Concept for Timebased Simulated Waveforms of the PANDA EMC

Philipp Mahlberg

Helmholtz-Institut für Strahlen- und Kernphysik

Universität Bonn

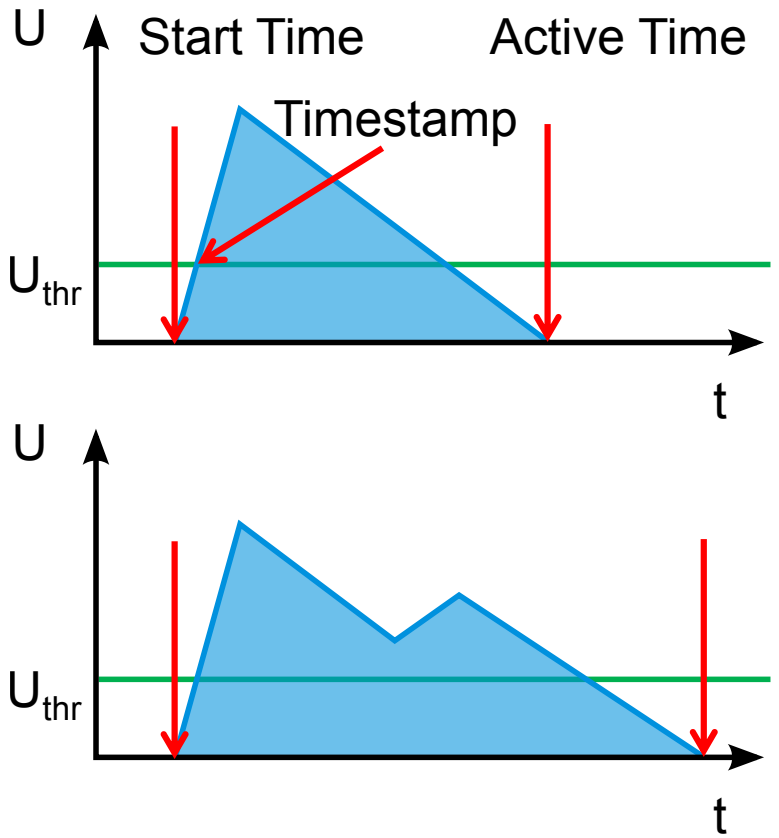
June 26, 2013

Outline

Classical timebased simulation concept

(as implemented in case of the MVD, STT . . .)

- pass to timebased buffers:
  - `digi object`
  - active time window, defined by [`startTime`, `activeTime`]
- (in time) overlapping `digis`: call function `Modifiy(oldData, newData)`
  - $\longrightarrow$ returns a single (combined) `digi object`
- store `digis` after `activeTime` has passed by

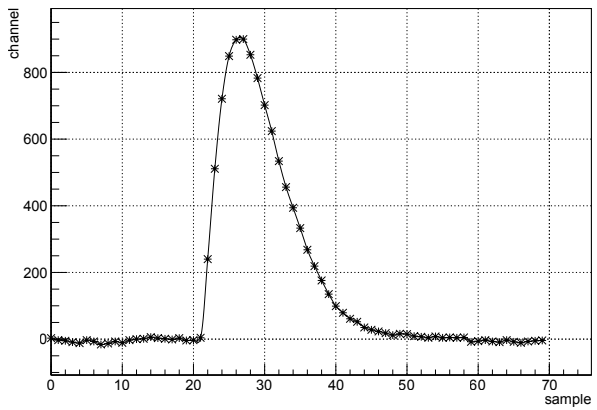


cf. Tobias Stockmanns: Time based simulation, Torino Computing Workshop, Jul. 2012

$\Rightarrow$ **suitable for lightweight** `digi objects`

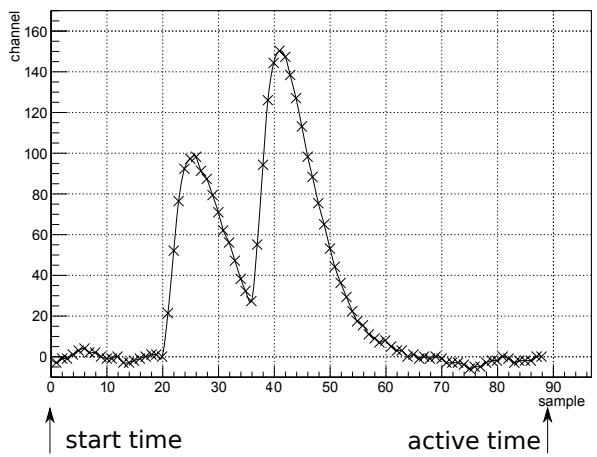
Timebased simulation of EMC waveforms

- simulate **analog signal** of photodetetectors (sampled and digitized by ADC)
- active time defines simulation window (experiment: continous data stream)



- use timebased simulation capabilities to construct waveforms consisting of multiple hits ($\longrightarrow$ pileup events)

$\rightarrow$ **study performance and systematics of feature extraction in physical context**



