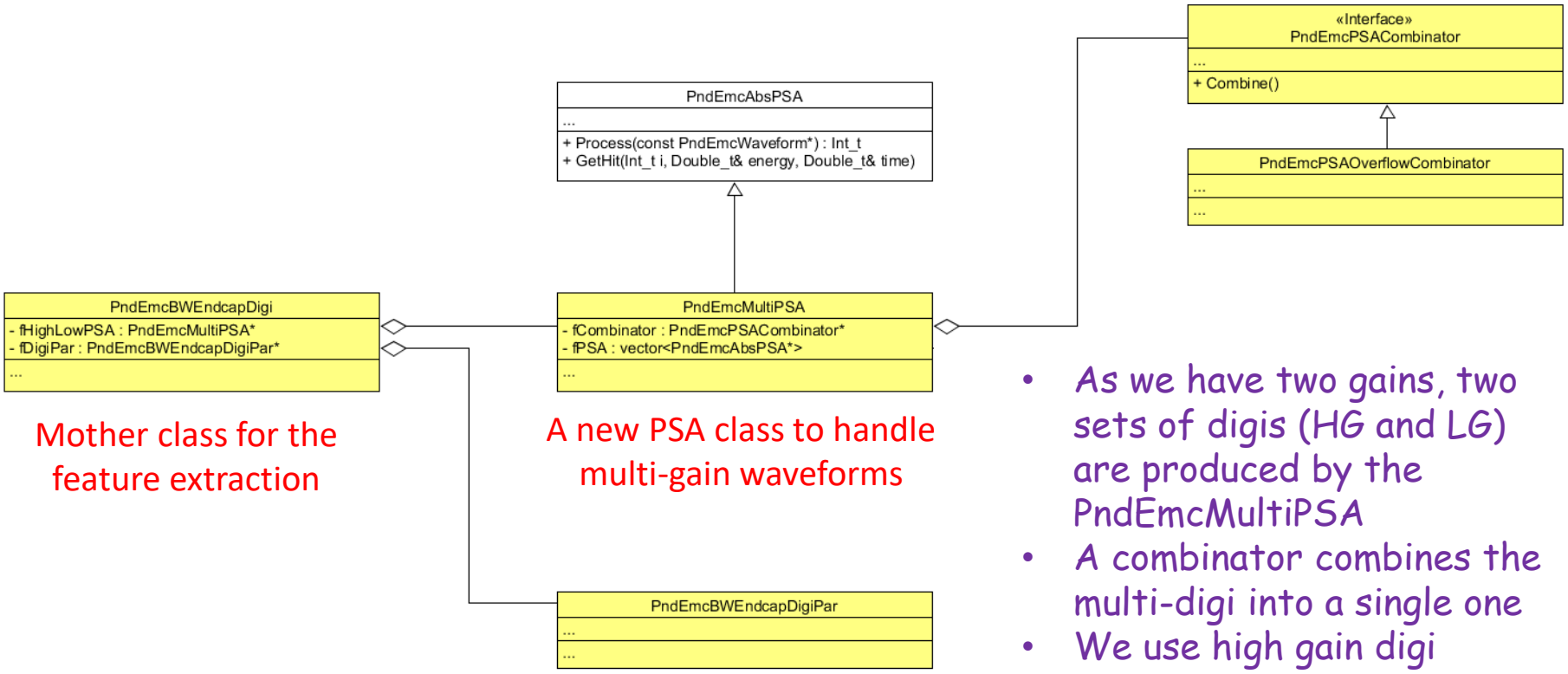
$$(tut_{peak}, hit_{val}) = (28,2)$$

Pule-up recovery in time-based simulation



- ✓ **We are able to produce the pile-up waveforms, and are able to separate them**
- ✓ **For instance, two digis are detected from this pile-up waveform**
- ✓ **The amplitude of the secondary waveforms need to be corrected, because the amplitude of the rising edge does not start from 0**

