Digitization Simulation using DigiSim

Guilherme Lima

for the ILC-software group at NIU

NORTHERN ILLINOIS
UNIVERSITY

ILC Simulations Workshop

Boulder, January 09-11, 2006

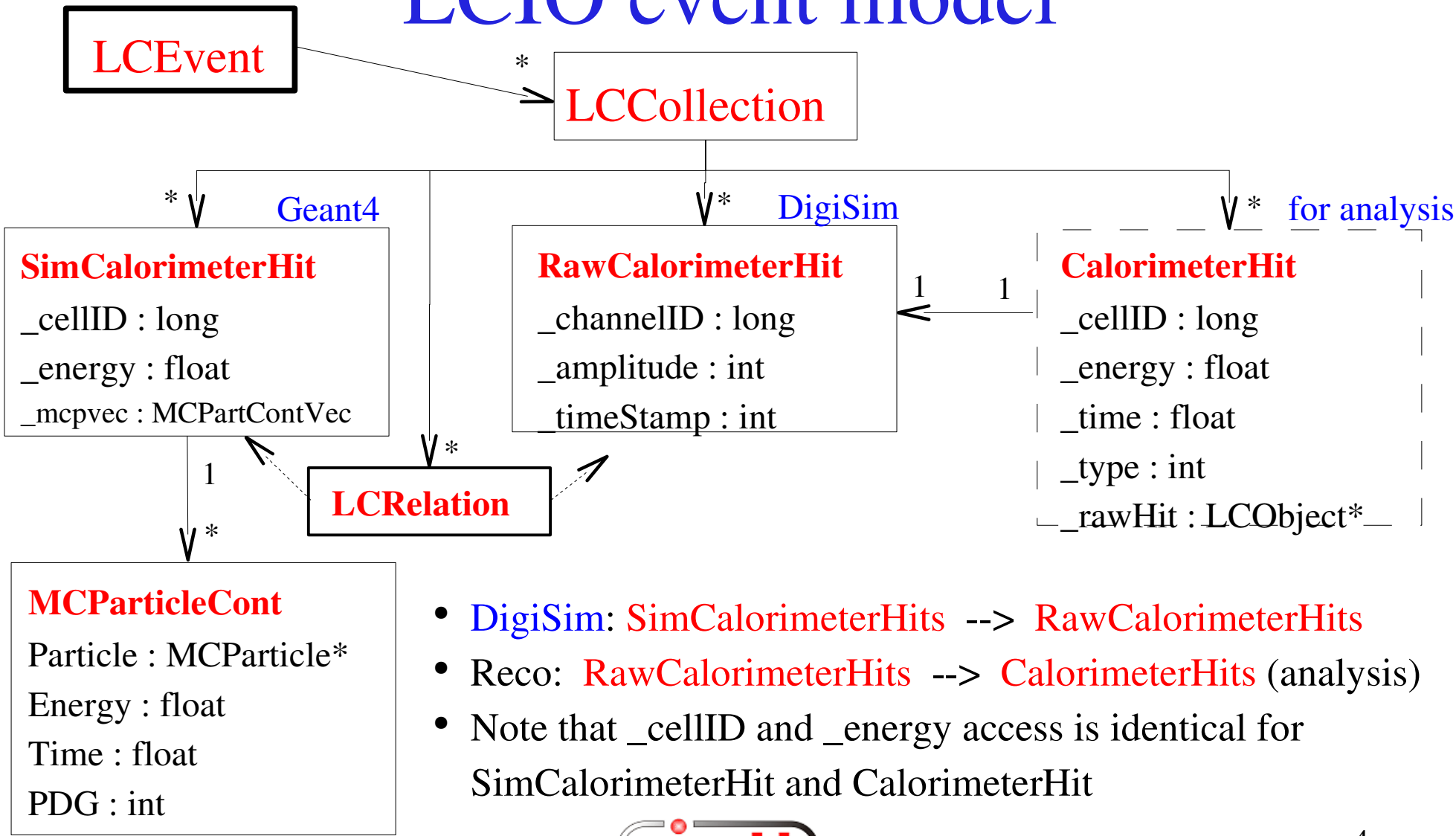
Talk Outline

- DigiSim: purpose and description
- Configuration
- An example: effects on hit energy distributions
- Current status

DigiSim

- Goal: a program to parametrically simulate the signal propagation and digitization processes for the ILC detector simulation
  an essential tool for comparing different detector technologies
- DigiSim role is to convert the simulated data (energy depositions and hit timings) into the same format AND *as close as possible* to real data from readout channels, while preserving all MC information from input data files
  - *As close as possible* means that all known effects from digitization process should be taken into account, if possible: cell ganging, inefficiencies, noise, crosstalks, hot and dead channels, non-linearities, attenuation, etc.
- Same reco / analysis software can be used for MC and real data
- DigiSim produces RawHits and (Digi)CalorimeterHits from SimCalorimeterHits