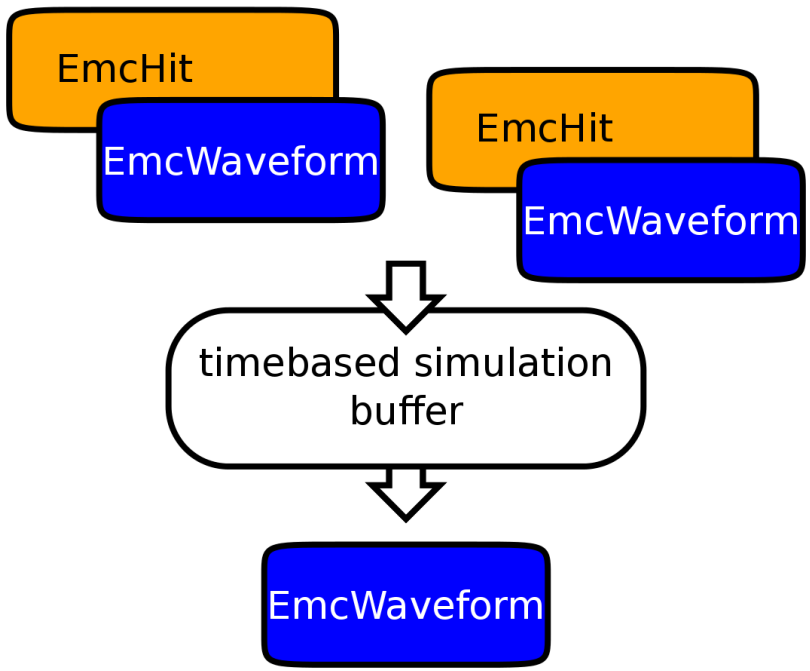
How to simulate pileup signals?

Using `Modify(...)` in case of analog signals:

$\longrightarrow$ summing up waveforms based on limited information

- pileup signal contains noise and digitization artifacts twice
- final waveform might be too short

$\longrightarrow$ unwanted systematic effects



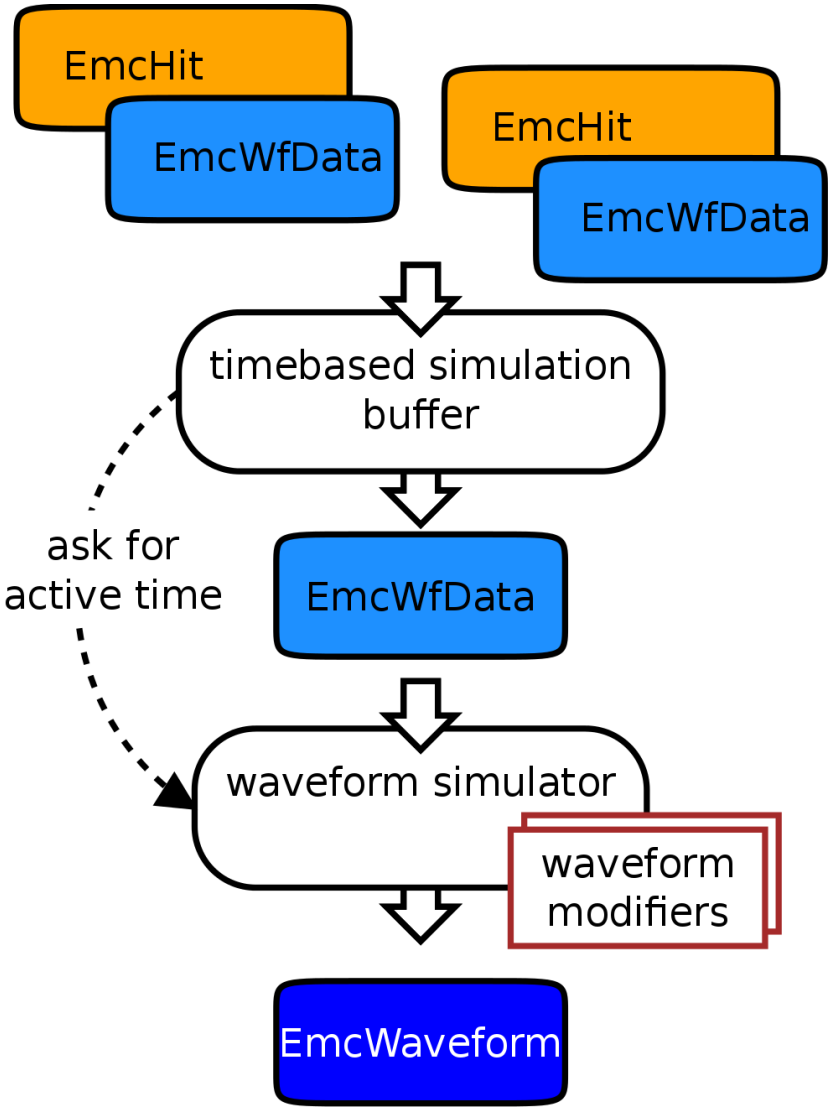
How to simulate pileup signals?

Decouple hit grouping and waveform simulation:

- `PndEmcWaveformData` stores hit information (energy, time)
- `Modify` adds additional hit

→ closer to `digi object` than waveform

- `PndEmcAbsWaveformSimulator` simulates waveform based on `PndEmcWaveformData`
- plug-able `PndEmcAbsWaveformModifiers` for noising, digitizing,...



Interplay between PndEmcWaveformData and PndEmcAbsWaveformSimulator

PndEmcWaveformData contains:

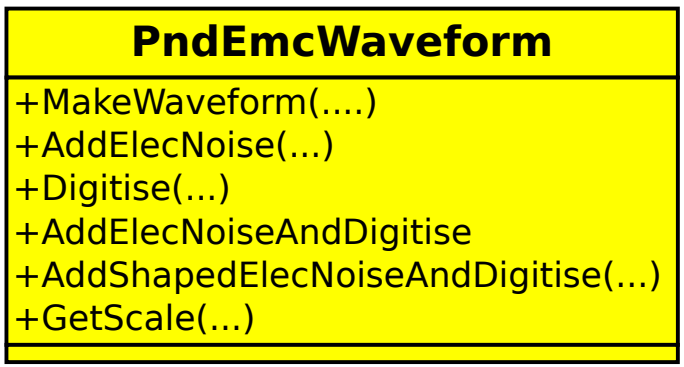
- time and energy information of underlaying hits
- reference to corresponding waveform simulators
- fulfills requirements on timebased simulation objects

