EMC simulation/reconstruction in pandaroot

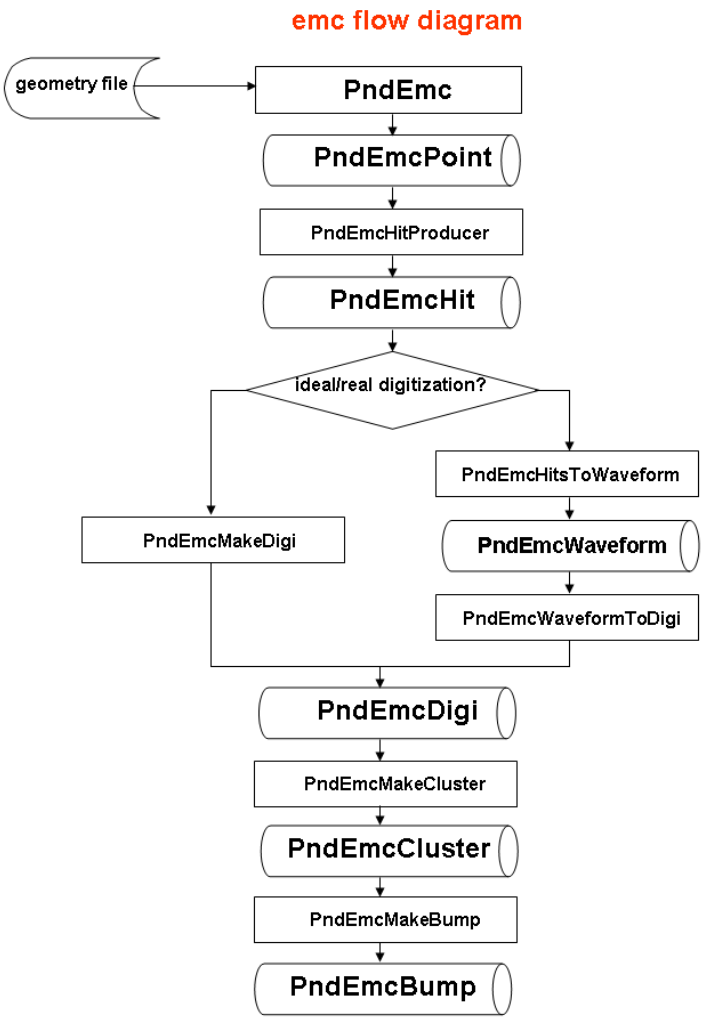
D. Melnychuk

24.07.2012

Plan

1. Simulation/geometry
2. Digitization
3. Reconstruction (cluster reconstruction, bump splitting)
4. Energy correction
5. Open problems

Data flow in EMC simulation



PndEmcHit

Double32_t fEnergy; // hit amplitude
Double32_t fTime; // time
Int_t fDetectorID; ///< Detector unique identifier
Double32_t fX, fY, fZ; ///< Position of hit [cm]

PndEmcDigi

Double_t fEnergy; // digi amplitude
Int_t fDetectorId;
Int_t fThetaInd;
Int_t fPhiInd;
TVector3 fWhere;

PndEmcCluster

std::vector<Int_t> fDigiList;
std::map<Int_t,Int_t> fMemberDigiMap; // Map <detId,digiIndex>
std::map<Int_t,Int_t> fLocalMaxMap; // Map<detId, digiIndex>
for the maxima
Double_t fEnergy;
TVector3 fWhere;
unsigned fNbumps;
Double_t fZ20; // Zernike moment (2,0)
Double_t fZ53; // Zernike moment (5,3)
Double_t fLatMom; // Lateral energy deposition within the cluster

Simulation/geometry

In pandaroot simulation EMC is divided into 5 modules:

1. Barrel EMC - downstream
2. Barrel EMC - upstream
3. Forward endcap
4. Backward endcap
5. Shashlyk calorimeter

Different versions of geometry exist for different modules, both in ascii and root format. Version of geometry is selected in simulation macro with:

```
PndEmc *Emc = new PndEmc("EMC",kTRUE);
Emc->SetGeometryVersion(1);
```

Default geometry is barrel EMC -ascii file, forward/backward endcap and shashlyk - root geometry. Dead material is included for forward endcap and shashlyk.

