

# Different Implementation of the LumiCal and BeamCal Clustering Algorithms

André Sailer

CERN-PH-LCD

FCal Clustering Meeting  
July 14, 2014

# Table of Contents



- 1 Introduction
- 2 Requirements
- 3 BeamCal Clustering Implementation
- 4 LumiCal Clustering Implementation
- 5 Conclusions

## BeamCal

- There are a few *implementation* for the BeamCal clustering algorithm
- The algorithm is always very similar same (subtraction of average background deposits, and finding of significant energy excess)
- Do we need many implementations of the BeamCal algorithm? We could save effort by just using one
- Maybe we can agree on the required functionality, and develop and use a single implementation
- When there is a new *algorithm* it should become part of the joined implementation

## LumiCal

- There is an existing implementation, and it should move to ILCSoft (needs some cleanup)

- For realistic estimates (especially in the BeamCal) backgrounds have to be included
- Full-simulation background-files can be quite large and it takes a significant amount of disk-space, memory, and time to read the full simulation files
  - ▶ At CLIC, where more than 1 bunch crossings is needed for each events this is even more of a problem
- One can instead
  - ▶ Extract (summarize) all the hits from the files and only store those
    - ★ Loses all MC information about background particles, but those are not really needed
  - ▶ Use only average energy deposits and standard deviation

## Requirements:

- The implementation should be able to handle either full simulation, summary, or averages
- There should be a processor/program which create summary or average files from simulation of background

# Requirements II: Different Detectors



- The implementations should work for the different full detector simulations: Mokka, SLIC, and DD4hep-based; and different BeamCal layouts in them
- Also for the standalone simulation: BeCaS
- Handle backgrounds:
  - ▶ From full simulation
  - ▶ via average and fluctuation files
- Should be integrated into the reconstruction frameworks, and enable LCIO for input/output
- Knowledge of the average background is always necessary, so it should be easy to create those files based on simulation of the backgrounds.

# Existing Implementations (that I know of)



- Standalone Clustering used for BeamCal, Marlin Processor used for ILD DBD
- Santa Cruz (I assume)
- Library written by me to estimate tagging efficiency for CLIC 3 TeV, made a bit more general

## Stand-alone library with thin wrappers for Marlin

### ■ Marlin Processors

- ▶ C++, object oriented implementation, presented at last Cracow FCal meeting
- ▶ Geometry read from GEAR file (as given by Mokka or written by hand (10 lines of XML))
- ▶ LCIO input for signal
- ▶ Individual bunch crossings of pairs are given as ROOT files containing only energy per pad (index used as cell ID)
- ▶ LCIO Output
  - ★ There is an additional processor to create the background files from full simulation

### ■ Stand-alone executable (ReconstructBecas)

- ▶ Reads BeCaS ROOT files

Placed in FCAL svn repository:

<https://svnsrv.desy.de/viewvc/FCAL/Software/FCalClusterer/>

- Written by Iftach Sadeh (Tel-Aviv University) in 2008, and documented in his Master Thesis:  
<http://alzt.tau.ac.il/~sadeh/LumiCalClusterer.html>
- I tried running it out of the box on CLIC\_ILD\_CDR events, which failed
- Took a bit of time to make it work (succeeded last Friday)
- The encoding of the hit position in the LumiCal had changed in the meantime
- Still missing:
  - ▶ Reading geometry information
  - ▶ LCIO output
  - ▶ Move all control variables to steering file
  - ▶ Tuning of parameters, fixing bugs



- A few implementations for BeamCal clustering: Can we agree to use one of them or merge them?
  - Can also have different algorithms in the same implementation
- Improve the LumiCal Clustering Processor, and make it available via ILCSoft

