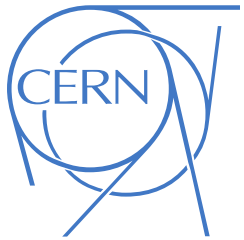
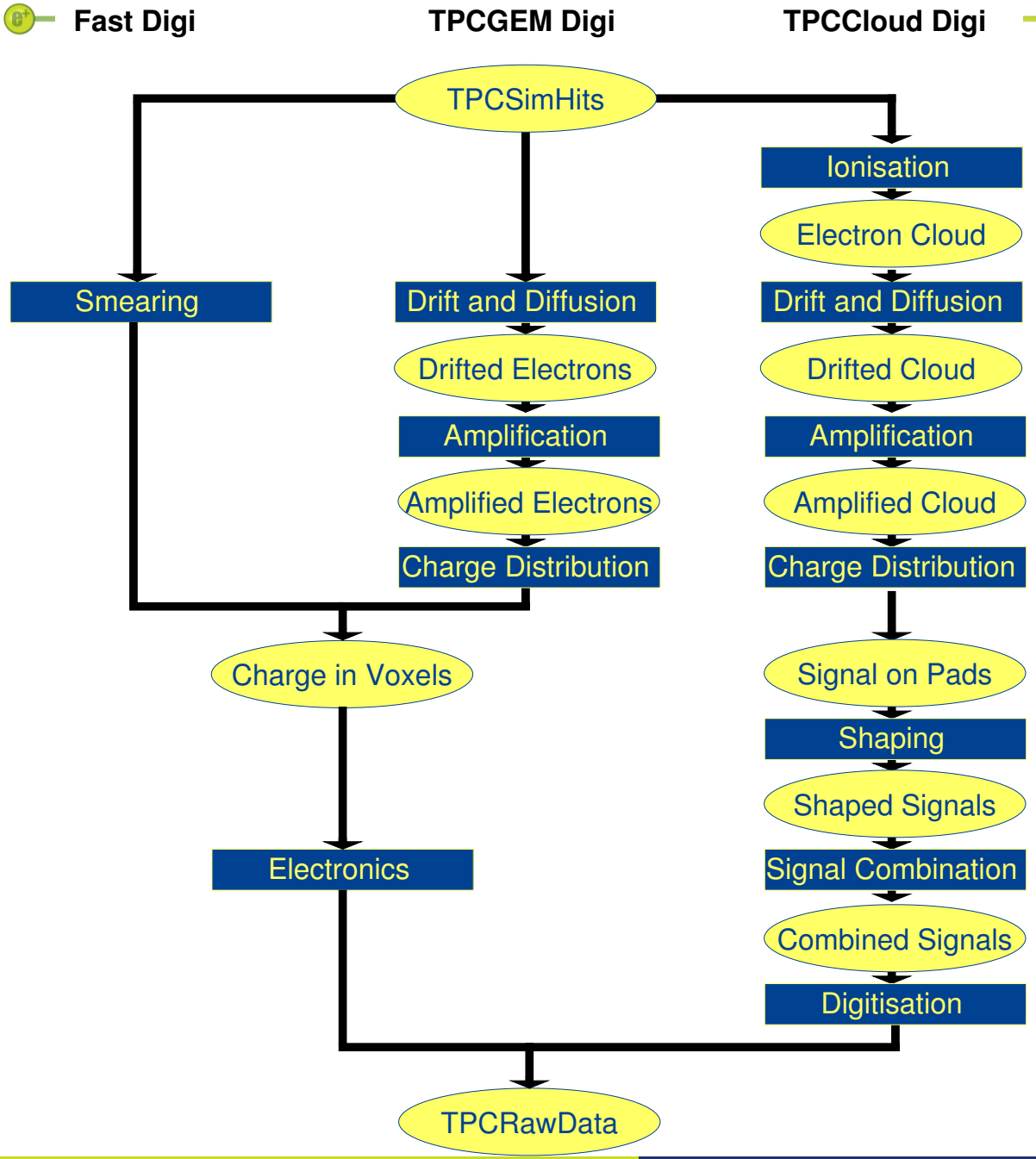
Simulation and Digitisation for MarlinTPC