PndEmcHitProducer

- Monte Carlo simulation produces a TClonesArray of PndEmcPoint, which are created for each interaction of the particle with active volume. However for the following analysis it's sufficient to know the energy deposited in each crystal. This summation of energy from PndEmcPoint for each crystal is performed in PndEmcHitProducer.
- The size of TClonesArray of PndEmcPoint may be huge and therefore to save disk space Monte Carlo simulation and PndEmcHitProducer runs in the same macro and PndEmcPoints are made transient, i.e. not stored to the disk.
- In PndEmcHitProducer the non-uniformity of the light collection in crystals can be taken into account based on experimental data. It can be switch on with the parameter

```
Use_nonuniformity:Int_t 1
```

in /macro/params/emc.par.

Running simulation

All the emc related macros are located in /macro/emc. In the subfolder /macro/emc/dedicated/ there are macro which analyse output of different stages in simulation/digitization/reconstruction/. Simulation macro

```
root [0] .x sim_emc.C
```

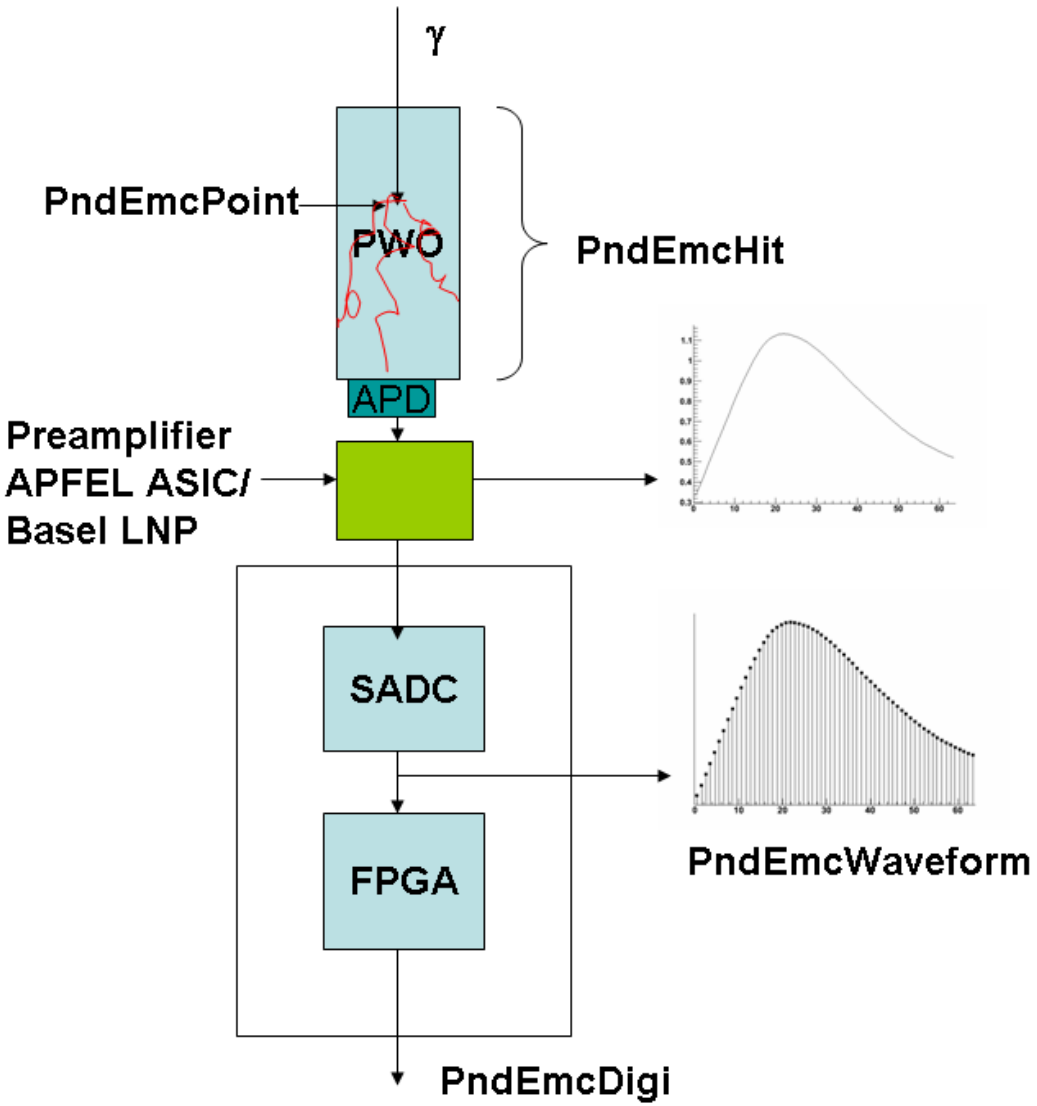
runs simulation of 10 events with 1 GeV photons in the range $\theta \in [0°180°]$. By default only emc geometry is included. To include the whole PANDA geometry the flag inside the macro should be set

```
Bool_t useFullPANDAGeometry=true;
```