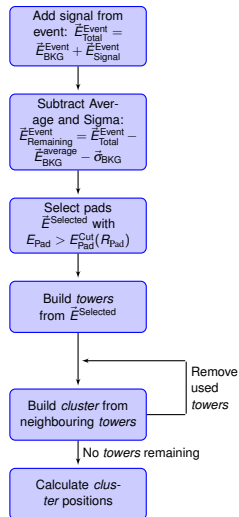
# Backup Slides

# BeamCal Electron Clustering Algorithm



## General Idea of Clustering

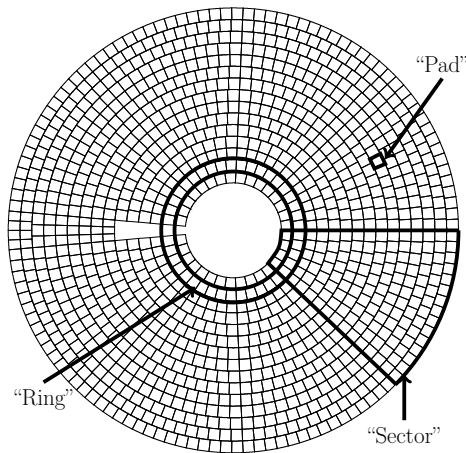
- 1 Energy in the BeamCal Pads is
$$\vec{E}_{\text{Total}}^{\text{Event}} = \vec{E}_{\text{Signal}}^{\text{Event}} + \vec{E}_{\text{BKG}}^{\text{Event}}$$
- 2 Subtract the average background plus one standard deviation for each cell
$$\vec{E}_{\text{Remaining}}^{\text{Event}} = \vec{E}_{\text{Total}}^{\text{Event}} - (\vec{E}_{\text{BKG}}^{\text{average}} + \vec{\sigma}_{\text{BKG}})$$
- 3 Select the pads with sufficiently large remaining energy, e.g.,  $E_{\text{Remaining}}^i > \sigma_{\text{BKG}}^i$
- 4 Find *towers* in the selected pads: Pads with the same position in different layers
- 5 Merge neighbouring *towers* into *cluster*
- 6 Calculate *cluster* position



# BeamCal Detector at CLIC



- 10 mrad to 45 mrad (ILC about 3.5 mrad)
- Absorber for incoherent pairs
- Mask for downstream elements (QD0, BPM)
- Radiation hard sensors required
- Electron tagging for background suppression
- Tungsten sandwich calorimeter, Molière radius of about 1 cm
- Pad size  $8 \times 8 \text{ mm}^2$
- ILC opening around outgoing beam pipe larger, different number of pads per sector



```
<gear>
<global detectorName="CLIC_ILD_CDR" />
<detectors>
<detector name="BeamCal" geartype="CalorimeterParameters">
  <layout type="Endcap" symmetry="2" phi0="0.000000000e+00" />
  <dimensions inner_r="3.201000000e+01" outer_r="1.500000000e+02"
    inner_z="2.880699600e+03" />
  <layer repeat="40" thickness="4.000800000e+00" absorberThickness="3.50000e+00"
    cellSize0="7.866000000e+00" cellSize1="1.000000000e+00" />
  <parameter name="beam_crossing_angle" type="double" value="2.000000000e+01" />
  <parameter name="cylinder_spanning_phi" type="double" value="6.108652382e+00" />
  <parameter name="cylinder_starting_phi" type="double" value="3.228859116e+00" />
  <parameter name="dead_area_outer_r" type="double" value="6.347400000e+01" />
  <parameter name="pairsMonitorZ" type="double" value="2.880849800e+03" />
  <parameter name="phi_segmentation" type="DoubleVec" value="0.190895 0.152716
    0.127264 0.127264 0.109083 0.0954477 0.0848424 0.0848424 0.0763582
    0.0694165 0.0636318 0.0636318 0.058737 0.0545415 0.0509054" />
  <parameter name="nPhi_segmentation" type="IntVec" value="4 5 6 6 7 8 9 9 10
    11 12 12 13 14 15" />
</detector>
</detectors>
</gear>
```

- Used to reconstruct clusters from BeCaS
- Multi dimensional arrays storing: energy, position, cell IDs, angles, ...
- No LCIO in or output
- Geometry information hard-coded, individual pad IDs (layer, r, phi) stored for all cells along with energy deposit and cell geometries.
- Collection of many small programs

- Copy of the standalone clustering to Marlin
- Signal LCIO input, background from a ROOT file containing averages/fluctuations (same file as for the standalone?)
- LCIO Output
- Does not work for different BeamCal geometries, too many hardcoded numbers