The combinator



Mother class for the feature extraction

A new PSA class to handle multi-gain waveforms

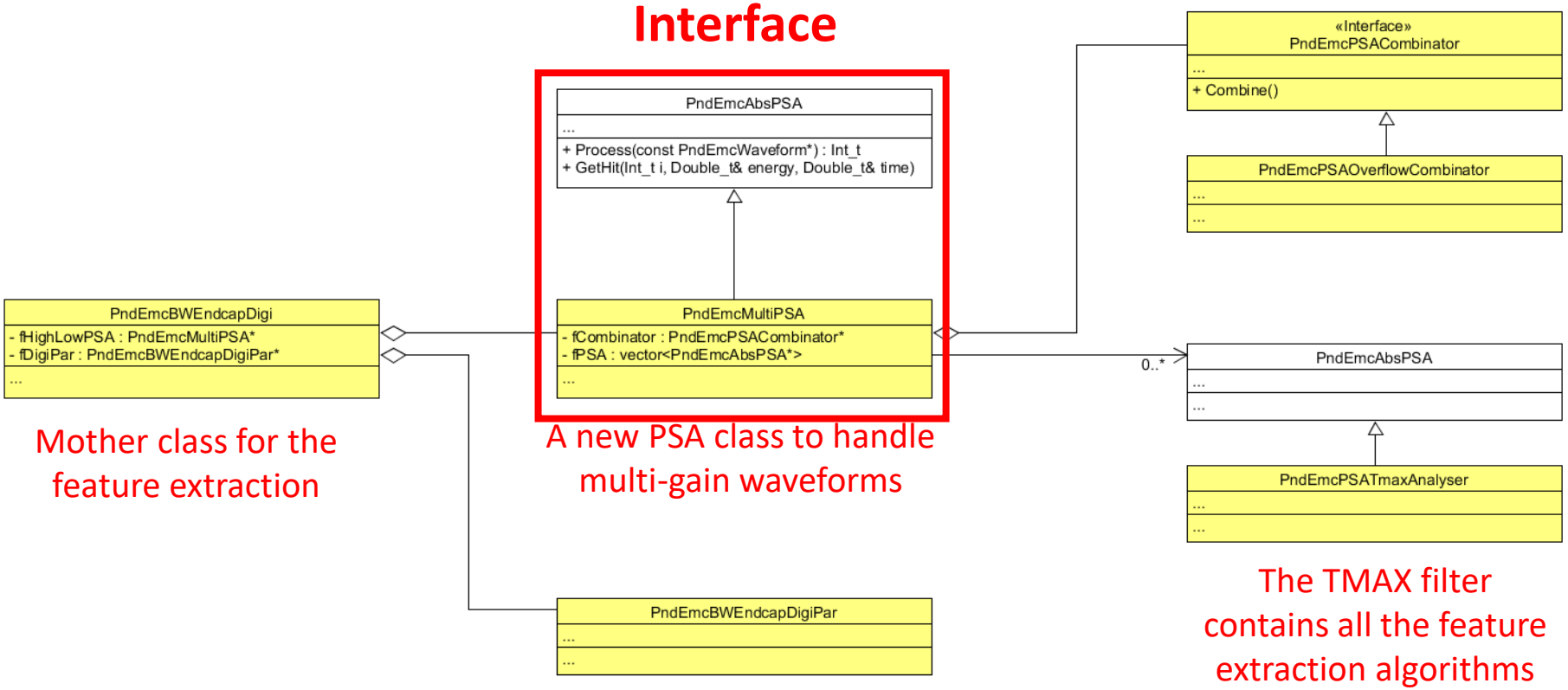
Digi pars (the same to the signal generator)

- As we have two gains, two sets of digis (HG and LG) are produced by the PndEmcMultiPSA
- A combinator combines the multi-digi into a single one
- We use high gain digi unless it is overflowed

Newly added for BW

Class diagram of the BW package

The combinator combines the multi-waveform input to a single output digi. Now we always use the high-gain waveform unless it is overflowed