PndEmcAbsWaveformSimulator provides:

- Simulate (PndEmcWaveformData*)
  - ↪ Construct PndEmcWaveform out of PndEmcWaveformData
  - ↪ Call PndEmcWaveformModifiers
- GetAbsoluteTimeInterval (PndEmcWaveformData*)
  - ↪ returns waveform simulation window

User's point of view

formerly:



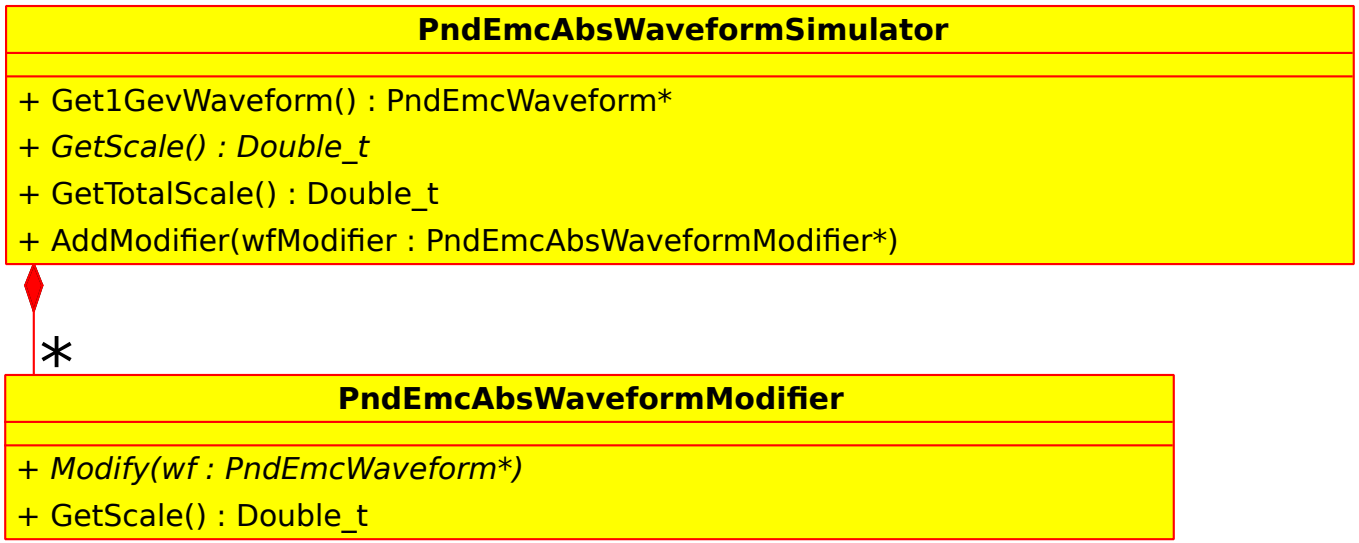

```
switch (module) {
   case 1:  // Barrel
   theWaveform->UpdateWaveform(...);
   break;
   case 2:  // Barrel
   theWaveform->UpdateWaveform(...);
   break;
   case 3:  // Fwd Endcap;
   theWaveform->UpdateWaveform(...);
   break;
}
```

- all simulation methods attached to `PndEmcWaveform` class
- user has to keep track of waveform type (Barrel, FwEndcap APD/VPTT,...) to supply correct parameters to waveform modification method

inflexible, uncomfortable?

User's point of view

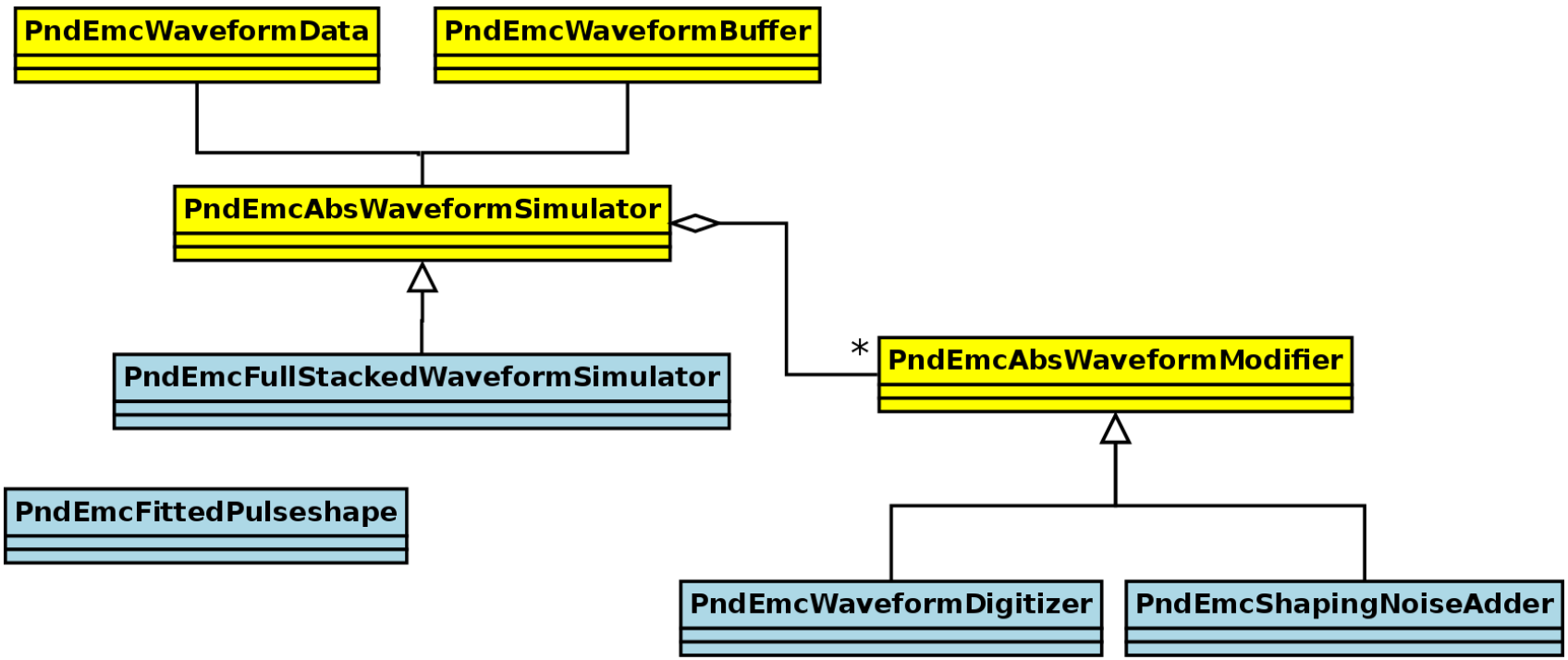
using waveform simulators:



