

Digitization Simulation using DigiSim

Guilherme Lima
for the ILC-software group at NIU



NORTHERN ILLINOIS
UNIVERSITY

Snowmass Workshop - Simulations
August 14-27, 2005

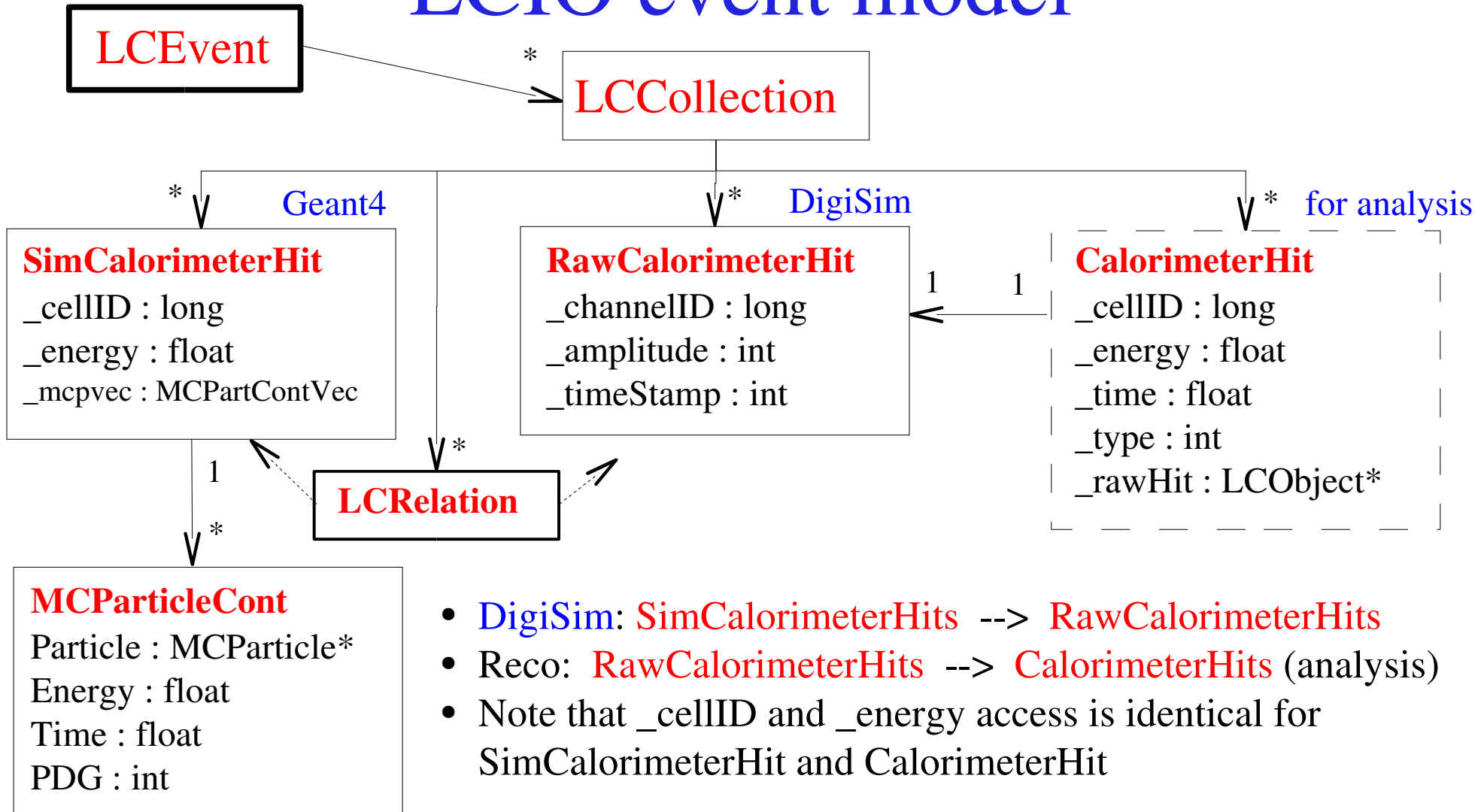
Talk Outline

- **DigiSim**: purpose and description
- Configuration
- An example: effects on hit energy distributions
- Usage instructions
- 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