LCIO event model

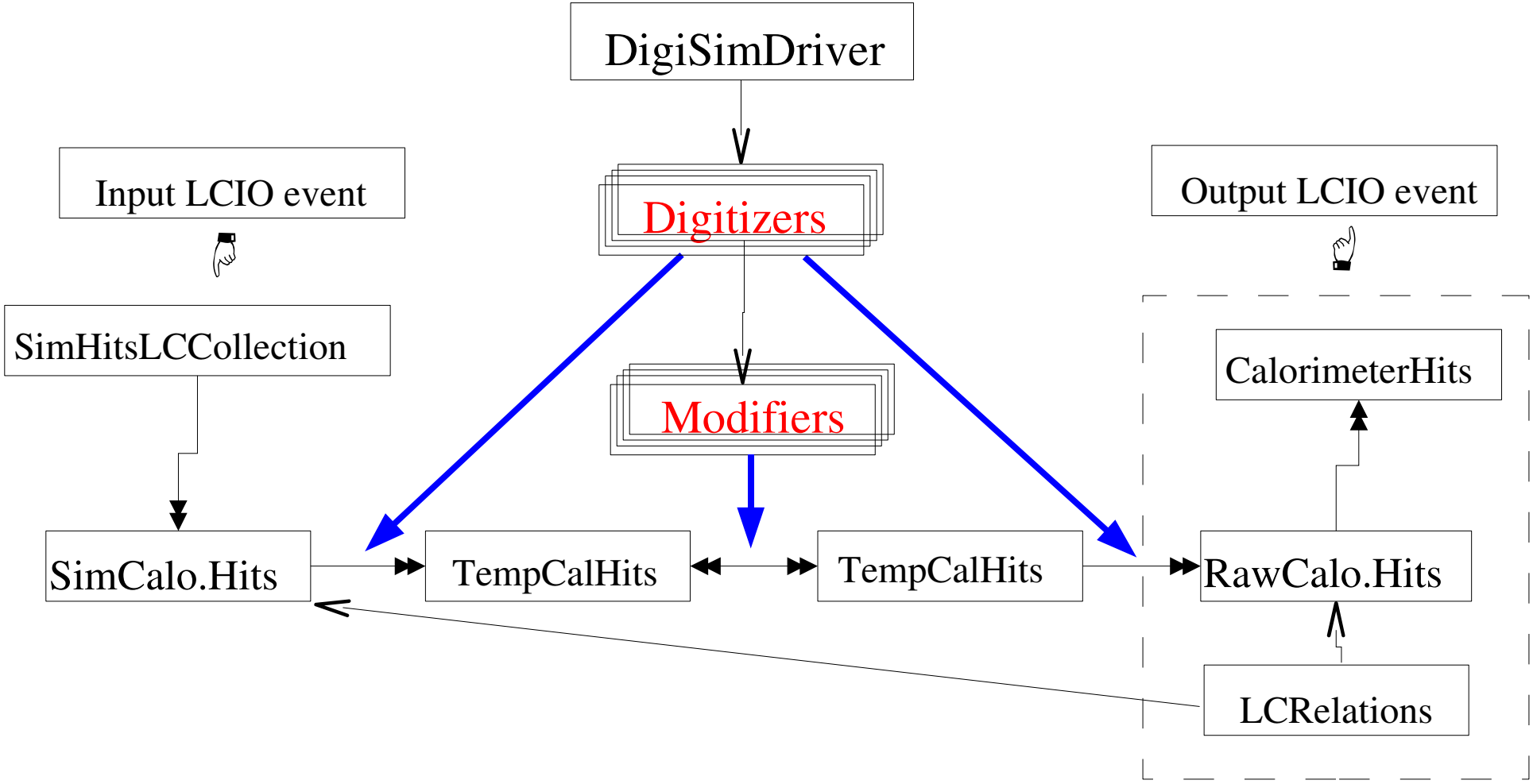


- DigiSim: SimCalorimeterHits --> RawCalorimeterHits
- Reco: RawCalorimeterHits --> CalorimeterHits (analysis)
- Note that _cellID and _energy access is identical for SimCalorimeterHit and CalorimeterHit

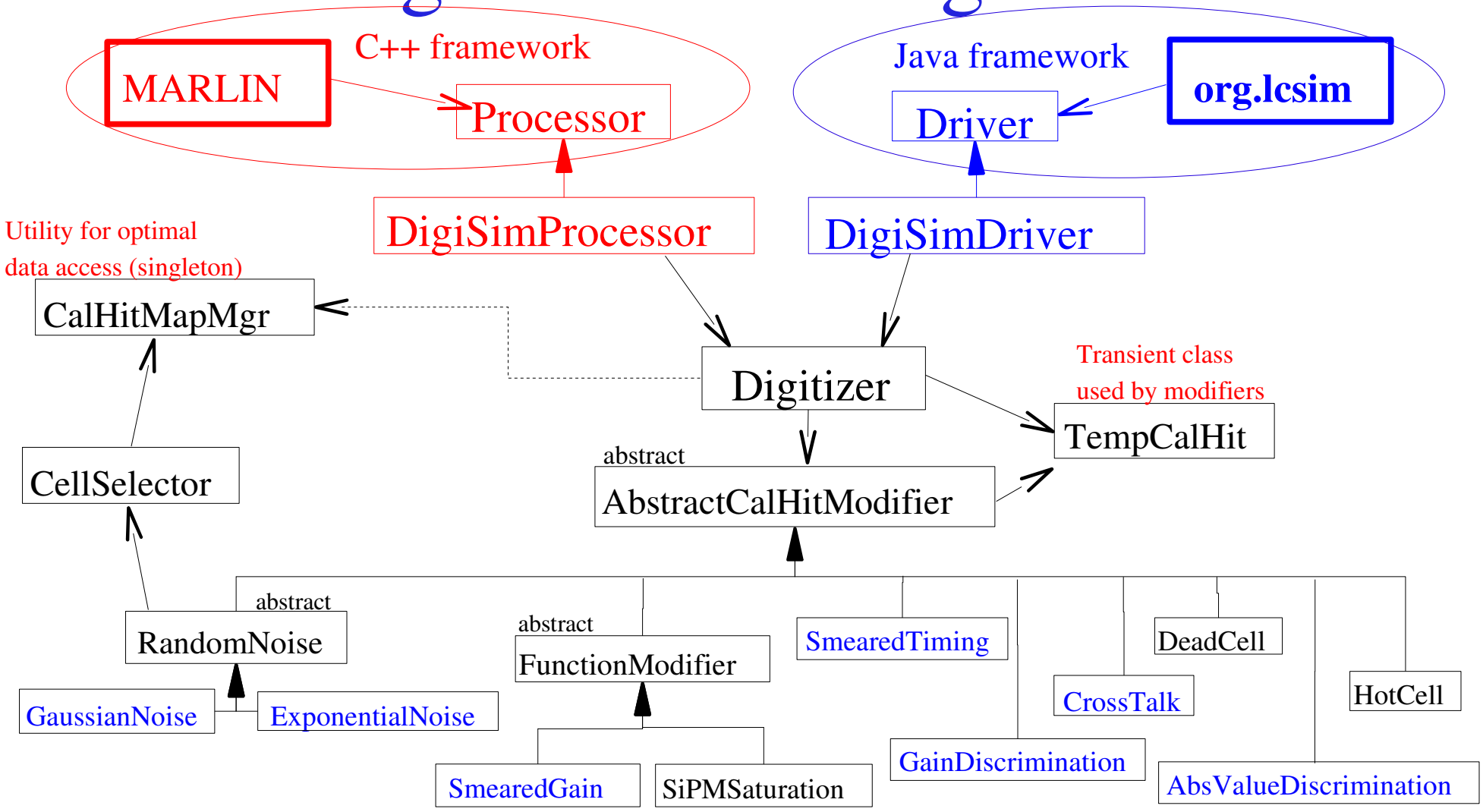
SimCalorimeterHits or CalorimeterHits?