Martin Killenberg



LCTPC Collaboration Meeting
DESY, Hamburg, 26. 3. 2011

Simulation and Digitisation Overview



Fast Digi

(MokkaToVoxelProcessor)

- Charge distribution without gas gain fluctuations

TPCGEM Digi

- Drift every single electron
- Voxel map allows overlay of multiple events (bunch train)
- Also works for pixel readout

TPCCloud Digi

- Drift one charge cloud per SimHit

This talk:

- TPCGEM Digi
- Summary of the changes during the last two years

Simulation 1

PrimaryIonisationProcessor

Old version

- One SimTrackerHit per electron
- Energy information not used

Current version

- One SimTrackerHit per cluster
- 26 eV per electron

$\Rightarrow$ Compatible with Mokka output

$\Rightarrow$ Smaller data size

Drift processor had to be adapted

- Calculate number of electrons from energy deposit
- Drift every individual electron
- Most clusters have only a few (1, 2, 3) electrons
- More realistic than Gaussian smearing

Simulation 2

Mokka

Default mode

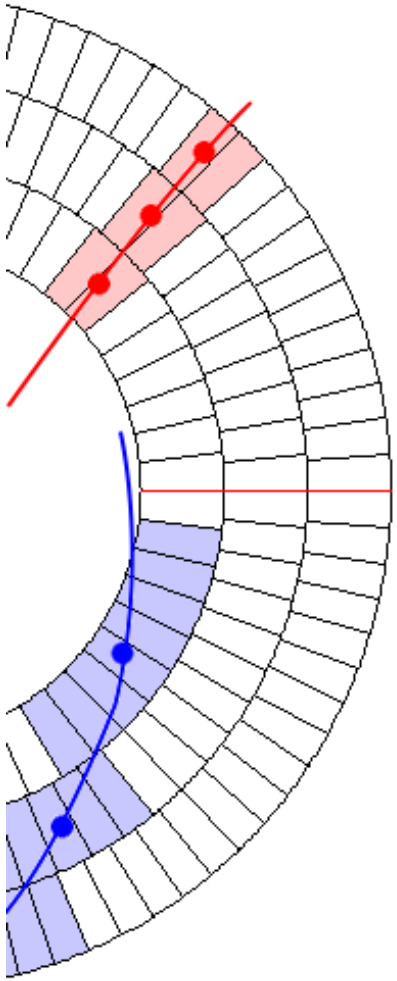
- One hit in the middle of the pad row
- Energy deposit is stored
- – Pad response cannot easily be calculated

`lowPt` mode

- Introduced for low energetic particle curling within one pad row
- Step length is limited (default 1 mm)

  For use with MarlinTPC
- Force all particles into this mode ("low" p_T threshold set to 3 TeV)
- Limit the step length to 50 µm

  Does this work?



Mokka TPC Driver

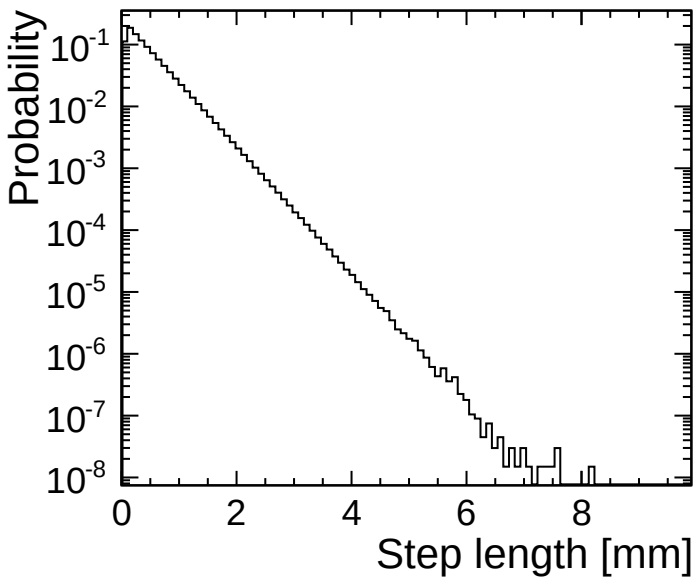
Geant4 uses parameterised energy loss

- Very small steps $\rightarrow$ very small energies (below ionisation threshold)
- Almost all steps are empty
- Not enough energy deposit

