DigiSim, a digitization framework for the Calice test beam

Guilherme Lima

Calice Workshop @ Montreal

May 10-12, 2006

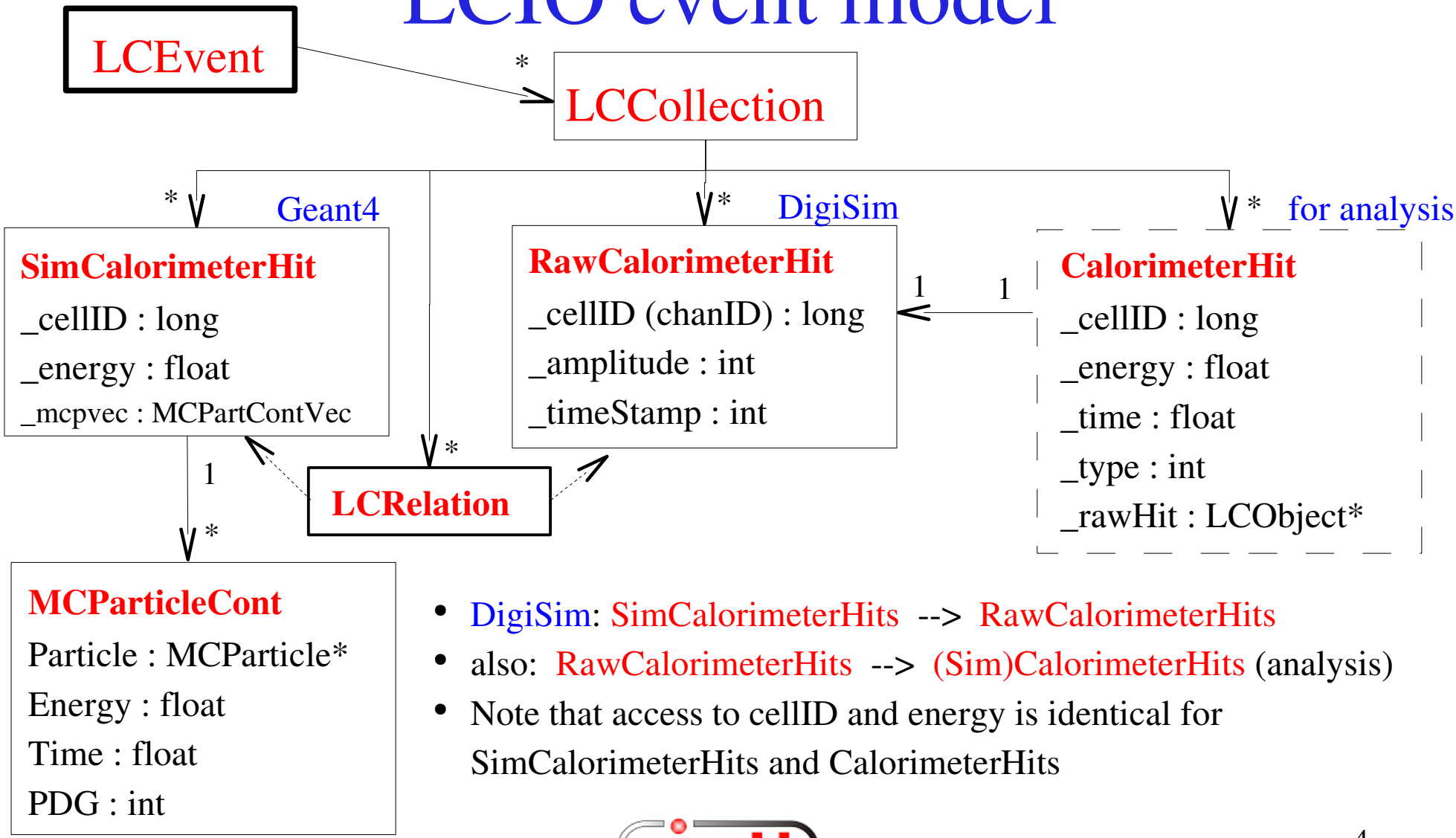
DigiSim: talk outline

- Purpose
- Digitization scheme envisioned in DigiSim
- Package description and existing features
- An example: some effects on typical hit energy distributions
- Configuring DigiSim
- Current status

Purpose of DigiSim

- Goal: a program to parametrically simulate the signal propagation and digitization processes for the ILC detector simulation
  an important 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 reconstruction & analysis software can be used for MC and real data
- DigiSim produces RawCalorimeterHits and/or calibrated (Sim)CalorimeterHits from (ideal) SimCalorimeterHits generated by Geant4.

LCIO event model



- DigiSim: SimCalorimeterHits --> RawCalorimeterHits
- also: RawCalorimeterHits --> (Sim)CalorimeterHits (analysis)
- Note that access to cellID and energy is identical for SimCalorimeterHits and CalorimeterHits

Possible DigiSim schemes

Average behavior

a) Average detector effects (crosstalk, sensor noise, signal collection, etc.)

b) Average electronics noise and xtalk

c) Final normalizations and thresholds

d) Digitization --> RawCalHits

e) Uniform calibration (MIPs to 1) raw(ints) --> calibrated hits(floats)

f) CalorimeterHits saved, to be compared with real data

Individual channel behavior

a) Non-uniform detector effects (crosstalk, sensor noise, etc.)

b) Non-uniform normalization (from MIPs to ADC counts): noise and pedestals from a DB (ADC counts)

c) Final thresholds (avg or chan-chan?)