Mother class for the feature extraction

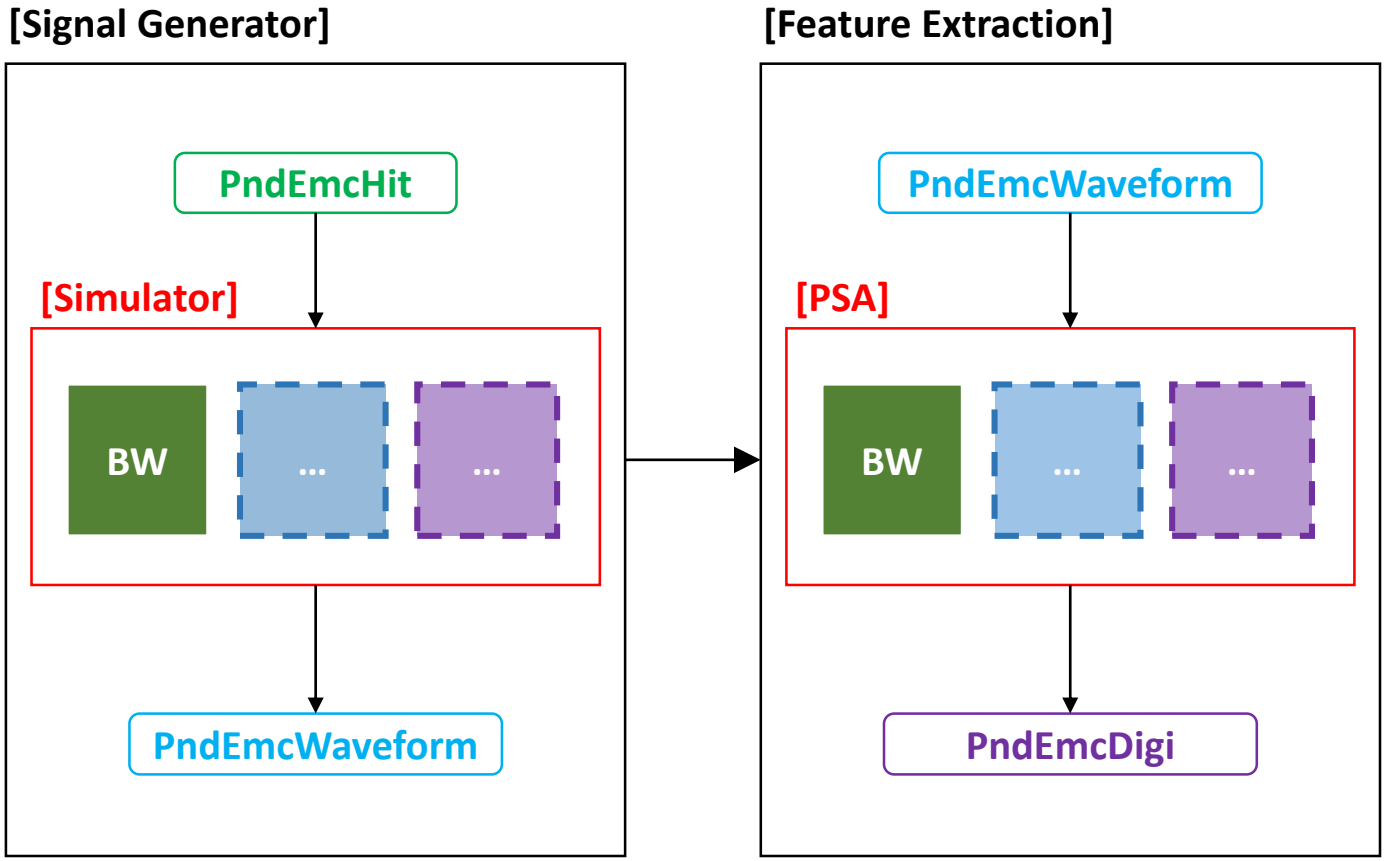
A new PSA class to handle multi-gain waveforms

The TMAX filter contains all the feature extraction algorithms

Digi pars (the same to the signal generator)