- Consider moving your reconstruction algorithms to use CalorimeterHits instead of SimCalorimeterHits
- How to do this:
  - All non-MC calls to SimCalorimeterHits (energy, position, time) can be transparently replaced with equivalent calls to CalorimeterHit objects.
    For MC-related methods, use (same) cellid as a key to access SimCalorimeterHits.
  - Detector name is used to select the correct DigiSim configuration file.
  - Configuration files exist for most Snowmass detectors
    - All RPC-based have identity configurations
    - All scintillator-based: SDJan03, sidaug05_scint, cdcaug05* have non-identity configurations (see later)
  - Identity DigiSim config files are available for helping people to get started with DigiSim output

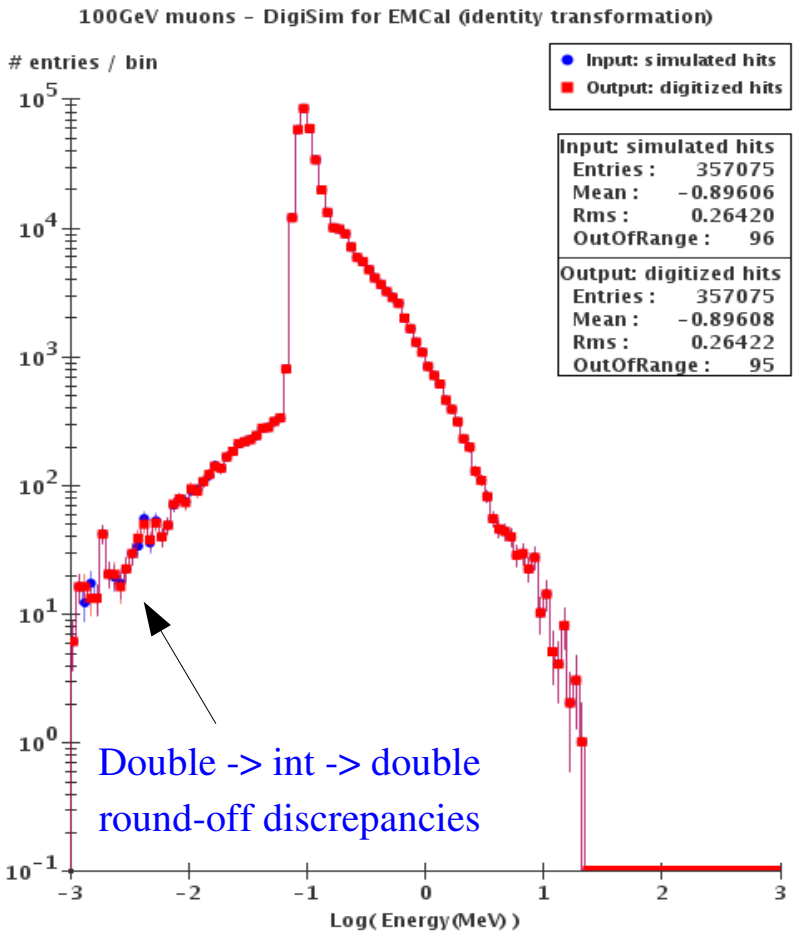
DigiSim event loop

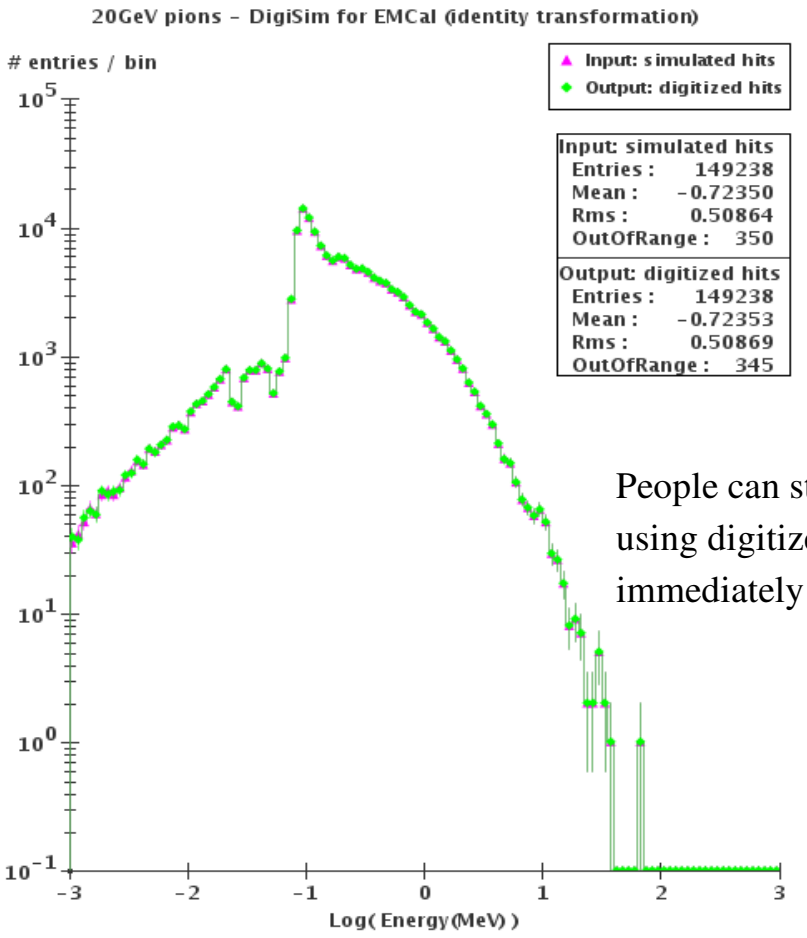


DigiSim class diagrams



An identity transformation





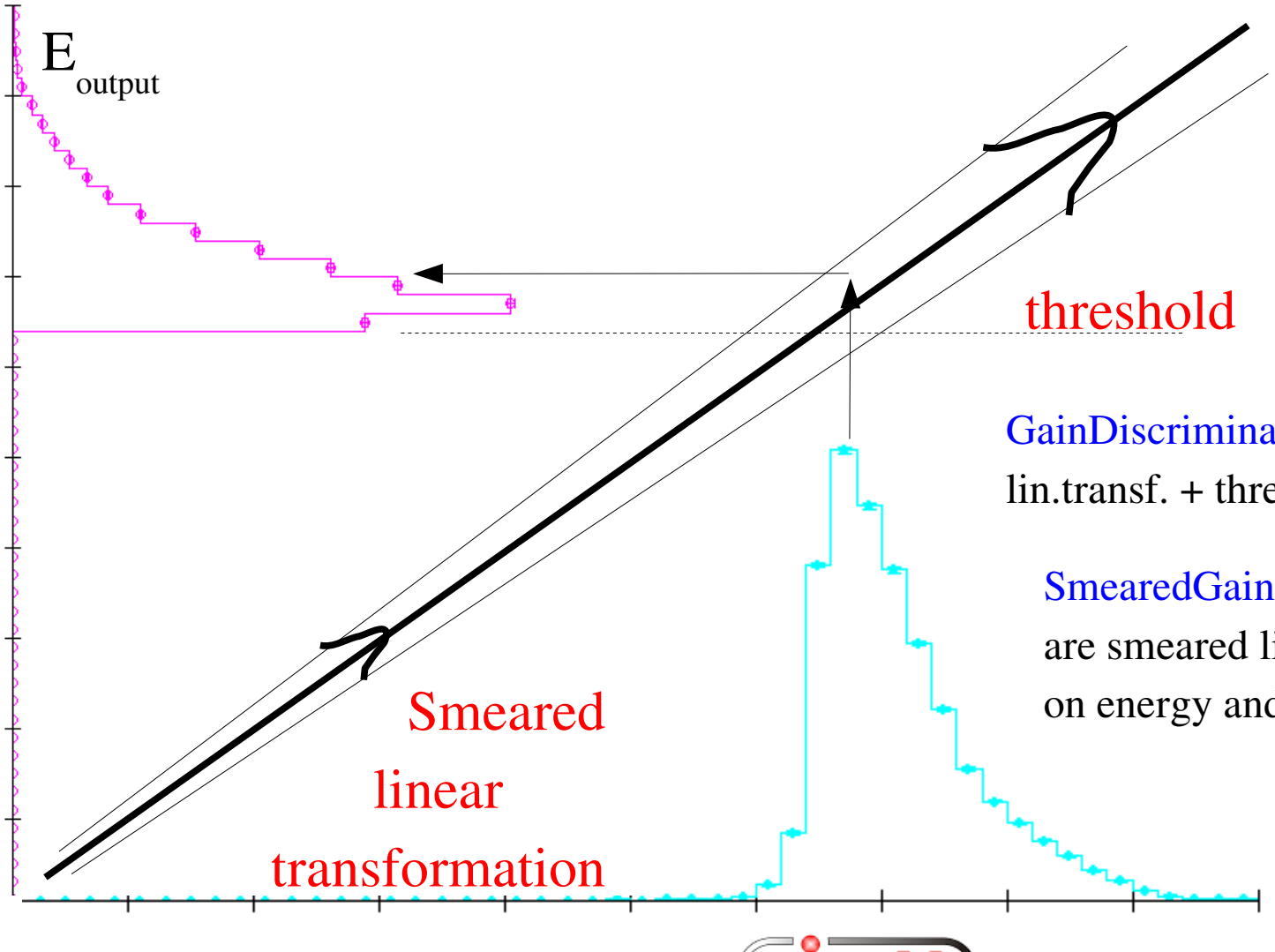
People can start using digitized hits immediately!!

Setting up DigiSim modifiers

- Modifiers' role is to *tweak* hit energy/timing
- Important: It helps to interpret the "energy" field according to the process to be modeled:

  Energy (GeV) --> (light yield) --> # produced photons --> (photon collection) --> # photons collected --> (Quantum effic) --> # photoelectrons --> (Gain) --> uAmp --> (signal integration) --> charge collected --> (Digitization) --> ADC counts
- DigiSim modifiers are just factors (or more generally, functions) which represent each step along the digitization process

A common transformation



GainDiscrimination is a smeared lin.transf. + threshold on energy

SmearedGain and SmearedTiming are smeared linear transformations on energy and time respectively

Setting up a DigiSim configuration file

```
##############################################
#  Example steering file for DigiSim
##############################################
.begin Global  ----------------------------------------
# specify one or more input files (in one or more lines)
LCIOInputFiles inputfile

# the active processors that are called in the given order
ActiveProcessors CalHitMapProcessor
ActiveProcessors EcalBarrelDigitizer
ActiveProcessors EcalEndcapDigitizer
ActiveProcessors HcalBarrelDigitizer
ActiveProcessors HcalEndcapDigitizer
ActiveProcessors OutputProcessor

# limit the number of processed records (run+evt):
MaxRecordNumber 500
.end Global  ------------------------------------------