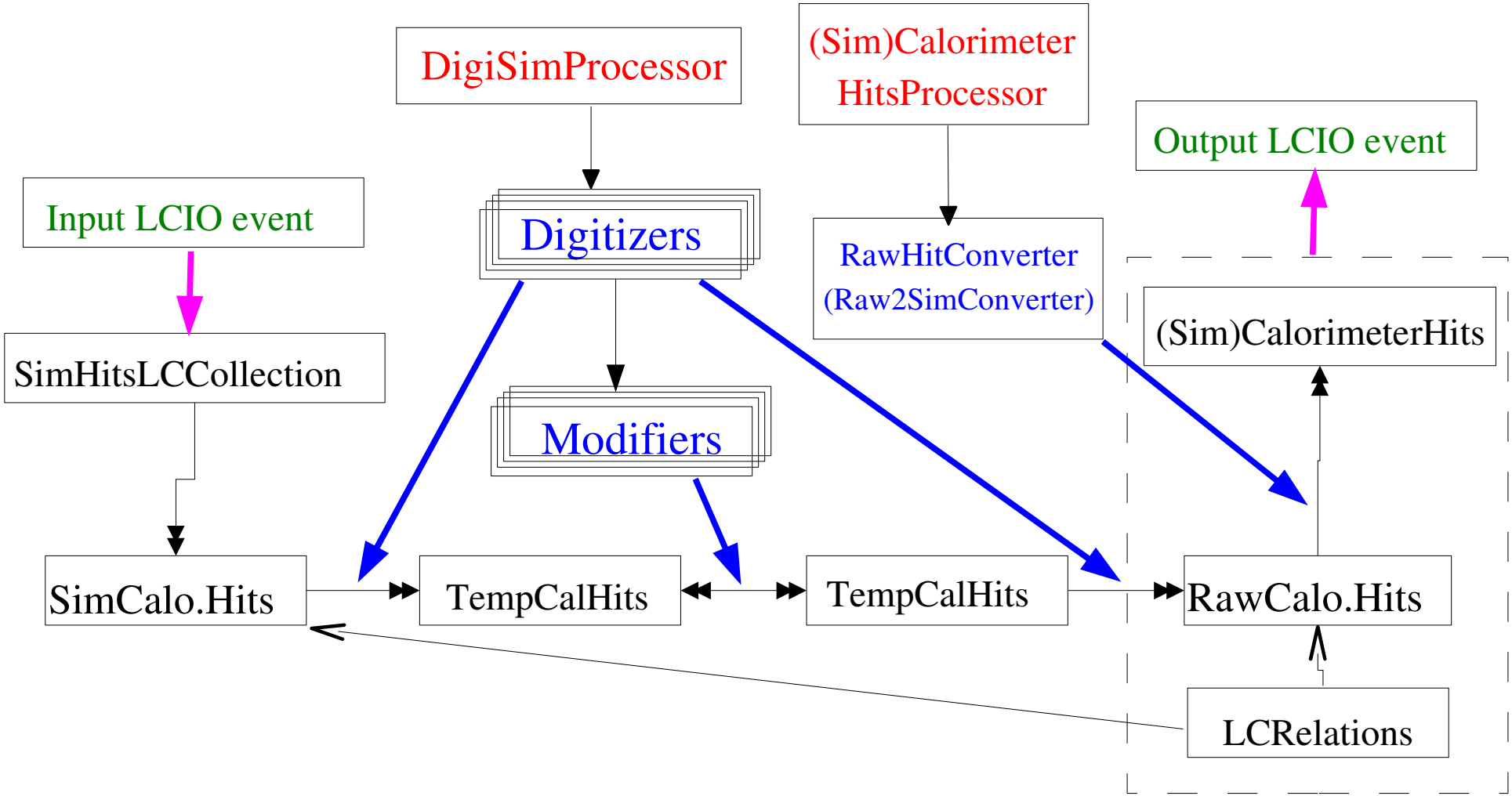
d) Same non-uniform calibration for data and MC (MIPS to 1)

e) CalorimeterHits saved, MC vs. data comparison (channel by channel)

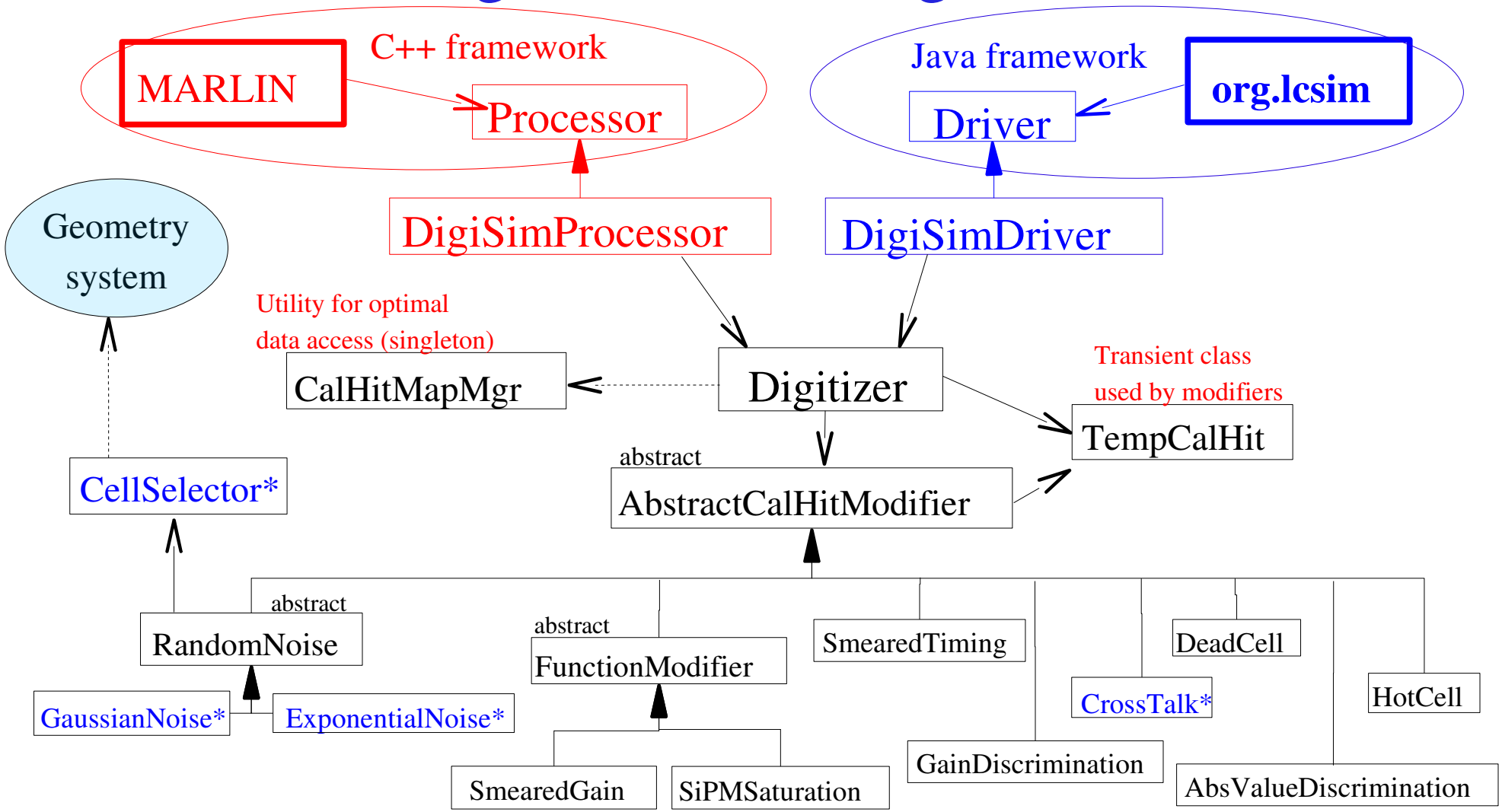
SimCalorimeterHits or CalorimeterHits?