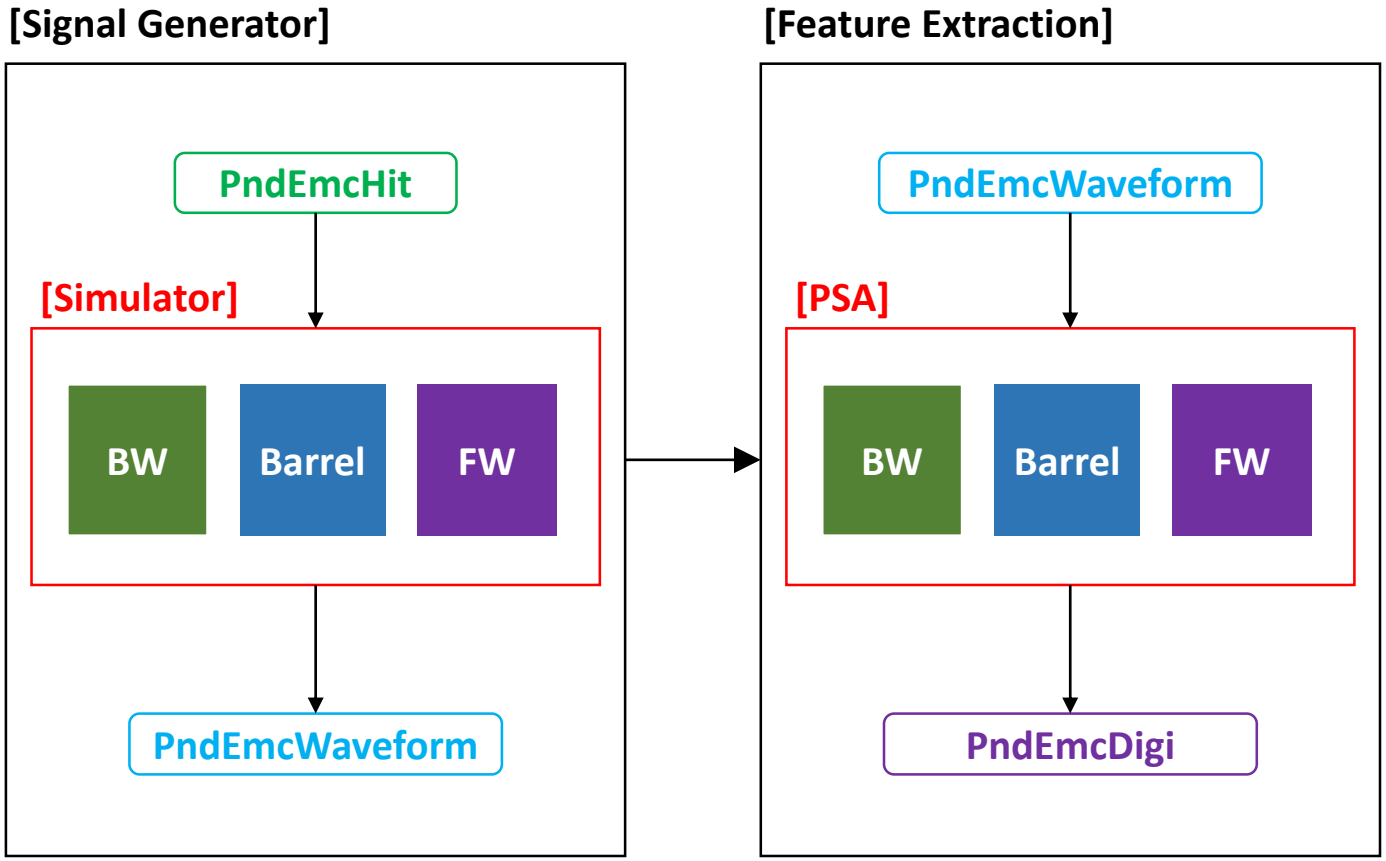
Newly added for BW

Discussion: how to combine digitization for all EMC models



- ✓ The interface classes handle the main work
- ✓ The overall procedure do not depend on concrete implementation

Discussion: how to combine digitization for all EMC models



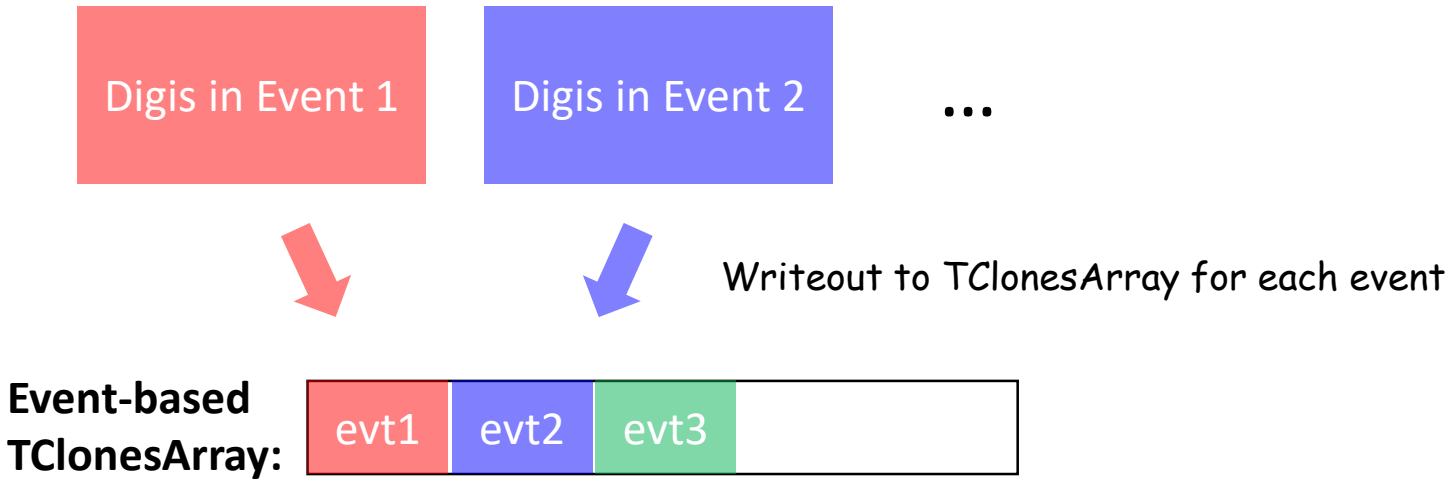
- ✓ Implement simulator/psa for other EMC modules
- ✓ Runtime choose the needed one based on detector ID

Summary and outlook

- **Have developed a digitization package for the backward endcap EMC**
- **The new code is finished and compiled. Preliminary tests show promising results. Will perform further test**
- **There are small updates about the feature extraction algorithms. Will upgrade to the latest one**
- **Need to integrate the code with other EMC modules. Further discussions will be needed**

Backups

The event-based simulation



PndEmcFullStackedWaveformSimulator

- determines length of simulation window via threshold
- if $threshold_{simulation} < threshold_{feature\ extraction}$: feature extraction will see "complete" pulses
- remaining drawback: rate of false-positive hits drastically reduced