#############################################################
.begin EcalBarrDigitizer
ProcessorType DigiSimProcessor

InputCollection        EcalBarrHits
OutputCollection       EcalBarrRawHits
Raw2SimLinksCollection EcalBarrRaw2sim

ModifierNames    EMBDigiIdentity
# modifierName    Type                  Parameters (floats)
EMBDigiIdentity    SmearedGain          100000000       0
.end ---------------------------------------------
```

One digitizer per subdetector

"Identity" factor 10^8 avoids precision loss in the conversion double -> int -> double

Configuring processors and modifiers

```
##########################################################
# A subdetector digitizer.  It instantiates one or more calorimeter hit
#  "modifiers", which together represent the full digitization process
.begin HcalBarrDigitizer

ProcessorType DigiSimProcessor

InputCollection  HcalBarrHits
OutputCollection HcalBarrRawHits
Raw2SimLinksCollection HcalBarrRaw2sim

ModifierNames HBlightYield HBcrosstalk HBlightCollEff HBPDQuEffic HBExpoNoise  HBGaussNoise
HBdiscrim HBGain

# Parameters:
# modifierName    Type                 gainNom  gainSig  thresh   thrSig
HBlightYield     GainDiscrimination  10000000        0       1        0

# Crosstalk                             mean    sigma
HBcrosstalk      Crosstalk               0.020     0.005

# Smeared gain parameters:              gain   gainSigma  thresh   thrSig
HBlightCollEff   GainDiscrimination  0.0111     0.0029       1        0
HBPDQuEffic      GainDiscrimination    0.15          0       1        0

(...Truncated... see file cdcaug05_np.steer in DigiSim area)
.end ------------------------------------------------------------
```

(As many as needed)

Simple example: configuration for the tile HCal

- Scintillation: 100 eV / photon , or 10^{+4} photons/MeV

  Ex: a MIP at normal incidence on 0.5cm-thick scintillator deposits ~ 0.9MeV, or 9000 photons

  ==> use GainDiscrimination modifier with 10^{+7} photons/GeV and a threshold at 1 photon

```
#modifierName   type                  gainNom  gainSig  thresh   thrSig
HBlightYield   GainDiscrimination  10000000        0       1        0
```

- Light crosstalk:

  first neighbors: 1.5% to 2% --> (2.0 +/- 0.5) %

```
HBcrosstalk      Crosstalk                0.020     0.005
```

- Photon collection efficiency (QE~15%):

  9000 scint.photons/MIP --> 15 PE/MIP detected (cosmics measurements at NICADD)