- DigiSim can produce either object as output from RawCalorimeterHits (recent feature)
- Efficiency / purity studies easier with SimCalorimeterHits
- All non-MC calls to SimCalorimeterHits (energy, position, time) can be transparently replaced with equivalent calls to CalorimeterHit objects. All variables should be defined as CalorimeterHits rather than SimCalorimeterHits.
- For MC-related methods, use (same) cell ID as a key to access the SimCalorimeterHits. If underlying object is a SimCalHit, a dynamic cast can be used to access MC info.
- There are identity* transformations available, and also simple energy and/or time threshold configurations, to simplify the transition to use of DigiSim.
  *Note: Identity includes a 10*6 to 10*8 factor to allow float-->int conversions with no data loss above thresholds.

DigiSim event loop

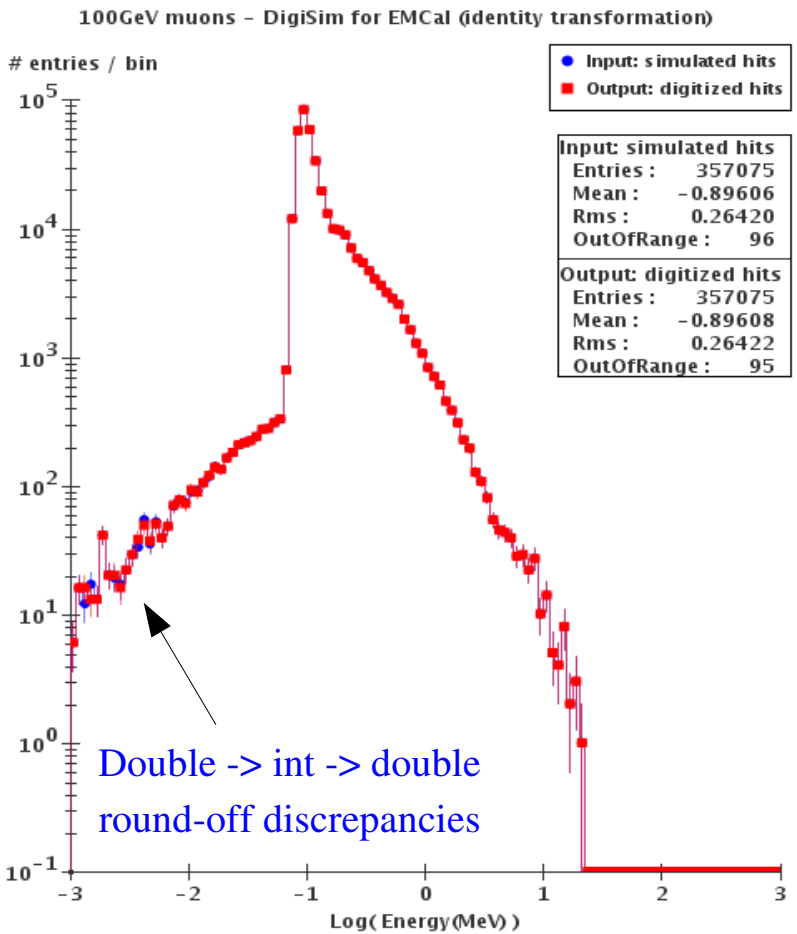


DigiSim class diagramS

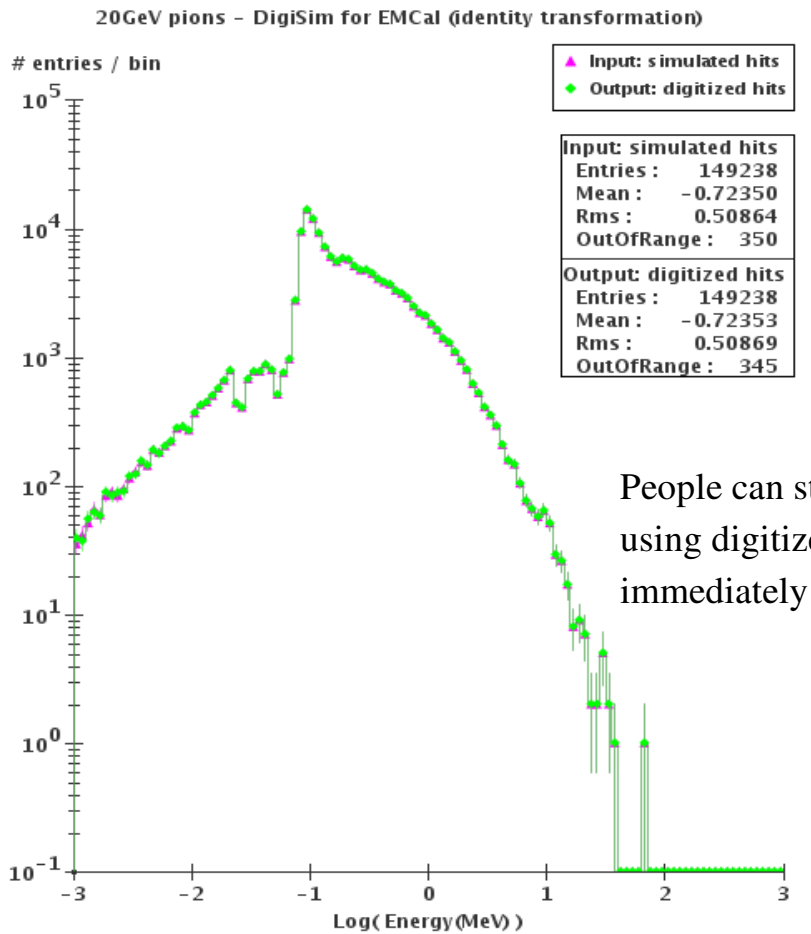


* Implementation depends on some desirable features, still not available in Gear.