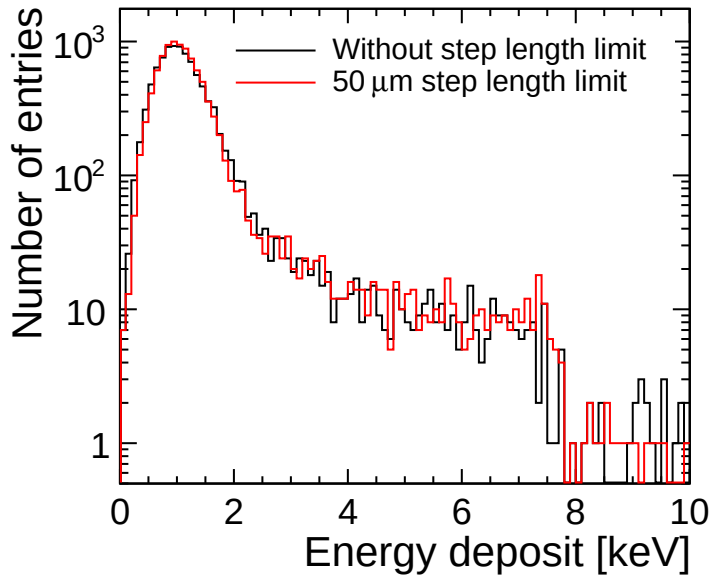
Solution:

- Save energy deposits below threshold for the next step, until ionisation threshold is reached

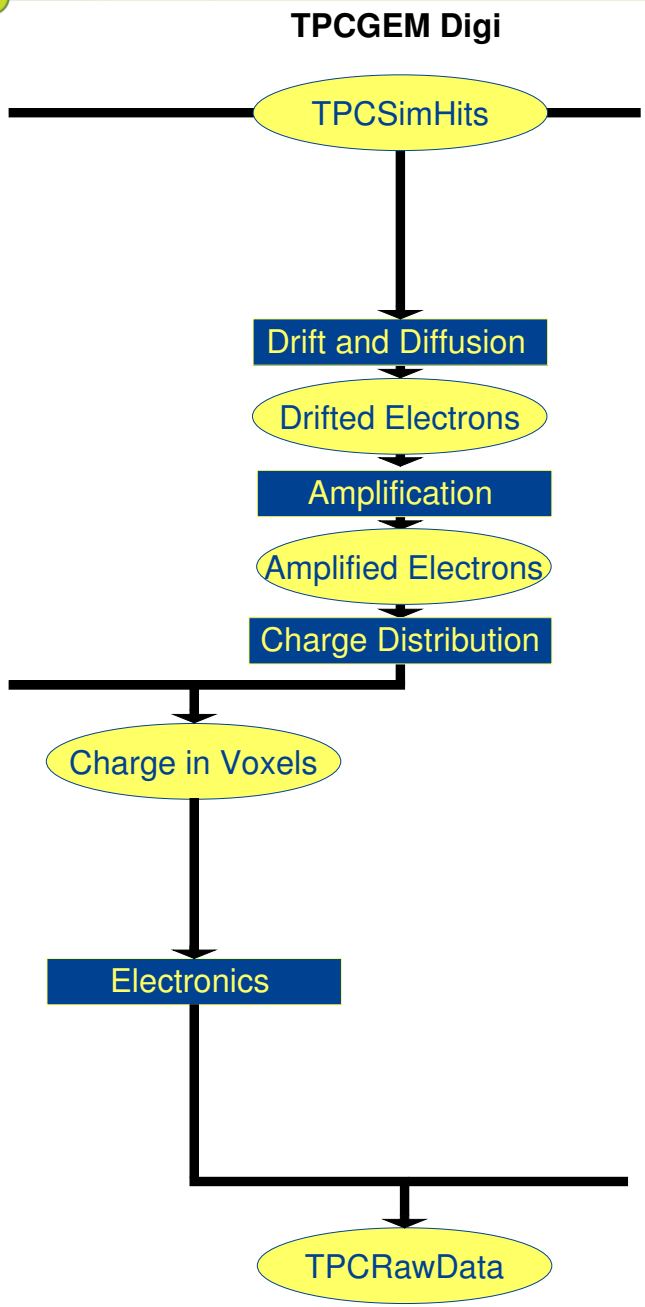
Distance between non-empty steps with 50 µm step length limit