1. init waveform simulators
2. plug in waveform modifiers
3. pass hits information to `PndEmcWaveformBuffer`, referencing corresponding waveform simulator

$\Rightarrow$ receive timebased simulated waveforms

Class overview



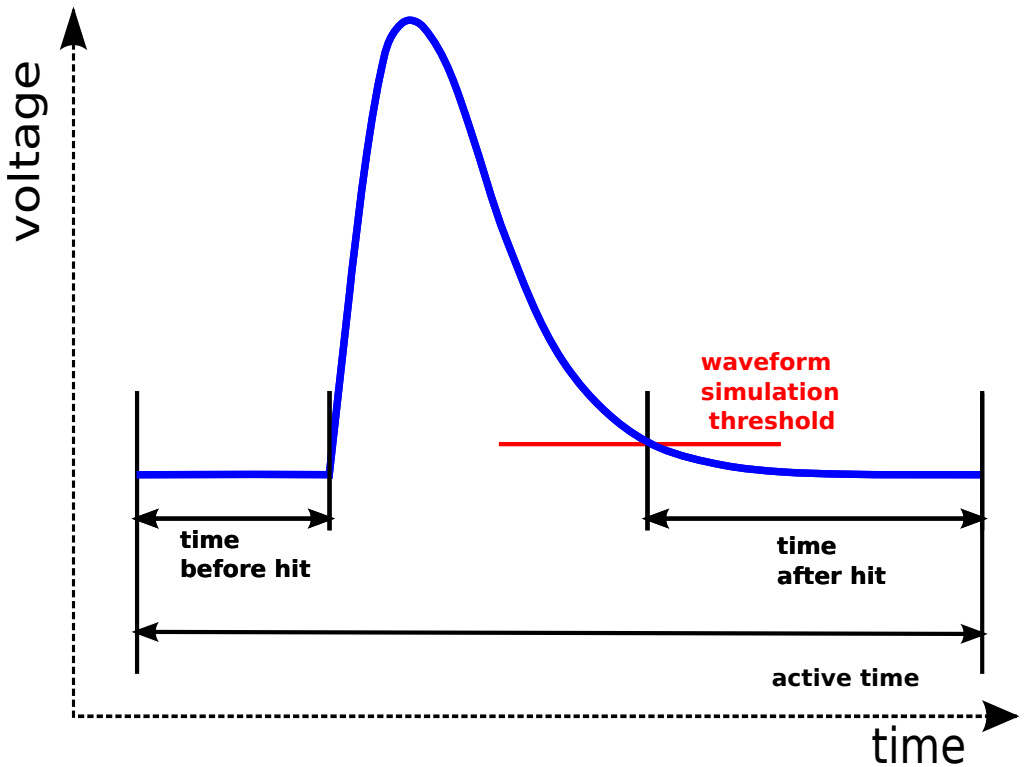
classes implementing concept

classes used for simulation of Forward Endcap waveforms

Simulation window length

`PndEmcFullStackedWaveformSimulator:`

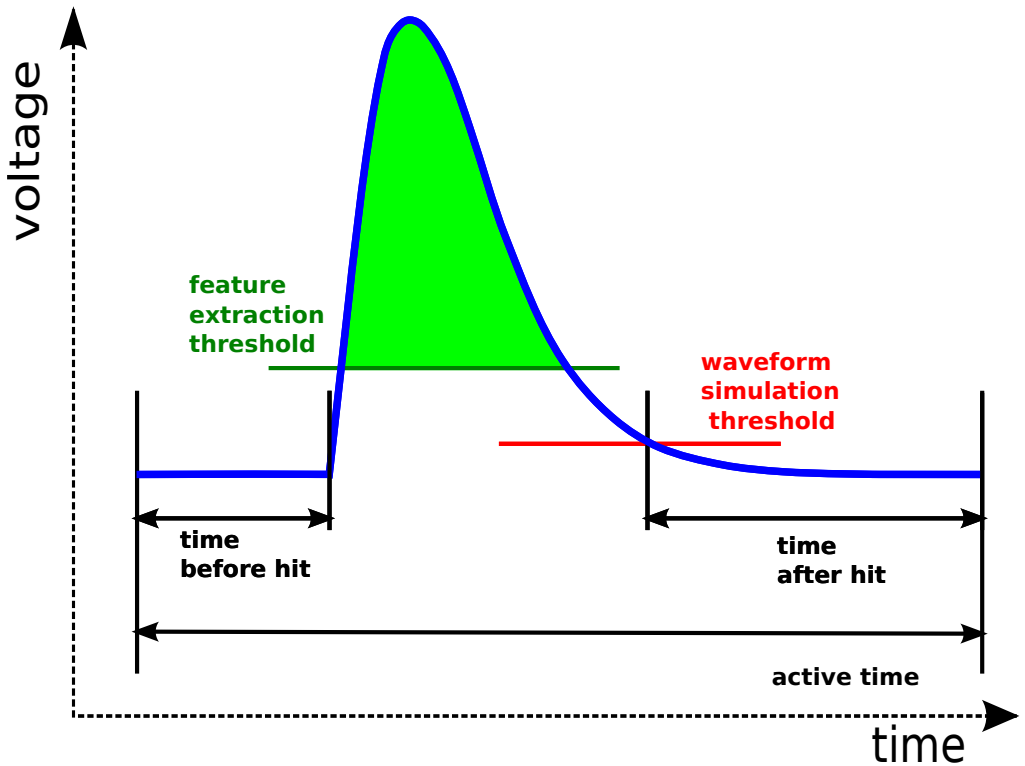
- determines length of simulation window via threshold
- if $threshold_{simulation} < threshold_{feature\ extraction}$: feature extraction will see "complete" pulses
- remaining drawback: rate of false-positive hits drastically reduced