An identity transformation



Double -> int -> double round-off discrepancies



People can start using digitized hits immediately!!

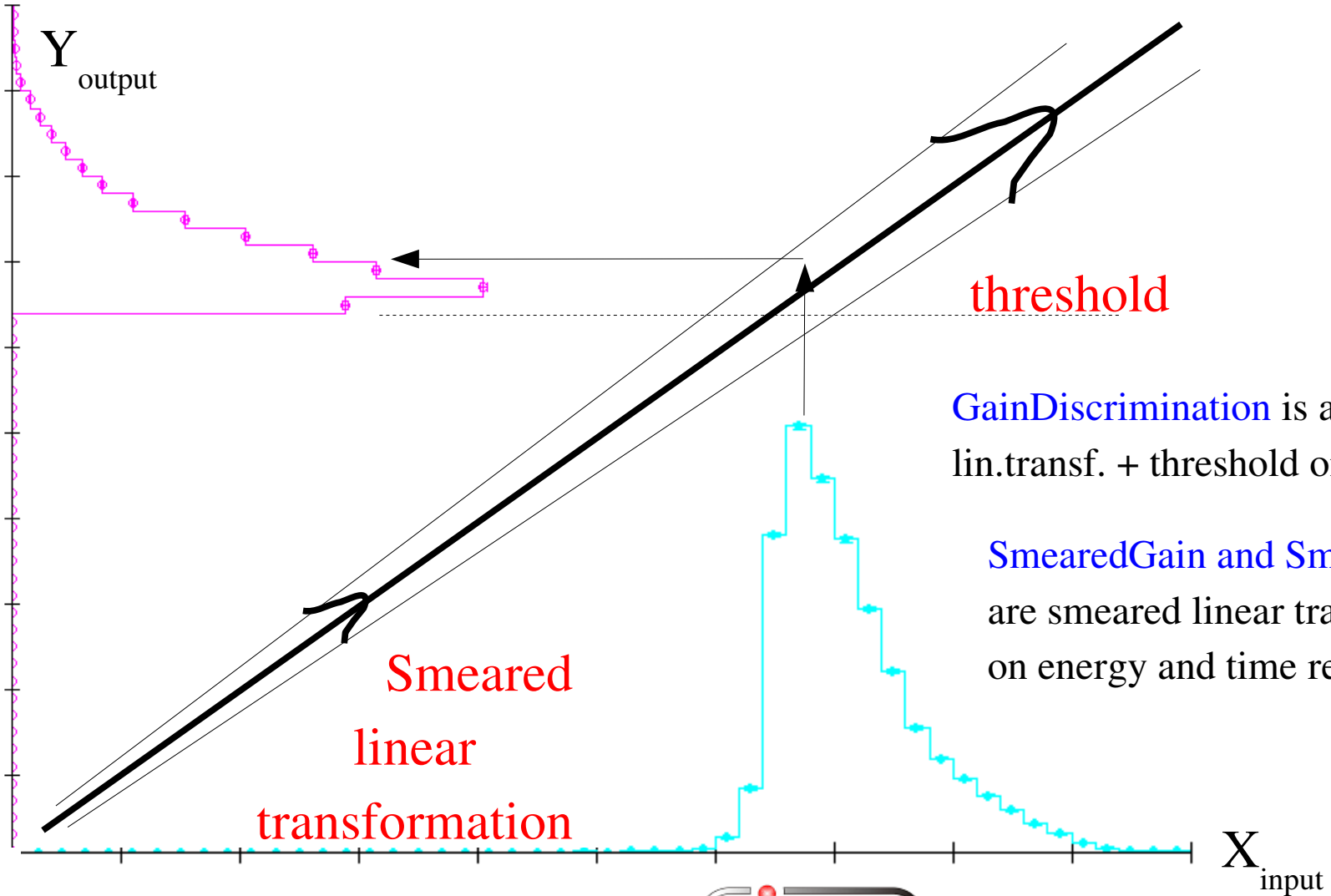
Setting up DigiSim modifiers

- Modifiers' role is to *tweak* hit energy/timing
- Useful: It helps to interpret the “energy” field according to the digitization process being 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 steering 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 HcalEndcapDigitizer
ActiveProcessors SimCalorimeterHitsProcessor
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 ------------------------------------------------
```

The same old-format steering files can be used for both Java and C++ versions.

The new XML-based steering files can only be used with Marlin/C++

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 digi.steer in DigiSim area)
.end ------------------------------------------------------------------
```

(As many modifiers 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 (Java only):

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

```
HBcrosstalk      Crosstalk                0.020     0.005
```

- Photon collection efficiency (with a QE ~ 15%):