To run simulation for a given number of events and momentum of photon:

```
root [0] .x sim_emc.C(nEvents, momentum)
```

Digitization



Pulseshapes, which correspond to output of preamplifier are produced (exponential, CR-RC, CR-2RC options) and are digitized with Nbits. Their amplitude is smeared with photon statistics and electronic noise is added to each bin. Then pulseshapes are analysed and their amplitude stored to PndEmcDigi::fEnergy (scaled by amplitude of 1 GeV signal). Two options for pulseshape analysis exist: simple parabolic fit and digital filters. Parameters relevant for digitization are stored in /macro/params/emc.par (all.par).
Feature extraction algorithm developed in KVI for implementation on FPGA still need to be implemented in pandaroot.

Preliminary version of time-based digitization is implemented.

Running digitization

Digitization macro:

```
root [0] .x digi_emc.C
```

time-based digitization:

```
root [0] .x digi_emc_timebased.C
```

Reconstruction

- Auxiliary classes (PndEmcTwoCoordinateIndex, PndEmcMapper, PndEmcStructure)
- Cluster reconstruction
- Calculation of cluster properties
- Calculation of cluster position
- Bump splitting

PndEmcTwoCoordinateIndex

- To enumerate crystals within cluster reconstruction algorithm the PndEmcTwoCoordinateIndex class is introduced, which combine one integer index with two indexes in 2D-plane in one object. Indexes are accessed with the following methods

```
PndEmcTwoCoordinateIndex::XCoord(), YCoord(), Index()
```

- Integer index of the crystal includes information about the following numbers (module, row, copy, crystal):

```
detectorID = module*100000000 + row*1000000 + copy*10000 +
crystal;
```

Here the module is for 1/2 - Barrel downstream/upstream, 3 - forward endcap, 4 - backward endcap, 5 - shashlyk calorimeter. Copy number reflects the fact that barrel EMC geometry is built by copying one slice 10×72 crystals in ϕ direction. Endcap are built from 4 copies wrt. beam pipe.

PndEmcMapper

- For cluster reconstruction algorithm it is necessary to determine if two crystals are neighbours in 2D-plane ((θ, ϕ) for barrel or (X,Y) for endcaps and shashlyk).
- Crystal is characterized by a single integer index and PndEmcMapper maps this index to two indexes in 2D-plane.
- PndEmcMapper should be initialized only once in program with

```
PndEmcMapper::Init(mapVersion)
```

and then can accessed with

```
PndEmcMapper* PndEmcMapper::Instance ()
```

- PndEmcMapper is implemented for each module in a separate sub-class and then mappers are combined with Add() method. Each combination of map versions for different modules define *mapVersion*.
- Main function of the PndEmcMapper is PndEmcTwoCoordIndex* GetTCI(Int_t DetectorId) which return *PndEmcTwoCoordIndex* object for integer index of the crystal.

PndEmcStructure

- PndEmcStructure provides an access to EMC geometry for reconstruction task. It uses as an input the TGeoManager containing EMC geometry created at simulation stage and produce the map

```
std::map<PndEmcTwoCoordIndex*, PndEmcXtal*> fTciXtalMap;
```

  between crystal indexes expressed in PndEmcTwoCoordIndex and PndEmcXtal objects which contain geometrical information for each crystal.
- PndEmcXtal contain information about position of the crystal (TVector3), its orientation (TGeoRotation) and its shape (TGeoTrap).
- Useful function in PndEmcStructure is *locateIndex*(*double theta*, *double phi*), which find the crystal at the given direction (θ, ϕ).
- PndEmcStructure follows a singleton design pattern, i.e. it is initialized at the first call of PndEmcStructure::Instance() and for subsequent calls previously initialized PndEmcStructure is used.