Energy deposit in 5 mm Argon gas



Voxelisation



Processor Interface: `processEvent()`

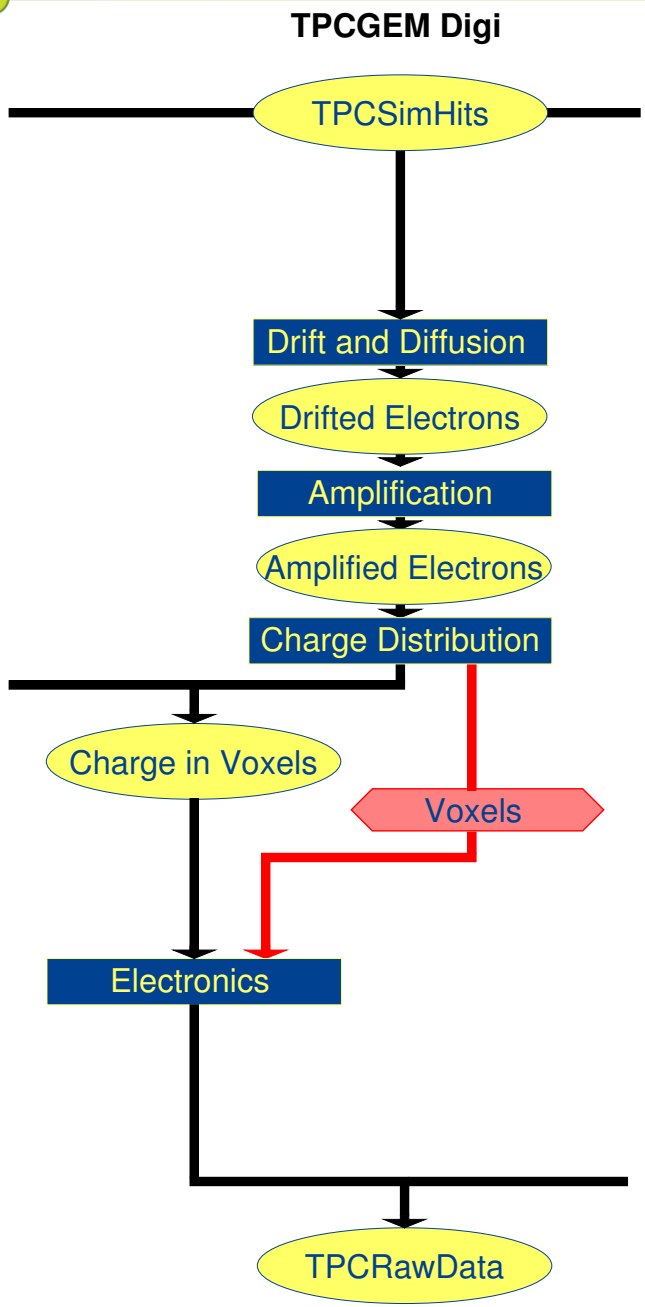
- Add a new lcio collection to the event (TPCVoxel, LCGenericObject)

Problem

- Voxel map internally uses only two integers

$\Rightarrow$ Data is duplicated, uses a lot of memory

Voxelisation



Processor Interface: `processEvent()`

- Add a new lcio collection to the event (TPCVoxel, LCGenericObject)