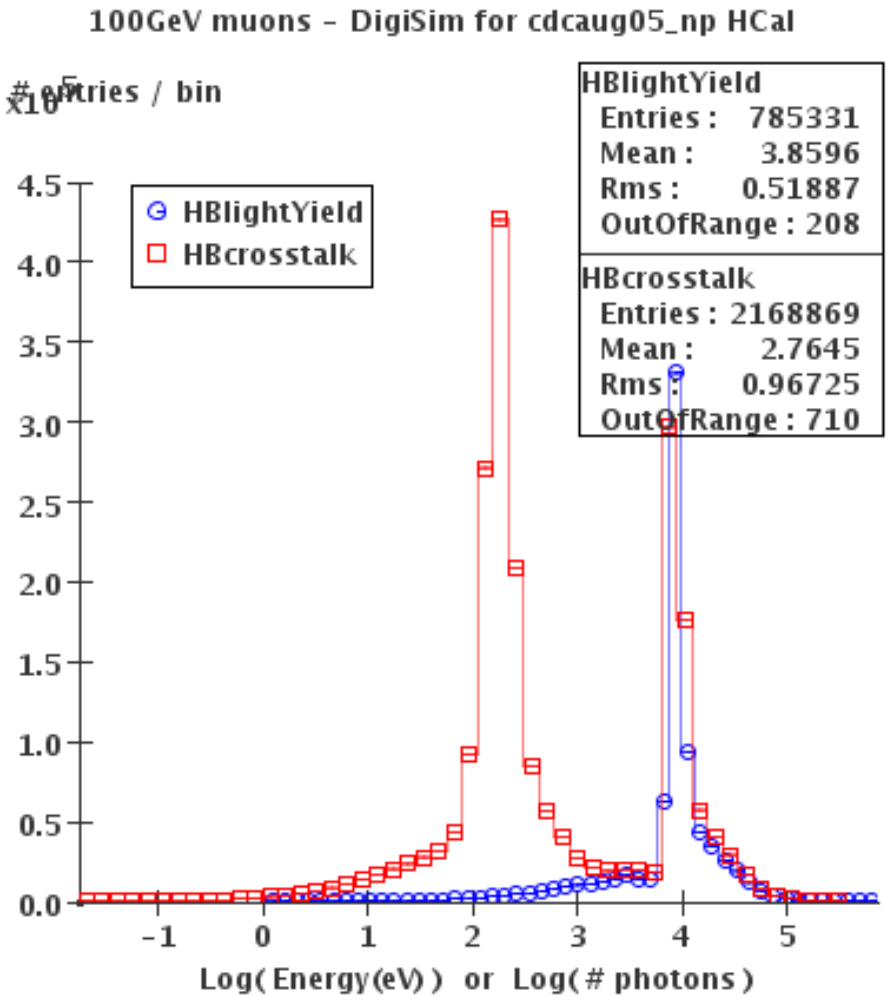
  15 / 0.15 = 100 incident photons => Eff_{coll} = 100 inc. / 9000 tot.scint. = 0.0111

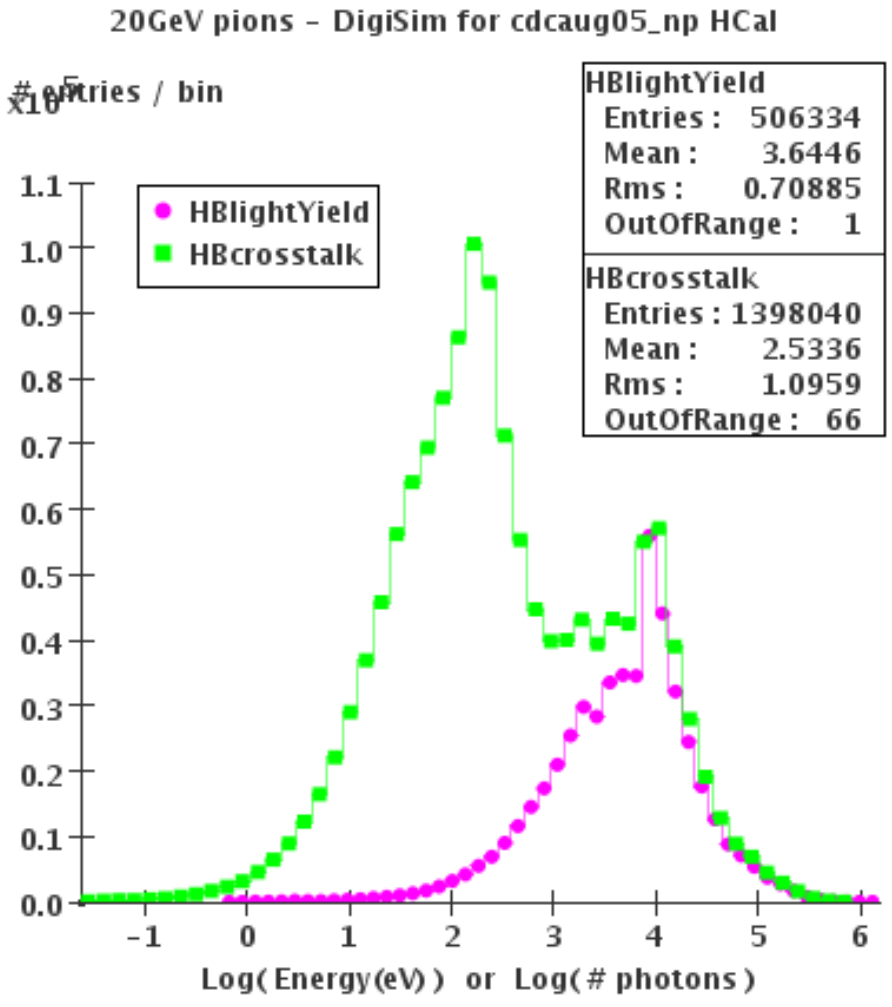
  Variance (Poisson): $sigma_N^2 = \langle N \rangle \rightarrow$ for $\langle N_{PE} \rangle = 15$, $(sigma_N / N) \sim 26\%$

  Therefore: $Eff_{coll} = 0.0111 \pm 0.0029$ => use GainDiscrimination with smearing