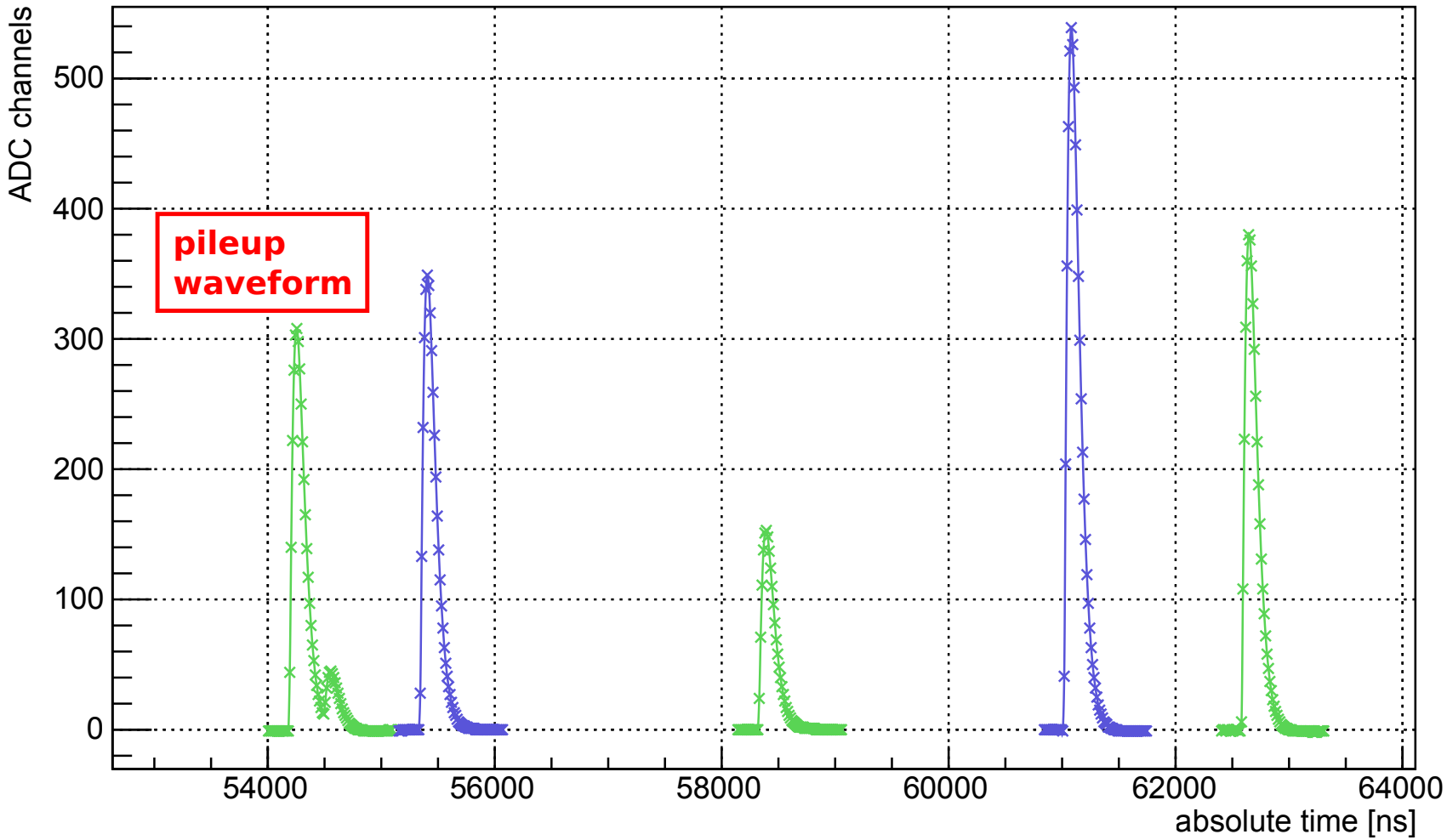
Simulation window length

`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



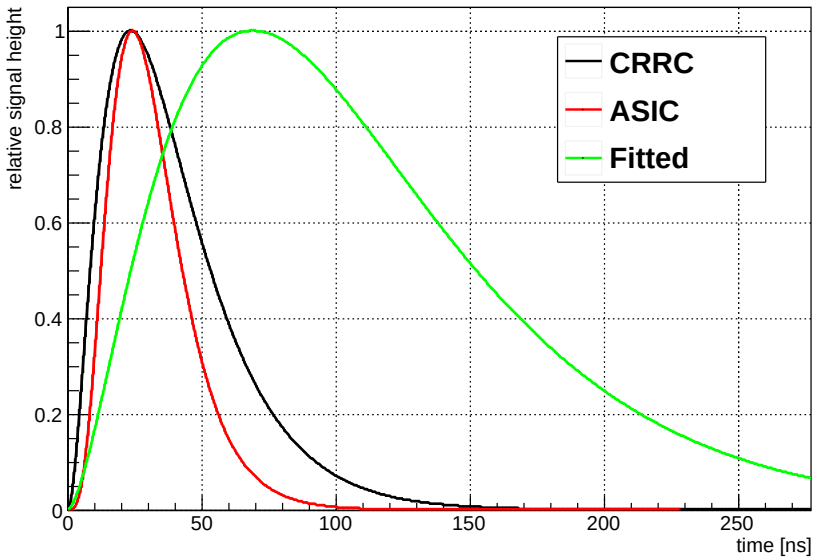
Simulated signals over absolute time



distinct simulated waveforms in green / blue color

Realistic parameters for the Forward Endcap

cf. talk in EMC session