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

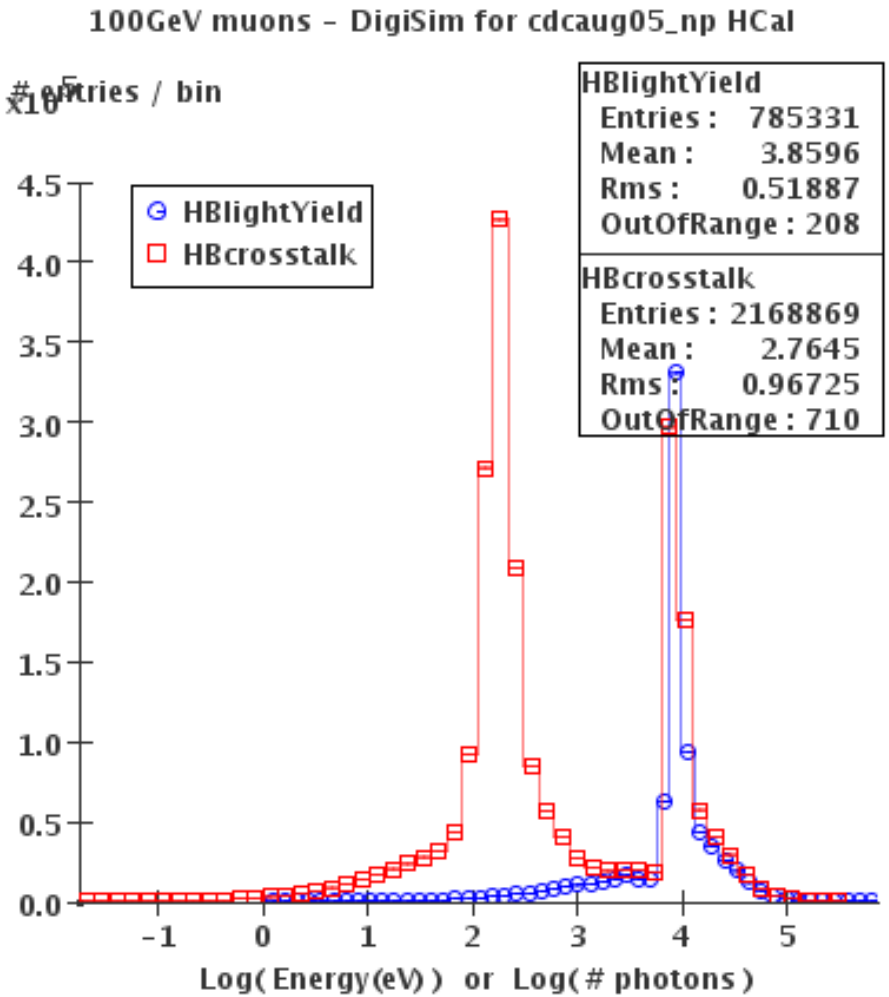
  15 / 0.15 = 100 incident photons on photosensor => Eff_{coll} = 100 inc. / 9000 tot.scint. = 0.0111

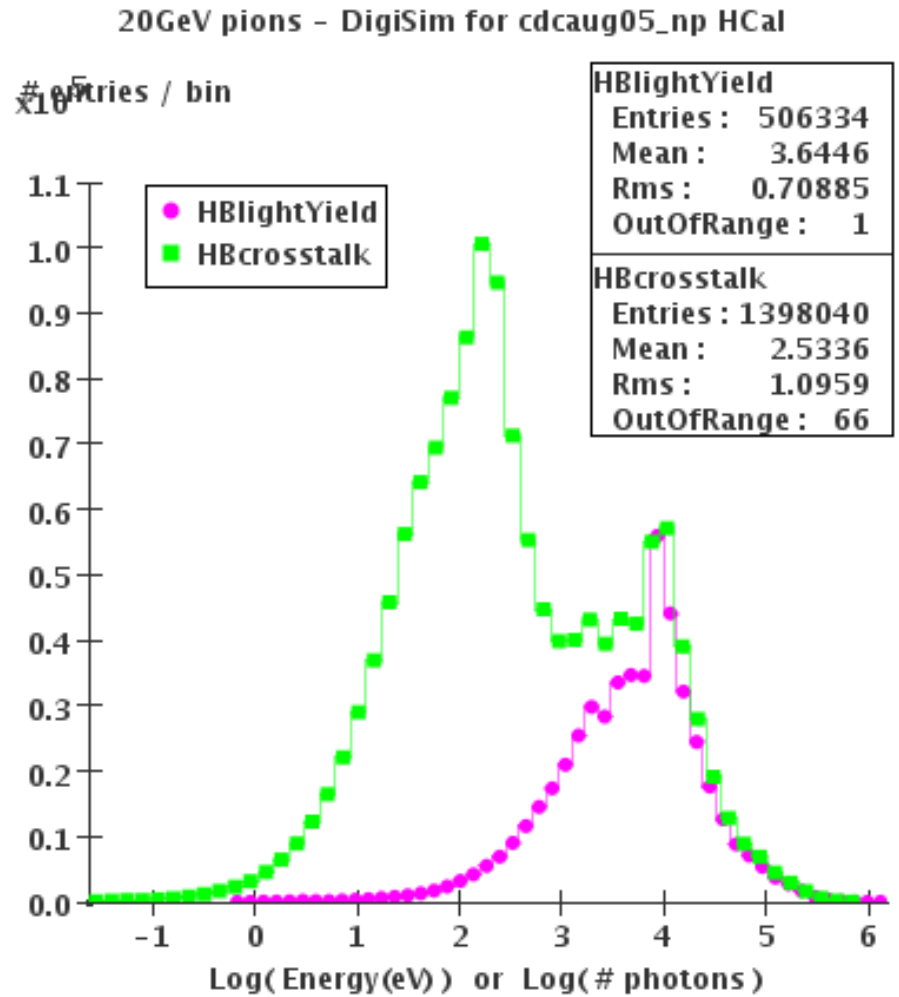
  Variance (Poisson): $sigma_N^2 = \langle N \rangle \rightarrow$ for $\langle N_{PE} \rangle = 15$, $(sigma_N / N) \sim 26\%$

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

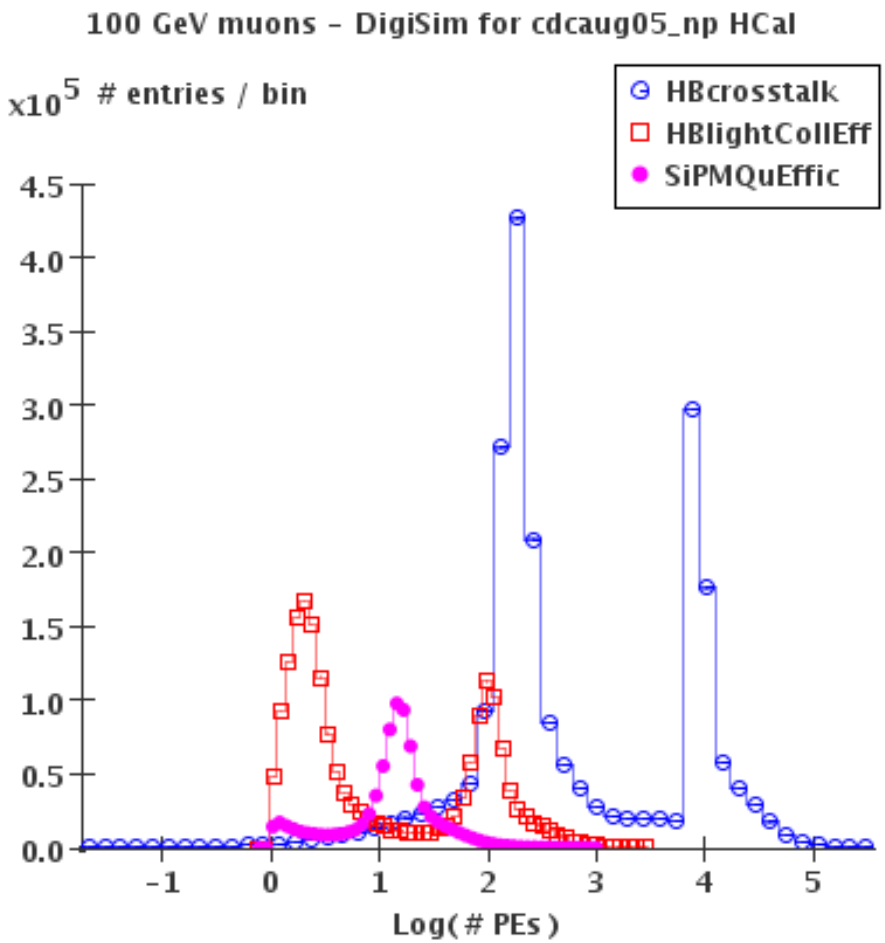
```
HBlightCollEff GainDiscrimination    0.0111    0.0029        1        0
```