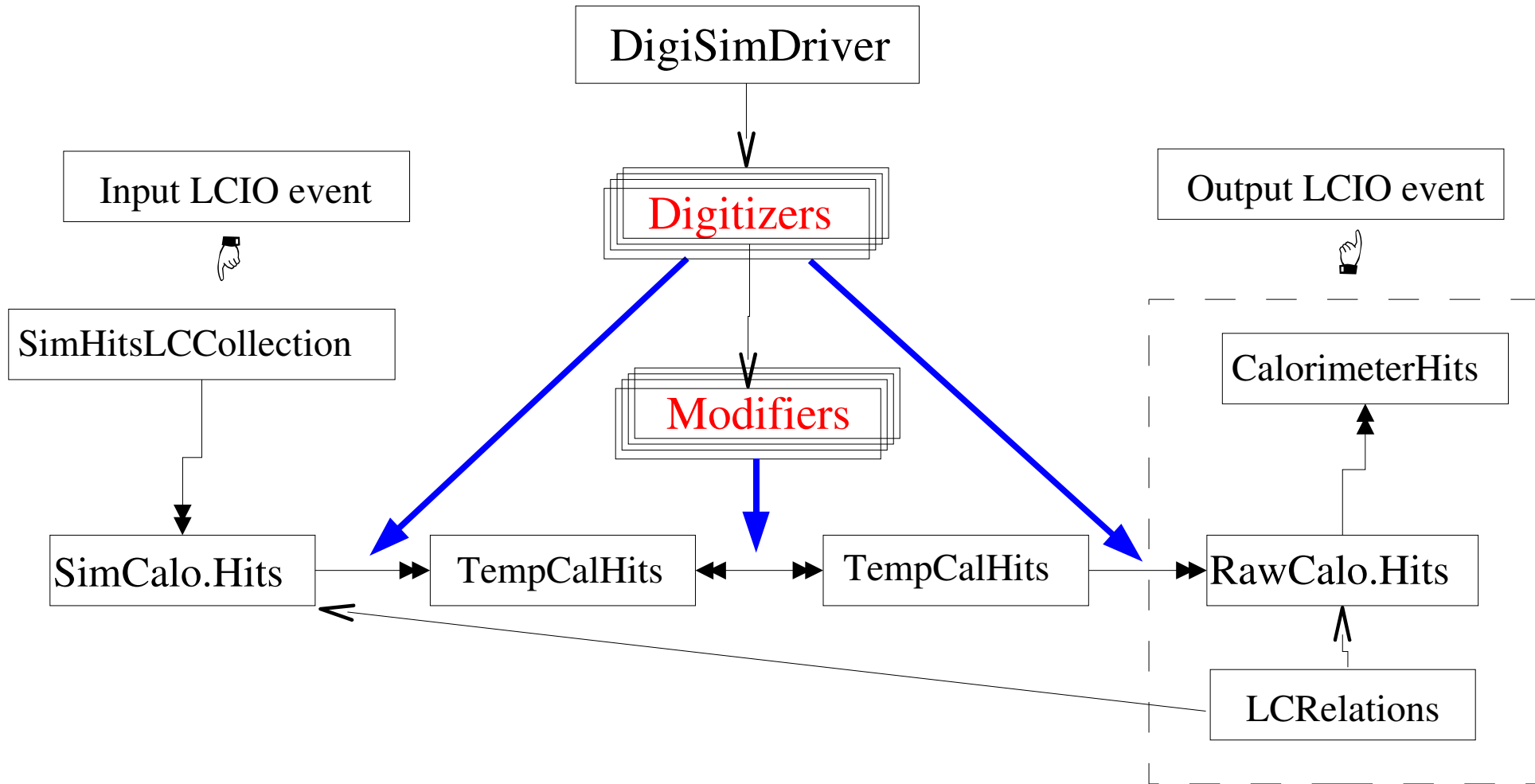
LCIO event model



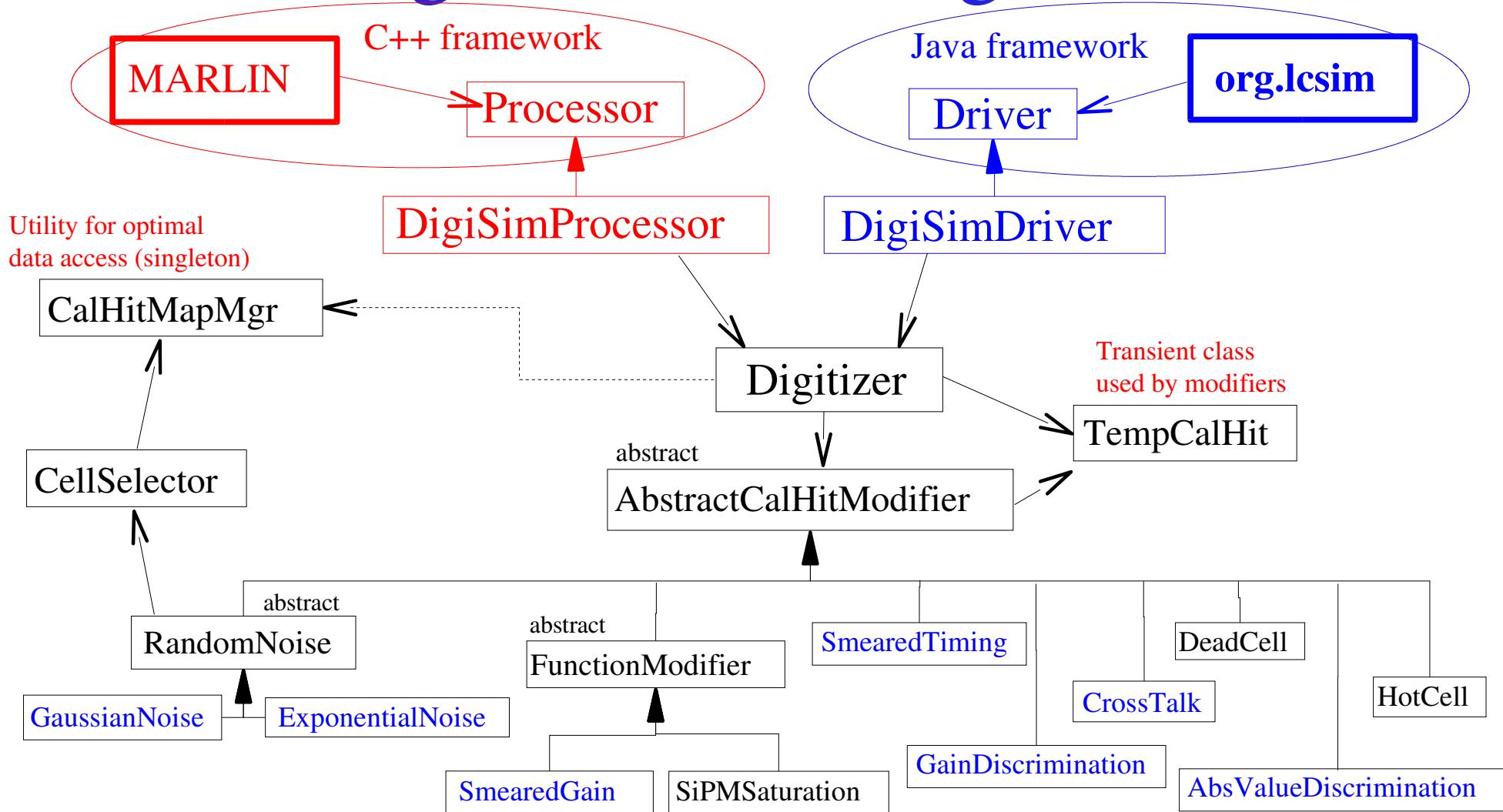
SimCalorimeterHits or CalorimeterHits?

- 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 values, use (same) cellid as a key to access SimCalorimeterHits.
 - Detector name is used to select the correct DigiSim configuration file.
 - Config files exist for all 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