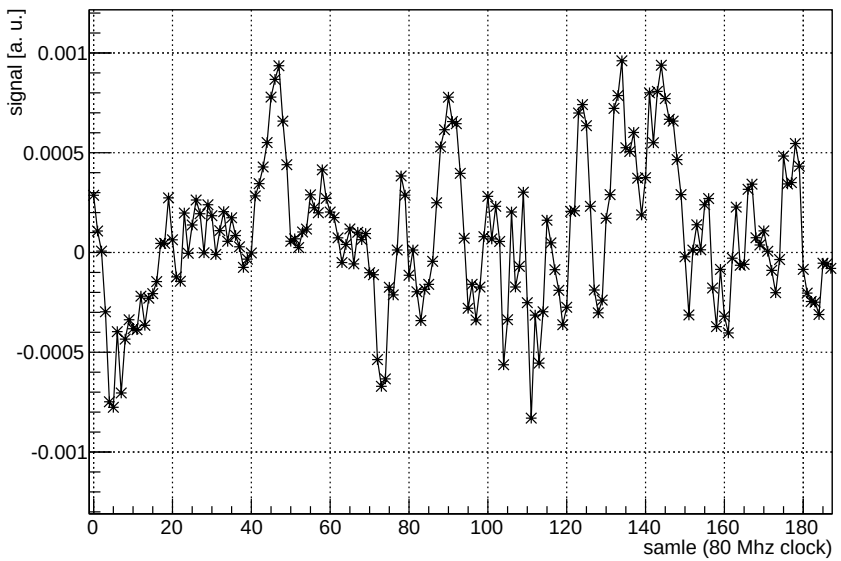


Pulseshapes used in PandaROOT:

- `CRRC`: Shashlyk calorimeter
- `ASIC`: everything else
- added fitted pulseshape to match with Forward Endcap conditions

implemented shaping noise:

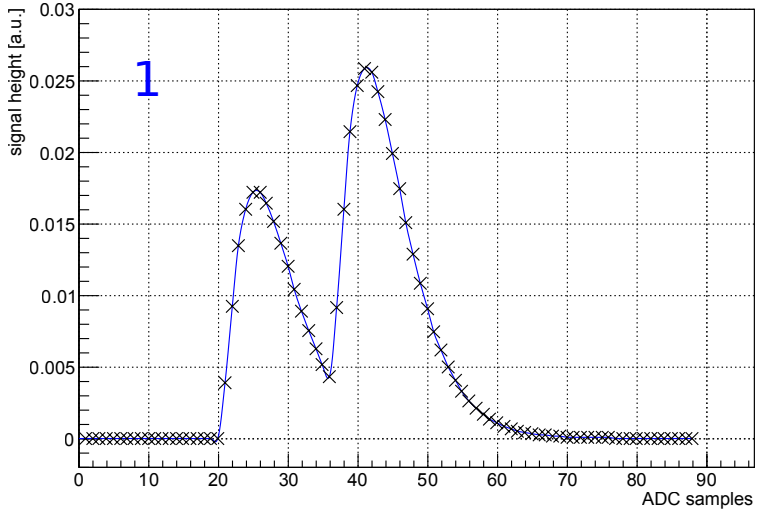
- main noise source
- noise is correlated in time
- until then in PandaROOT: only white noise