Cluster reconstruction

- Electromagnetic shower during its development deposes energy in a set of neighboring crystals which is called cluster.
- Cluster reconstruction is performed in the PndEmcMakeCluster task. Algorithm loops over all digis and check if it should belong to existing clusters, i.e. it is a neighbour of any crystal in existing clusters and its energy is above the threshold (3 MeV). If yes, it is added to existing cluster, if no, new cluster is created.
- Properties of the cluster are calculated in separate classes, derived from one abstract interface class *PndEmcAbsClusterProperty*. To calculate properties TClonesArray of PndEmcDigi should be accessible.

Reconstruction macro:

```
root [0] .x reco_emc.C
```

Cluster properties

The following cluster properties can be calculated:

- PndEmcClusterProperties: energy, position and mass.
- PndEmcClusterEnergySums: energy of maximum digi, energy of 9 central crystals and 25 central crystals
- PndEmcClusterDistances: radial and angular distance between the digi and the cluster centroid
- PndEmcClusterMoments: first and second moment of energy with respect to centre, minor and major axis of the cluster
- PndEmcXClMoments: Zernike moment

Example

```
TFile* f = new TFile("cluster_emc.root"); //file you want to analyse
TTree *t=(TTree *) f->Get("cbmsim") ;
TClonesArray* cluster_array=new TClonesArray("PndEmcCluster");
t->SetBranchAddress("EmcCluster",&cluster_array);

t->AddFriend("cbmsim", "digi_emc.root");
TClonesArray* digi_array=new TClonesArray("PndEmcDigi");
t->SetBranchAddress("EmcDigi",&digi_array);

// Make geoManager accessible for PndEmcStrucrure
TFile* file = new TFile("simparams.root");
file->Get("FairBaseParSet");

PndEmcMapper::Init(6);

for (Int_t j=0; j< t->GetEntriesFast(); j++)
{
t->GetEntry(j);
for (Int_t i=0; i<cluster_array->GetEntriesFast(); i++)
{
    PndEmcCluster *cl=(PndEmcCluster*)cluster_array->At(i);
    PndEmcXClMoments *xClMoments = new PndEmcXClMoments(*cl, digi_array);
    Double_t Z42=xClMoments->AbsZernikeMoment(4, 2, 15);
    std::cout<<"Z42="<<Z42<<std::endl;
}
}
```

Cluster position

The spatial position of a cluster is calculated via a center-of-gravity method:

$$X = \frac{\sum_i w_i x_i}{\sum_i w_i}$$

The radial energy distribution, originating from a photon, decreases approximately exponentially. Therefore, a logarithmic weighting is used:

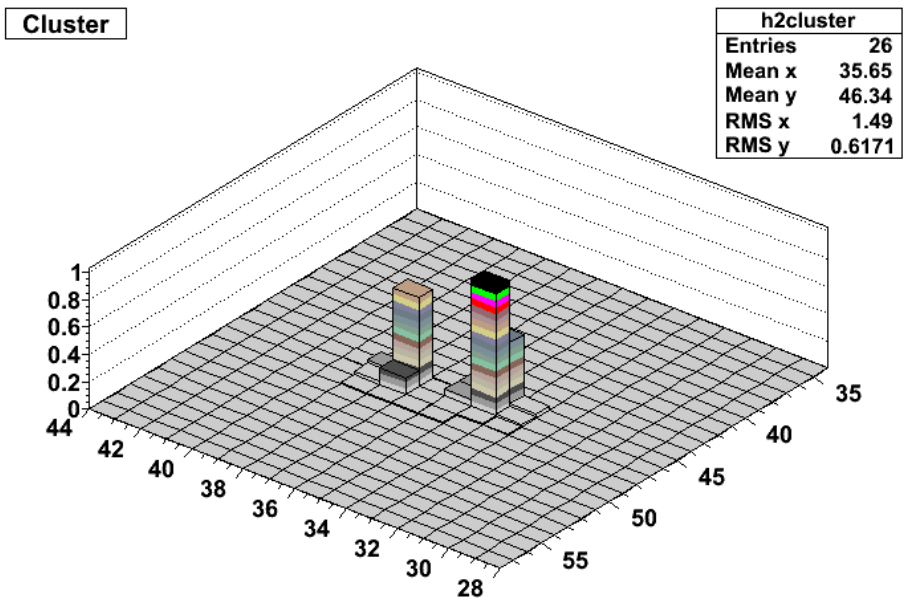
$$w_i = max(0, A(E_{cluster}) + \ln \frac{E_i}{E_{cluster}})$$