Setting up a 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

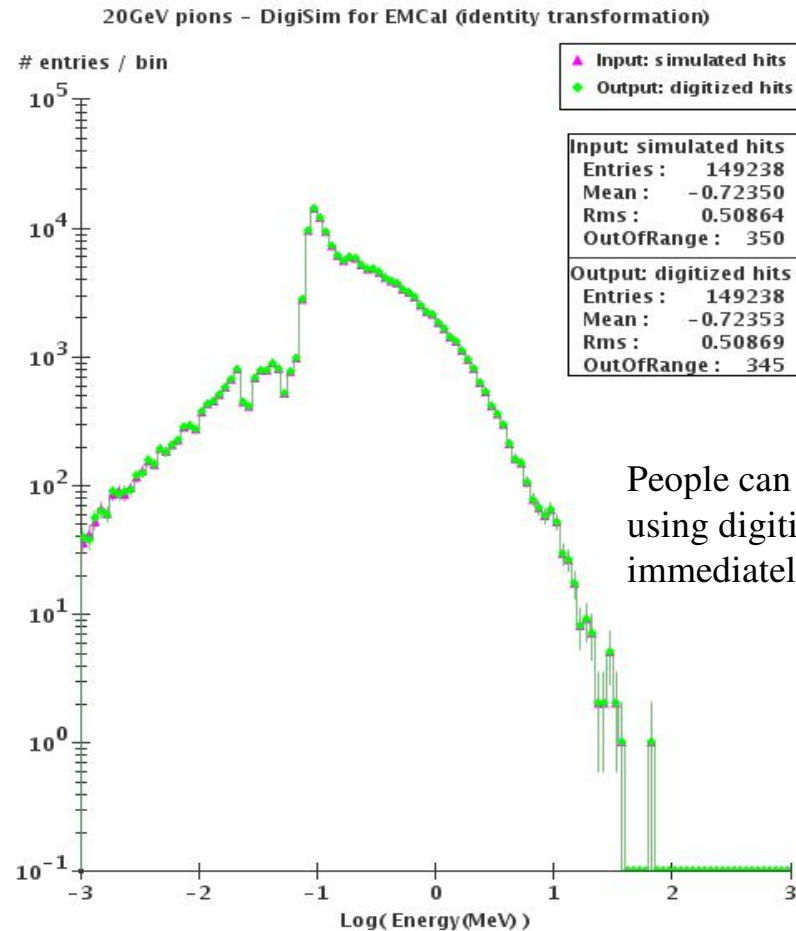
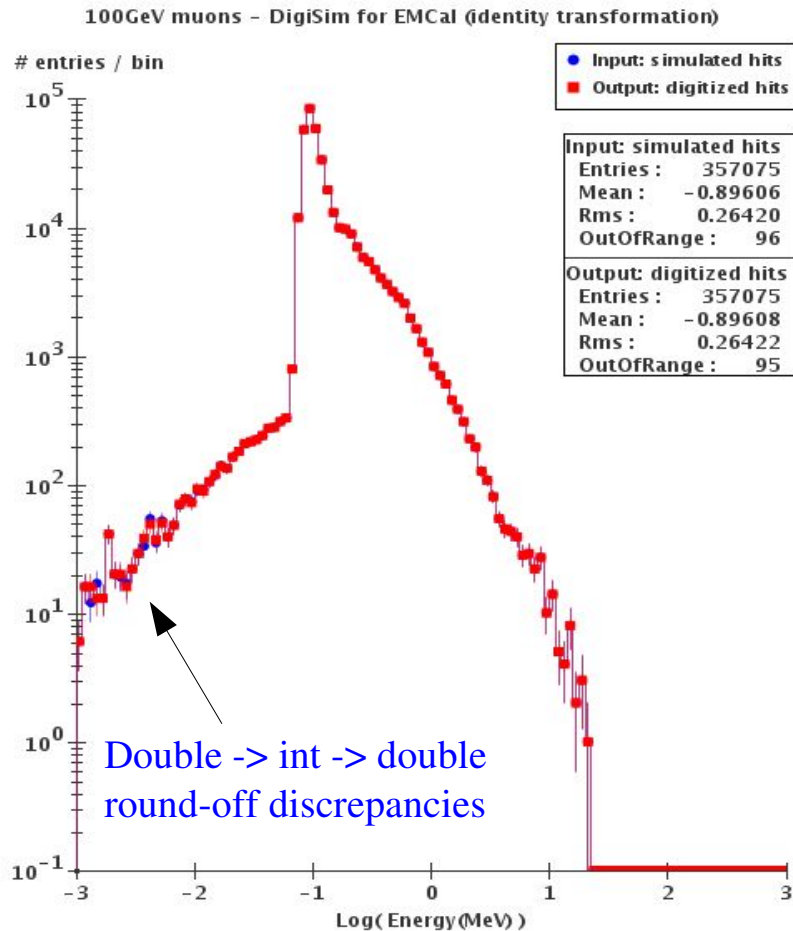
InputCollection      EcalBarrHits
OutputCollection     EcalBarrRawHits
Raw2SimLinksCollection EcalBarrRaw2sim

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

An identity transformation



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 $\sim 0.9\text{MeV}$, 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 $\Rightarrow \text{Eff}_{\text{coll}} = 100 \text{ inc.} / 9000 \text{ tot.scint.} = 0.0111$