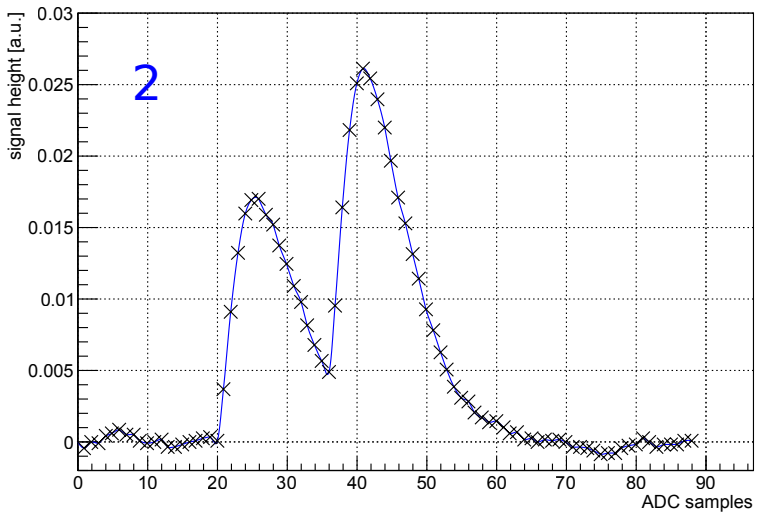


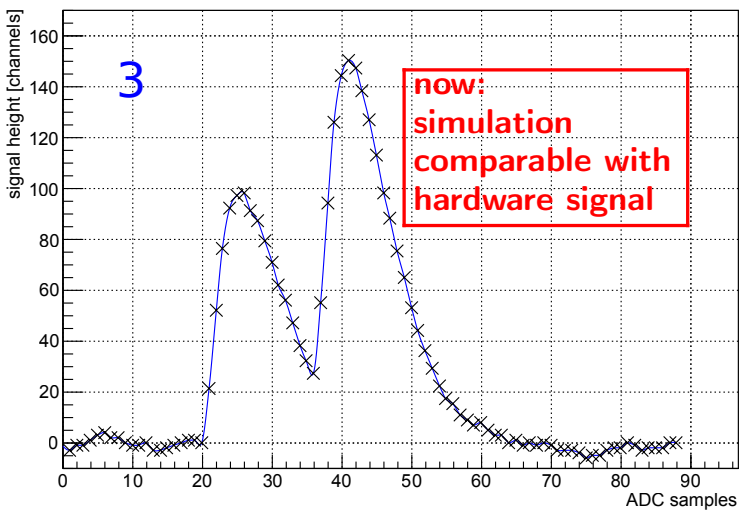
Exemplary signal simulation

pileup in VPTT (lowgain) channel
$E_1 = 92\,\text{MeV}$, $E_2 = 130\,\text{MeV}$,
$\Delta t = 200\,\text{ns}$

1. plain signal simulation
2. shaping noise adding
3. digitization

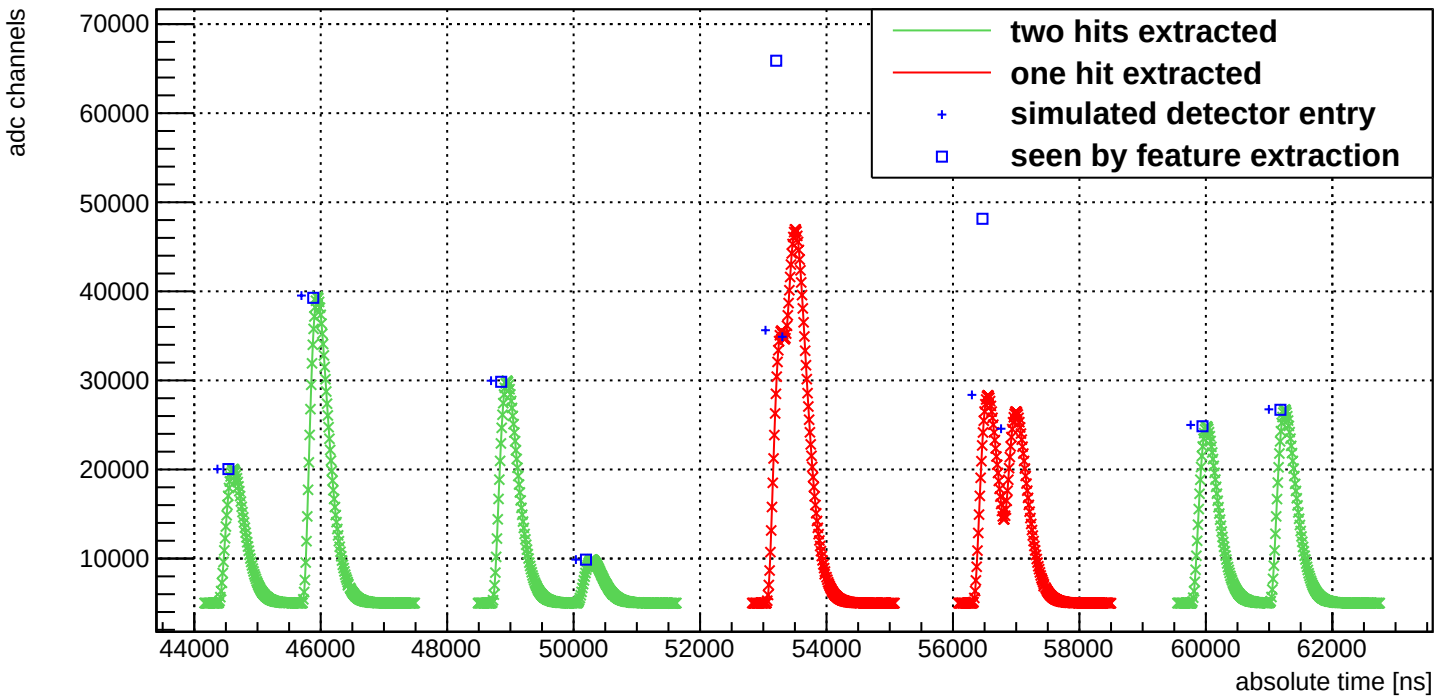






Testing FPGA based feature extraction

simulated two-hit waveforms (shown rates non realistic):