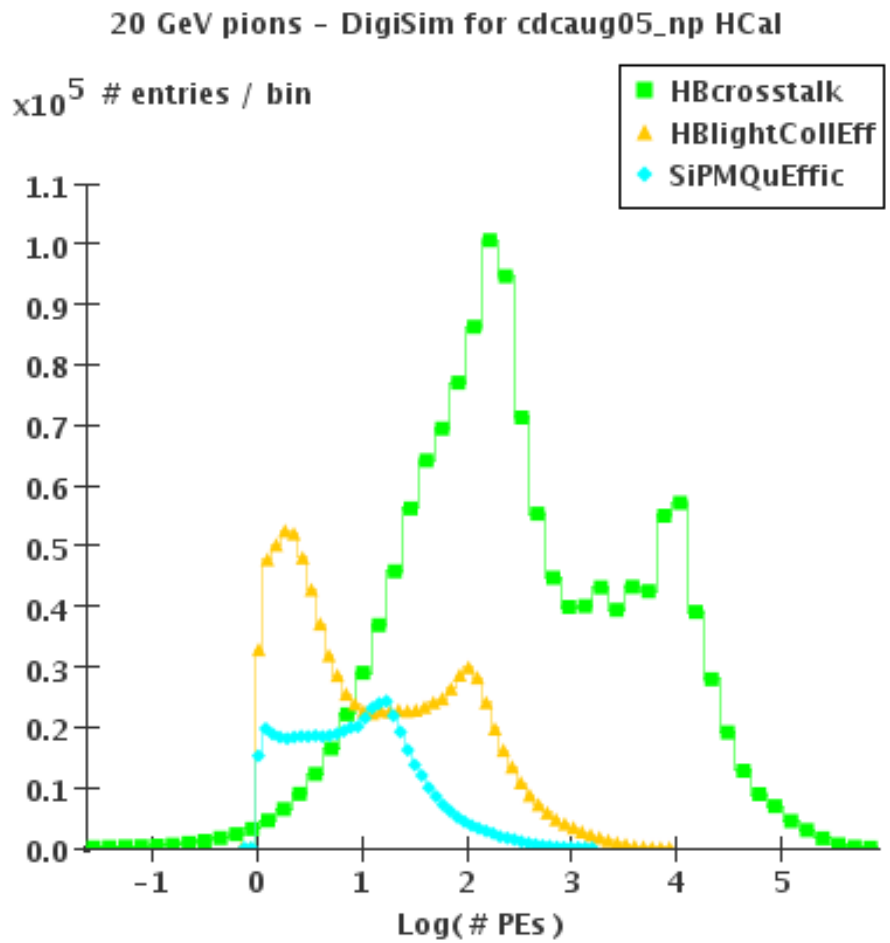
Effect of each modifier: Light cross-talk





Effect of each modifier: Photodetection





Simple example: parameters from the tile HCal

- Photosensor detection efficiency: QE ~ 15 %

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

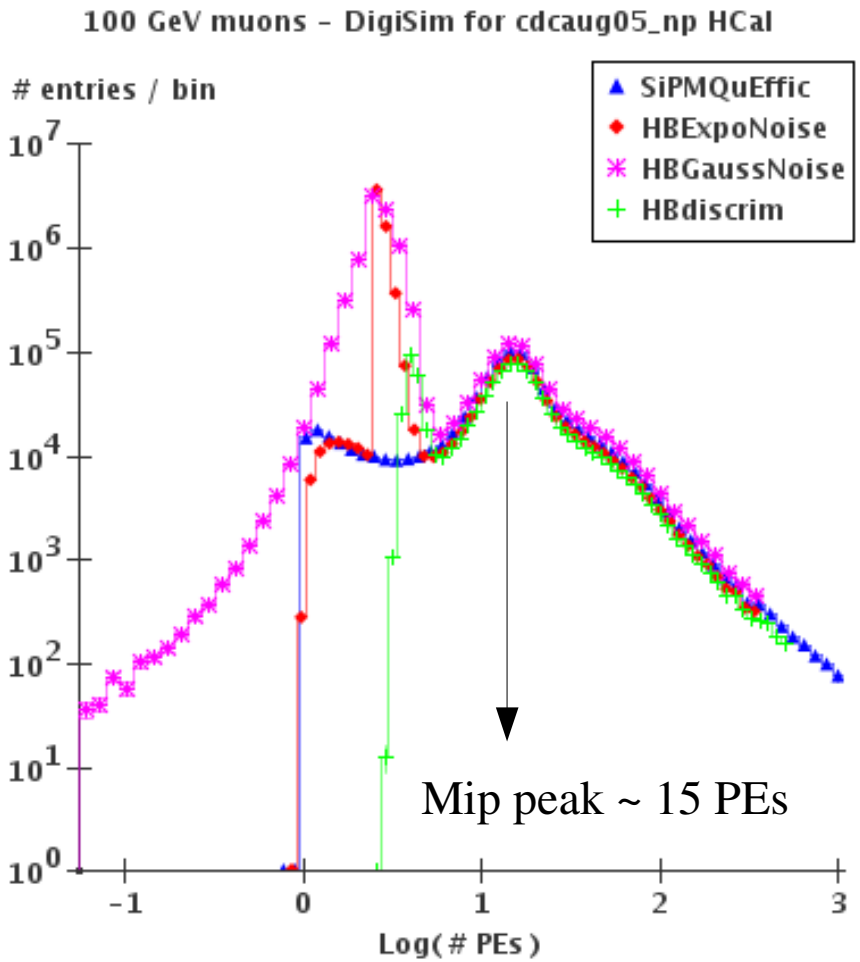
- Noise simulation:
  - Photosensor noise: exponential distribution (guess: mean 0.6)
  - Electronics noise: gaussian distribution (guess: mean 0, sigma 1.6, keep +/- tails)