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

Therefore: $\text{Eff}_{\text{coll}} = 0.0111 \pm 0.0029 \Rightarrow$ use GainDiscrimination with smearing

HBlightColleff	GainDiscrimination	0.0111	0.0020	1	0
-----------------------	---------------------------	---------------	---------------	----------	----------

Simple example: numbers 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:** $\frac{1}{4}$ 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
```

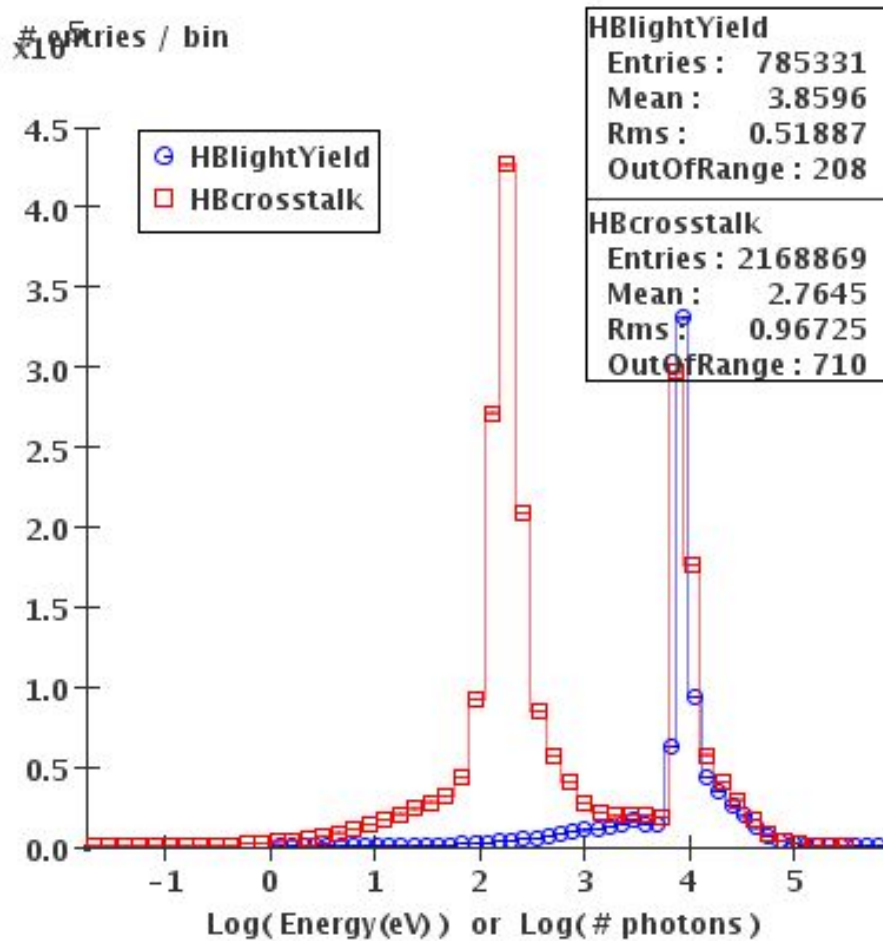
Configuring processors and modifiers