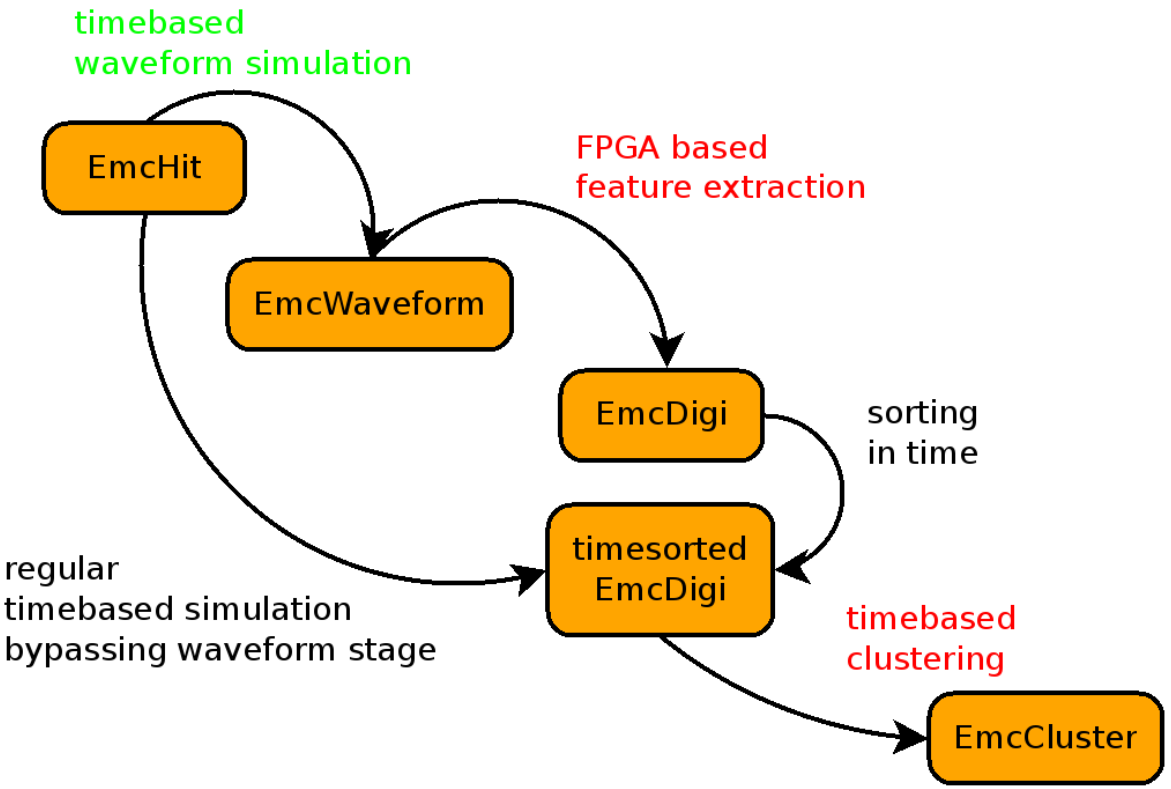


⇒ feature extraction works for well separated pulses

⇒ (real) pileup waveforms cannot be recovered (up to now)
more recent version offers pileup detection
($\neq$ recovery of energy, time information)

Summary and Outlook

- implemented flexible and expendable simulation of waveforms for the EMC for PandaROOT
- tuned simulation parameters for the Forward Endcap



needed to perform physical simulations:

- feature extraction capable to recover pileup pulses
- timebased clustering algorithm

Backup slides

How to use it – Setting up the waveform simulator

```
/* create Pulseshape*/
Double_t tau = 68.7;     //ns
Double_t N = 1.667;
PndEmcAbsPulseshape* fPulseshape = new PndEmcFittedPulseshape(tau, N);

Double_t sampleRate = 0.08;    //80 Mhz
Int_t nBits = 14;

Double_t energyRange = 15.;    //GeV
Double_t noiseWidth = 0.002;     //GeV

Double_t samplingBeforeFirstPulse = 250;     //ns
Double_t samplingAfterLastPulse = 250;     //ns
Double_t cutoff = 0.001;    //GeV
Double_t activeTimeIncrement = 50.;    //ns


/* set up waveform simulator */
PndEmcFullStackedWaveformSimulator* wfSim = new PndEmcFullStackedWaveformSimulator(
        sampleRate, fPulseshape);
wfSim->Init(samplingBeforeFirstPulse, samplingAfterLastPulse, cutoff, activeTimeIncrement);

/* add waveform modifiers */
wfSim->AddModifier(new PndEmcShapingNoiseAdder(wfSim->GetPulseRaiseTime(), sampleRate,
        noiseWidth, wfSim->GetTotalScale()));
wfSim->AddModifier(new PndEmcWaveformDigitizer(nBits, energyRange, wfSim->GetTotalScale()));
```

How to use it – Setting up buffer and pass data to it

```
/* Create and activiate timebased simulation buffer */
PndEmcWaveformBuffer* wfBuffer = new PndEmcWaveformBuffer("EmcWaveform", "
        PndEmcWaveform", "Emc", fStoreWaves);
wfBuffer = (PndEmcWaveformBuffer*) ioman ->RegisterWriteoutBuffer("EmcTimebasedWaveform",
        wfBuffer);
```

hand over energy and time information of `EmcHit theHit` to it

```
/* create waveform data object */
PndEmcWaveformData* wfData = new PndEmcWaveformData(theHit->GetDetectorID(), fWfSim);

/* attach FairLink, hit energy, absolute hit time and corresponding waveform simulator to it */
FairLink linkToHit(-1, ioman->GetEntryNr(), "EmcHit", iHitIndex, 1.0);

//ATTN: timebased simulation framework uses nano second, emc second as time unit
wfData->AddHit(linkToHit, eventTime + theHit->GetTime()*1.0e9, theHit->GetEnergy());

/* pass waveform data object to timebased simulation buffers */
wfBuffer->FillNewData(wfData);
```

Sequence diagram of timebased waveform generation