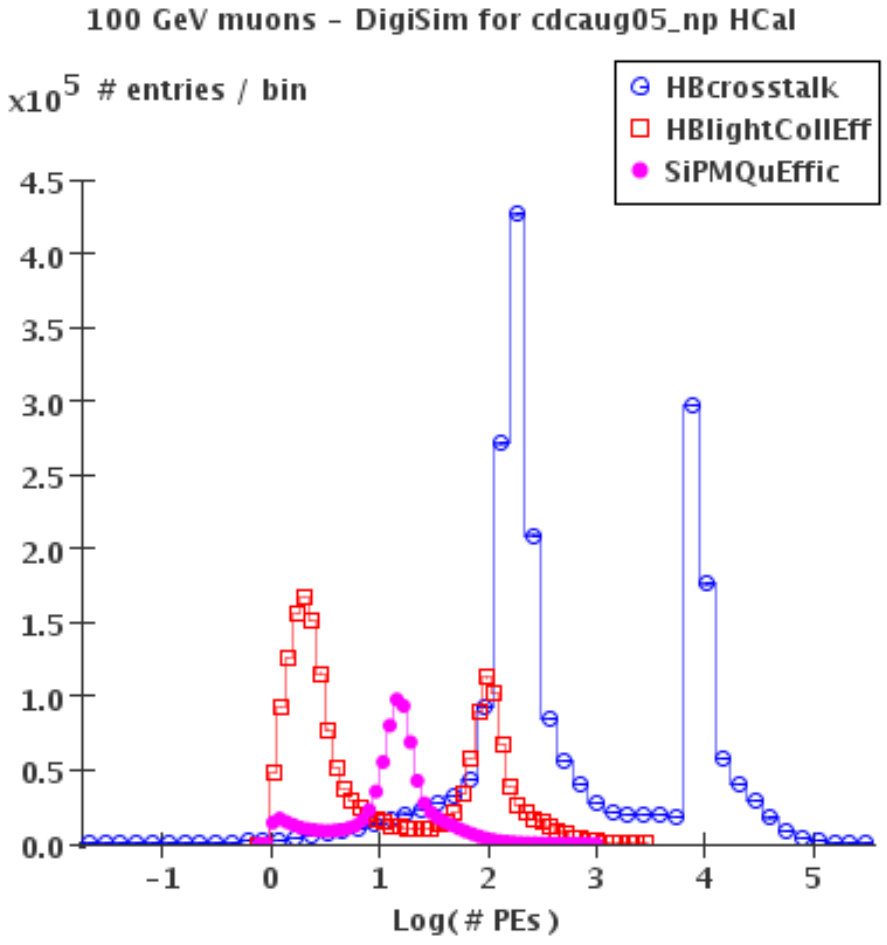
```
HBlightCollEff GainDiscrimination    0.0111   0.0029       1        0
```

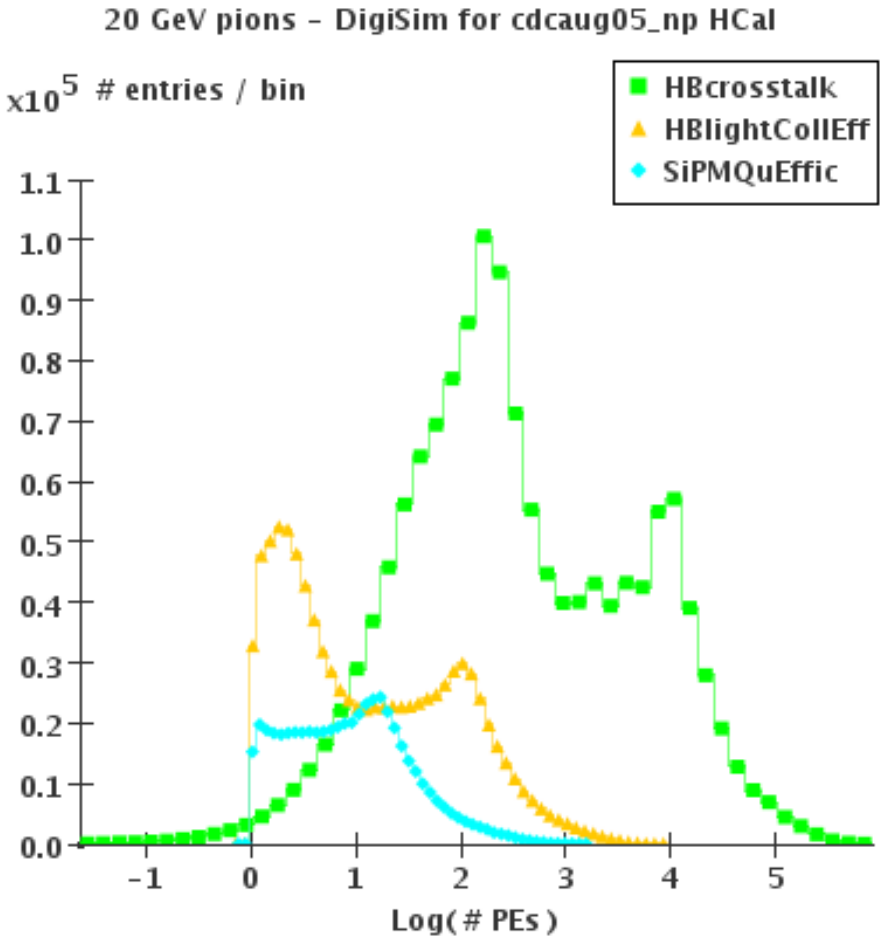
Light cross-talk





Photodetection





Simple example: parameters from the tile HCal

- Photosensor detection efficiency: QE ~ 15 %

```
HBPDQuEffic      GainDiscrimination    0.15          0       1        0
```

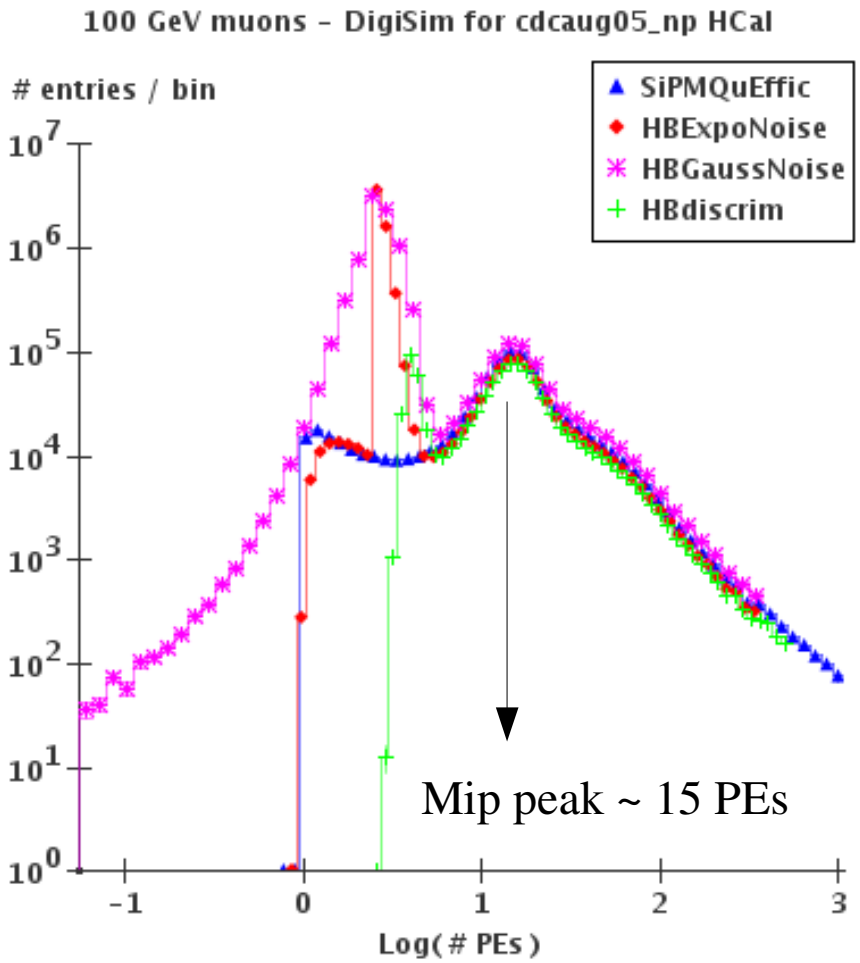
- Noise simulation:
  - Photosensor noise: exponential distribution (guess: mean 0.6)
  - Electronics noise: gaussian distribution (guess: mean 0, sigma 1.6, keep +/- tails)