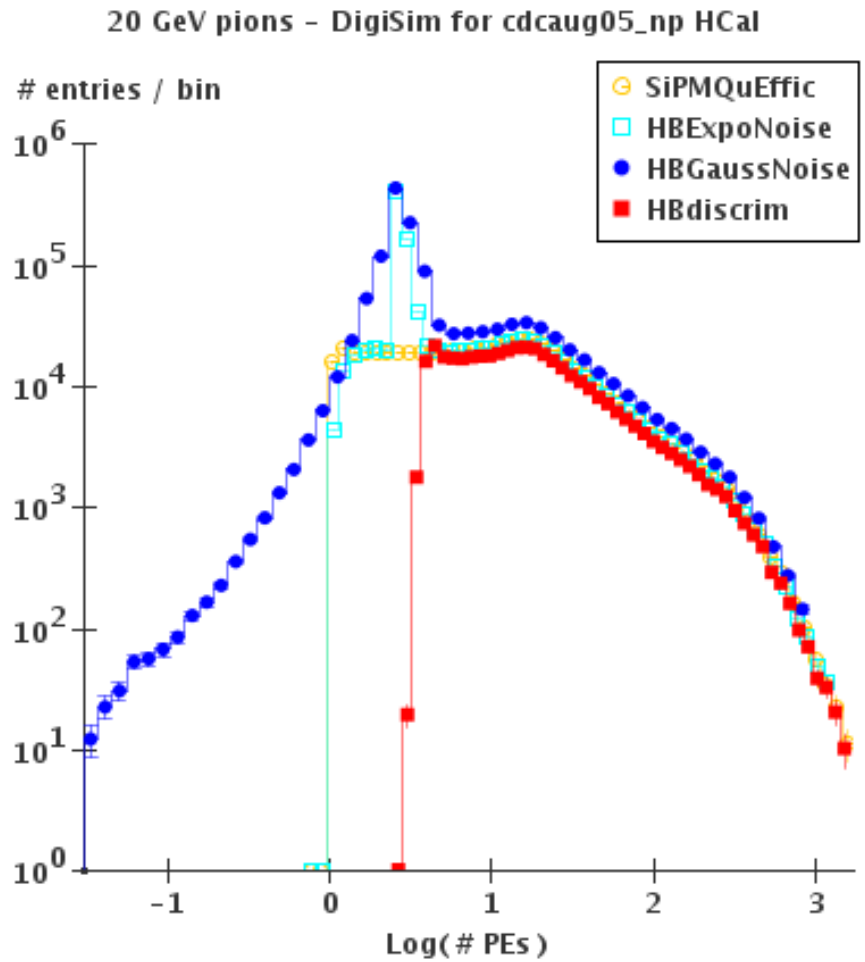
```
# GaussNoise parameters:        sys   be    Ecut   TimeNom  TSig   Mean  Sigma
# Note: sigma<0 means that threshold acts on absolute value only
HBGaussNoise   GaussianNoise      3     0    2.5       100   100    0.0  -0.58

# 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)

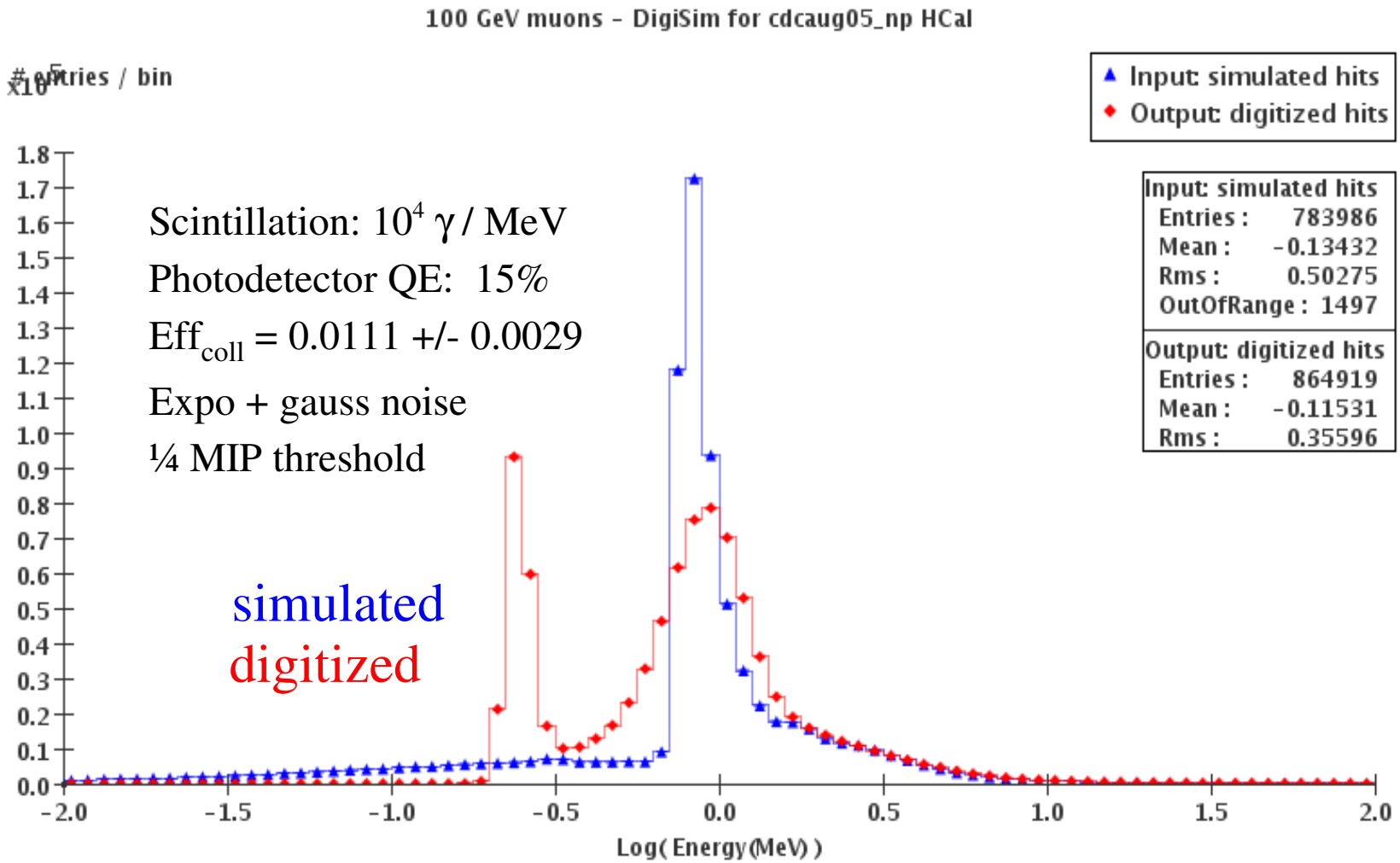
```
# Discrimination                                  threshold      sigma
HBdiscrim    AbsValueDiscrimination                     4        0.25
###HBdiscrim GainDiscrimination       1       0        4        0.25
```

Noise and discrimination

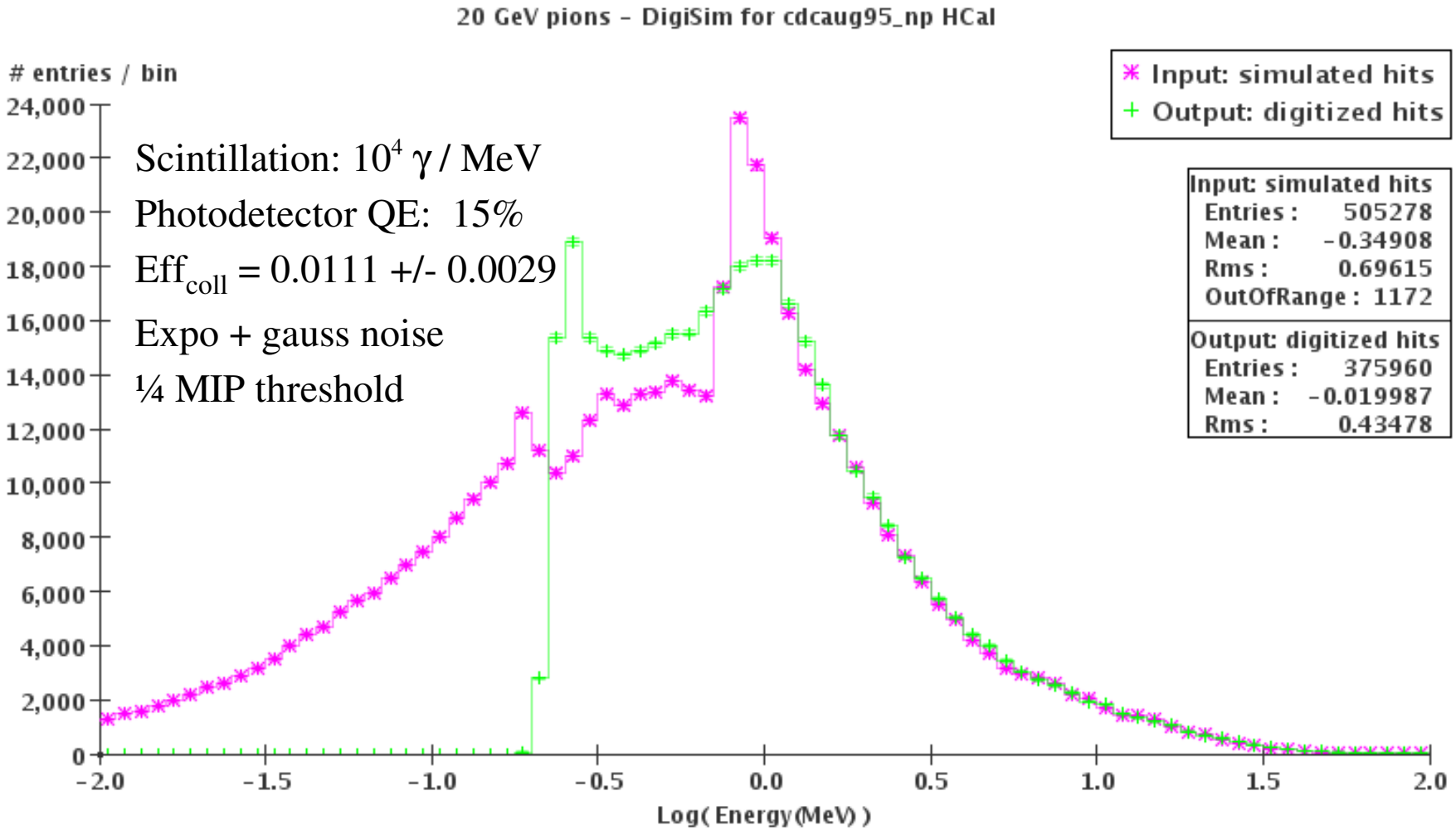




HCal scintillator digitization (preliminary)



HCal scintillator digitization (preliminary)



DigiSim for the test beam

- Cell ganging can, in principle, be integrated into the DigiSim structure
- Questions DigiSim may help to answer in the beam test:
  - Average detector behavior is good enough for simulating high-performance calorimetry?
  - How much can simulation performance be improved with non-uniform parametrizations?
  - How important are the technology-specific effects for the final detector performance?
- 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 EM calorimeters, trackers, and RPC- or GEM-based HCal detectors.
  See DigiSim configuration example presented in this talk.

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, but some functionality is missing (crosstalks and noise modeling), hopefully to be implemented soon
  - Calibrated (Sim)CalorimeterHits can be directly compared to calibrated real data
- Both C++ and Java versions are available through official CVS servers
  C++ in the Calice CVS repository and Java in the LCSim CVS repository
- 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
- Documentation available from http://nicadd.niu.edu/digisim, including build instructions
- Comments are welcome: lima@fnal.gov and/or rmcintos@nicadd.niu.edu