$A(E_{cluster})$ is an energy dependent cutoff parameter which guarantees that the weights are positive and removes crystals with very low energy.

$$A(E_{cluster}) = a - b \cdot e^{-c \cdot E_{cluster}}$$

where the currently used values of parameters are a=3.6, b=1.549, c=2.543 and still need to be optimized.
Cutoff parameter varies between 2.1 for the lowest and 3.6 for the highest photon energies. That means, only crystals with a deposited energy of more than 12% (for lowest energy)/ 2.7% (for highest energy) of the total cluster energy are considered to calculate the photon position.

Bump splitting



π^0 with high energy can produce in EMC single cluster with two local maxima (bumps) for each of the γ from the decay $\pi^0 \to \gamma\gamma$

Two step algorithm:

1. Finding of local maxima
2. Digis are shared between bumps in iterative procedure.

Weights

$$w_{i,d} = \frac{E_d \cdot e^{-2.5 \cdot r_{i,d}/R_m}}{\sum_j E_d \cdot e^{-2.5 \cdot r_{j,d}/R_m}}$$

where $r_{i,d}$ - distance between d-th digi and i-th bump, R_m - Moliere radius of the material.

Bump splitting

Bump splitting is performed within two tasks:

1. PndEmc2DLocMaxFinder - find local maxima
2. PndEmcExpClusterSplitter - share digis between bumps

They are combined into one wrapper task PndEmcMakeBump. This task runs in the same macro with cluster reconstruction (PndEmcMakeCluster):

```
root [0] .x reco_emc.C
```

Energy correction

Emc cluster energy correction should correct for the energy leakage in dead material and via rear side of crystals. The energy leakage is dependent on energy and polar angle. Two methods for corrections are implemented at the moment and they can be called via one class PndEmcClusterCalibrator.

```
PndEmcAbsClusterCalibrator * calibrator=
PndEmcClusterCalibrator::MakeEmcClusterCalibrator(method, version);
```

where

1. method=1 - correction from the interpolation on 2-dimensional histogram
2. method=2 - correction from parametrization

The following parametrization is used $E_{\gamma,\mathrm{cor}} = E * f(\ln E, \Theta)$ and

$$\begin{aligned} f(\ln E, \Theta) = exp(a_0 + a_1\, lnE + a_2\, ln^2 E + a_3\, ln^3 E \\ + a_4\, cos(\Theta) + a_5\, cos^2(\Theta) + a_6\, cos^3(\Theta) \\ + a_7\, cos^4(\Theta) + a_8\, cos^5(\Theta) \\ + a_9\, lnE\, cos(\Theta)) \end{aligned}$$

Parameter "version" should take into account different versions of correction based for example on different geometries, including dead material or Geant3 vs Geant4. At the the moment only one version is provided based on Geant3 simulation.

Energy correction

The corrected values for energy and position of PndEmcCluster cluster are accessed:

```
Double_t energy_corrected= calibrator->Energy(cluster);
TVector3 v_corr = calibrator->Where(cluster);
```

The macros to calculate correction parameters are located in */macro/emc/dedicated/EnergyPosCorrection*

Calibration based on π^0 peak reconstruction is implemented by B.Roth (http://ep1.rub.de/ bernhard/emc_calibration) but still is not included in pandaroot.

Open problems/tasks

1. Simulation

   Implementation of dead material for barrel and backward endcap.

2. Digitization
   - Time-based simulation including pile-up study at the level of waveform.
   - Implementation of feature extraction algorithm developed for FPGA at KVI.

3. Reconstruction
   - Cluster reconstruction algorithm after time-based digitization
   - Pre-Shower detection and correction
   - Split-Off recognition (hadronic/electromagnetic)

4. + Neverending optimization of existing code wherever it's necessary.