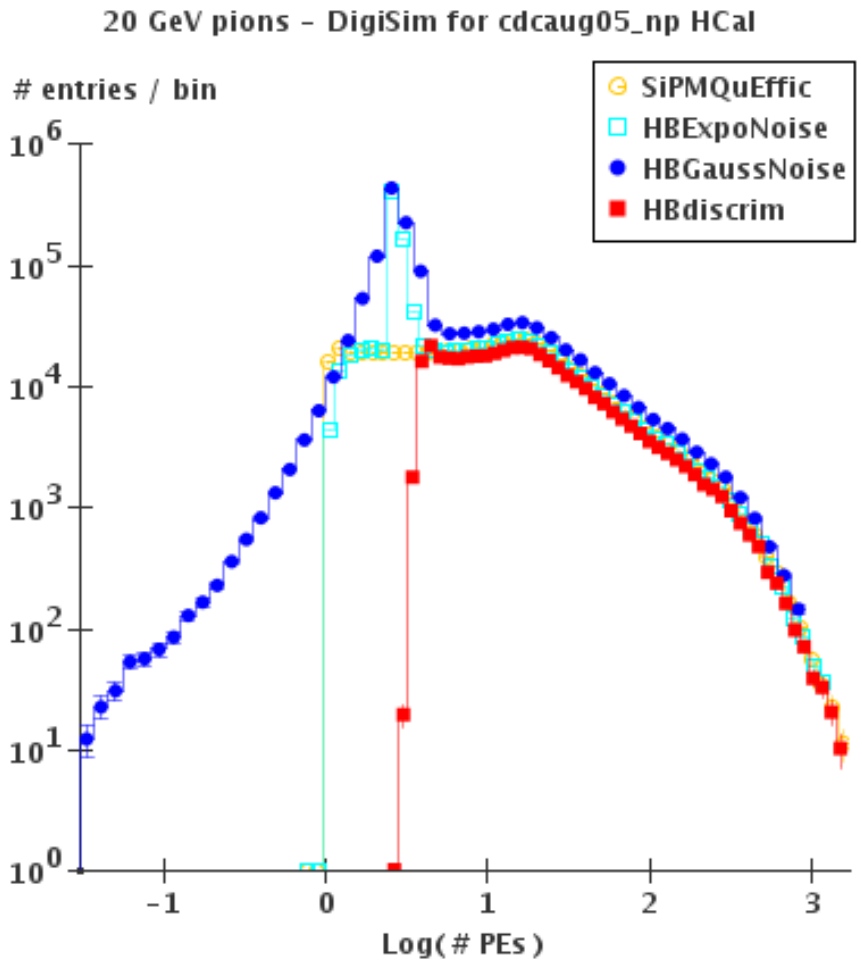
```
# GaussNoise parameters:        sys   be    Ecut   TimeNom  TSig   Mean  Sigma
# Note: sigma<0 means that threshold acts on absolute value only
HBGaussNoise   GaussianNoise      3     0    2.5       100   100    0.0  -0.58

# ExponentialNoise parameters:  sys    be   Ecut   TimeNom  TSig   Mean
HBExpoNoise    ExponentialNoise   3     0    2.5       100   100   0.23
```

- Discrimination: ¼ MIP cut ~ 4 PEs: threshold at 4 +/- 0.25 (on abs value)

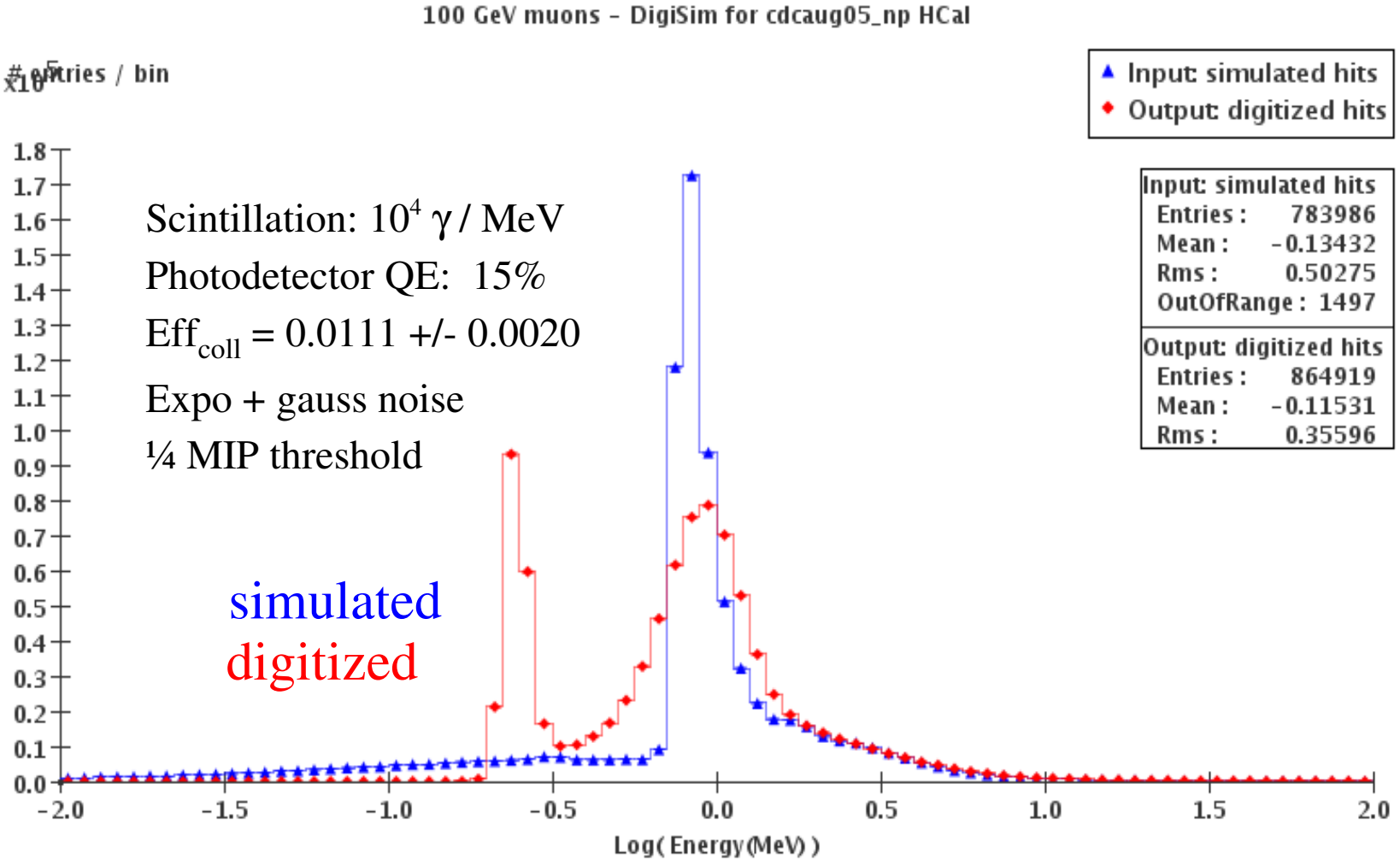
```
# Discrimination                                threshold      sigma
HBdiscrim    AbsValueDiscrimination                   4       0.25
###HBdiscrim GainDiscrimination       1       0       4       0.25
```