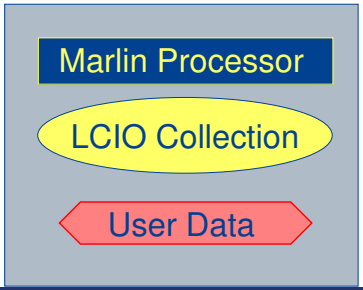
Problem

- Voxel map internally uses only two integers

$\Rightarrow$ Data is duplicated, uses a lot of memory

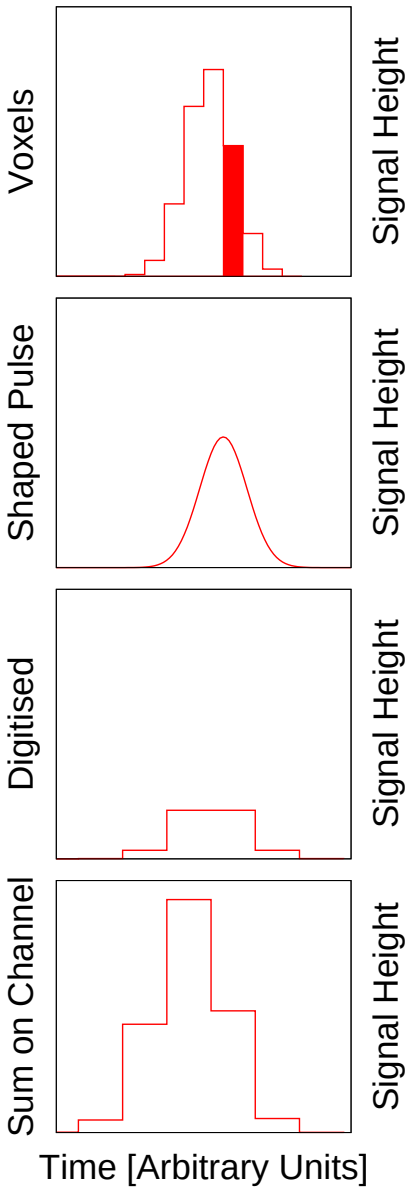
Solution

- Bypass the processor interface
- TPCElectronicsProcessor directly reads the lightweight two-integer representation using static `fetchVoxelCollection` function from ChargeDistributionProcessor



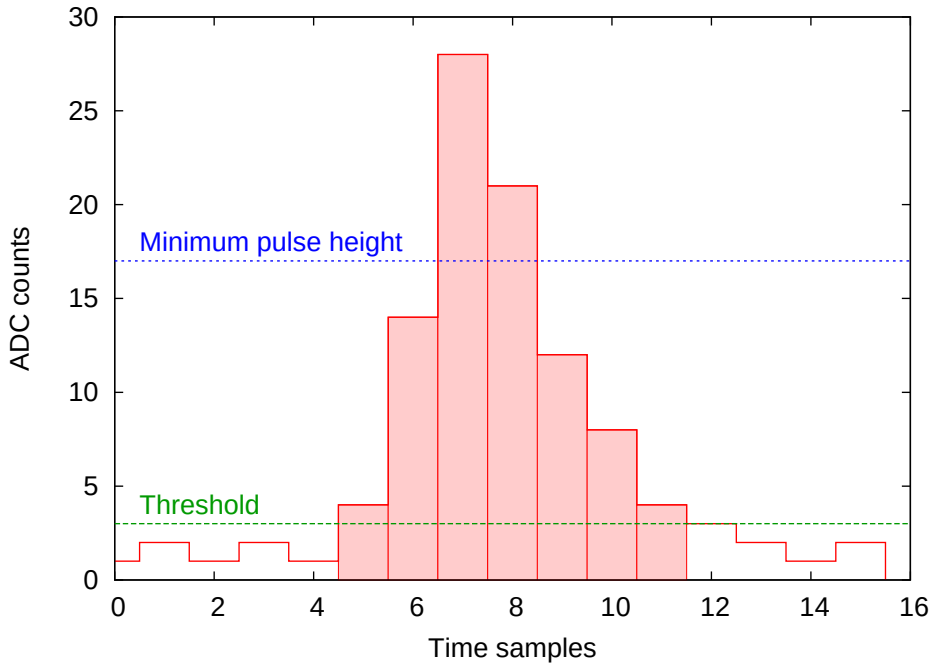
Memory consumption reduced by a factor 5 in this step!

TPCElectronicsProcessor



The TPCElectronicsProcessor simulates *n* bit charge sensitive FADC (e. g. ALTRO)

- For the charge **in each voxel** a Gaussian shaping is applied
- The shaped signal is digitised
- The digitised signals for all voxel in one channel are summed up
- Double-threshold zero suppression



- Threshold performs zero suppression.
- Minimum pulse height allows higher cut to accept a pulse.
- Allows low noise rate without cutting into the tails.

Background

Occupancy study: One full bunch train of beam induced background

- 100 million incoherent e^+e^- pairs
- 1000 $\gamma\gamma \rightarrow$hadrons events
- beam halo muons

1. Overlaying background and physics (or background for one event)
   - OverlayProcessor
2. Overlaying events in a bunch train
   - Voxel map in ChargeDistributionProcessor
   - ChargeDistributionProcessor does correct drift time offset

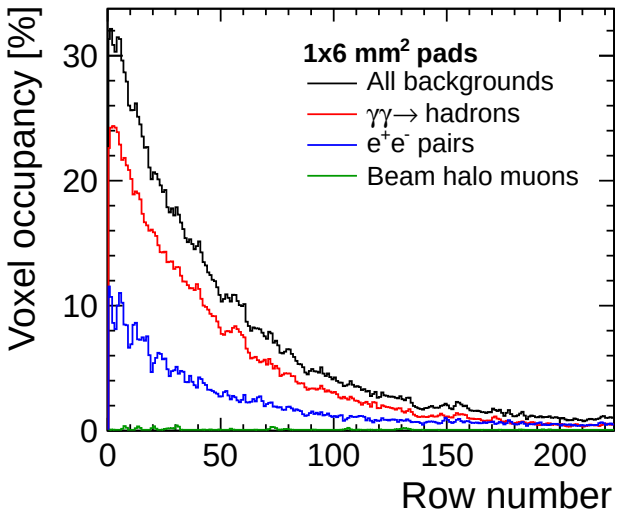
Background

Occupancy study: One full bunch train of beam induced background

- 100 million incoherent e^+e^- pairs
- 1000 $\gamma\gamma \rightarrow$hadrons events
- beam halo muons

1. Overlaying background and physics (or background for one event)
   - OverlayProcessor
2. Overlaying events in a bunch train
   - Voxel map in ChargeDistributionProcessor
   - ChargeDistributionProcessor does correct drift time offset

Problem: One BT of one component is already too large for the memory



Background

Occupancy study: One full bunch train of beam induced background

- 100 million incoherent e^+e^- pairs
- 1000 $\gamma\gamma \rightarrow$hadrons events
- beam halo muons

1. Overlaying background and physics (or background for one event)
   - OverlayProcessor
2. Overlaying events in a bunch train
   - Voxel map in ChargeDistributionProcessor
   - ChargeDistributionProcessor does correct drift time offset

   Problem: One BT of one component is already too large for the memory
   - Digitise reach component separately
   - Write out after each event
   - Digitise with higher ADC resolution to avoid threshold effects
3. Overlay digitised raw data
   - OverlayRawDataProcessor (just merge RawData collections, no overlap check)
   - MergeRawDataProcessor (merge overlapping pulses, apply correct thresholds)
   - Can this be used for electronics noise?

Work in Progress

Make digitisation aware of multiple module

- Extend the voxel map to have channel + module as key
- Almost ready, working version in Bo's branch

Handle second half TPC correctly

- Comes almost for free once the multi module option is ready

Implement pixel readout

- Improve the memory consumption of the voxel map
  (std::map has 3 64bit pointers per entry for internal management)
- Implement TPCPix end plate and digitisation
  - 64-bit pixel index
  - MicroMegas-like gas gain fluctuations
  - Pixel crosstalk

Summary

Simulation

- Mokka driver in `lowPt` mode can be used for very small steps
- Available since Mokka tag 07-07-p03

Digitisation

- TPCGEM digitisation is fully compatible with Mokka
- Improved memory footprint
- ADC with double-threshold zero suppression

More details in LCD-Note-2011-025.

Outlook

- Extension for multiple modules almost ready
- Digitisation for Ingrid-like pixel readout