Noise and discrimination

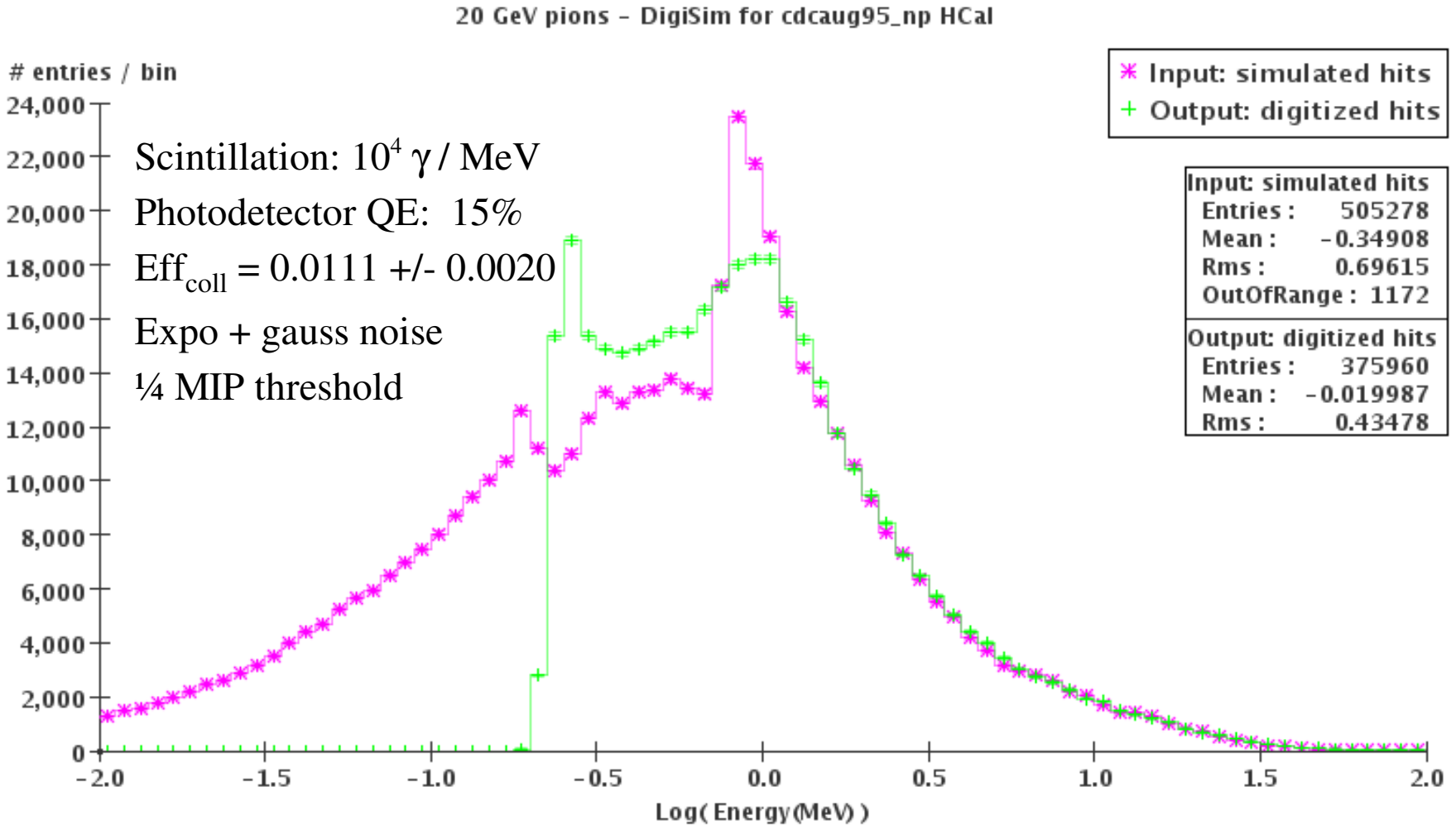




HCal scintillator digitization (preliminary)



HCal scintillator digitization (preliminary)



DigiSim Status

- A digitization simulation package, DigiSim, has been developed at NICADD/NIU
  - Java version released is full featured. Same configuration file as C++ (Marlin steering file)
  - C++ version partly available. Same basic structure, missing are crosstalks and noise modeling, as they depend on geometry-aware classes (CGA?)
  - Several generic modifiers are available (smeared linear transforms, crosstalk, noise, discrimination, etc. (Note: crosstalk and noise modifiers are not available in C++ version)
  - LCRelations implemented to associate raw hits to one or more corresponding simulated hits
- DigiSim can be run in either a stand-alone mode to produce persistent lcio output, or as an on-the-fly preprocessor to reconstruction/analysis.
  In the former case, raw/digi hits and LCRelations are saved into the output LCIO files, in addition to all the (untouched) MC information present at DigiSim input.
- A test version of a digitizer for a tile HCal barrel currently exists
  It can be used as an example to implement other subdetectors, like endcaps, ECal, RPCs, GEMs, tracker detectors. See example presented in this talk.

DigiSim status (cont.)

Other people are encouraged to add DigiSim configurators for RPCs, GEMs, trackers, and to make sure their algorithms can easily make the switch to use digitized data.

- Both C++ and Java versions are available through official CVS servers
  C++: released in the Calice CVS repository
  Java: released in the LCSim CVS repository. Download instructions from the confluence pages
- Documentation available from http://nicadd.niu.edu/digisim, including downloading and building instructions
- Send any questions or comments to: lima at nicadd.niu.edu