```
#####  
# 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 1000000 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 -----
```

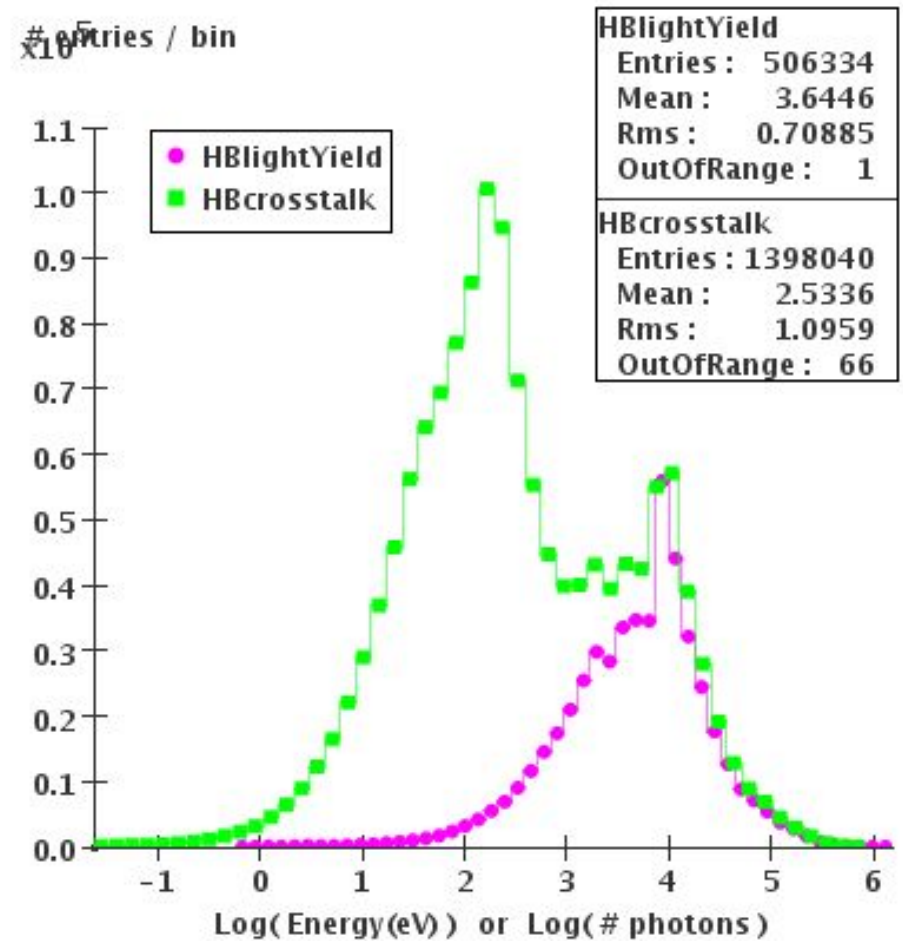
(As many as needed)

Crosstalk

100GeV muons - DigiSim for cdcaug05_np HCal

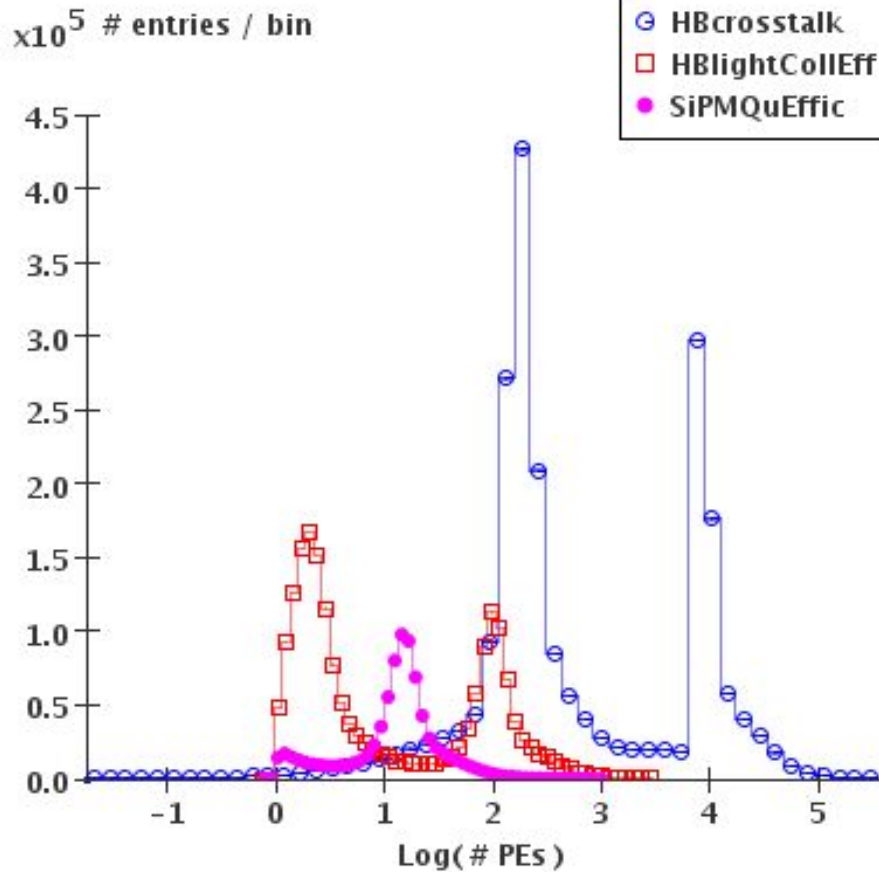


20GeV pions - DigiSim for cdcaug05_np HCal

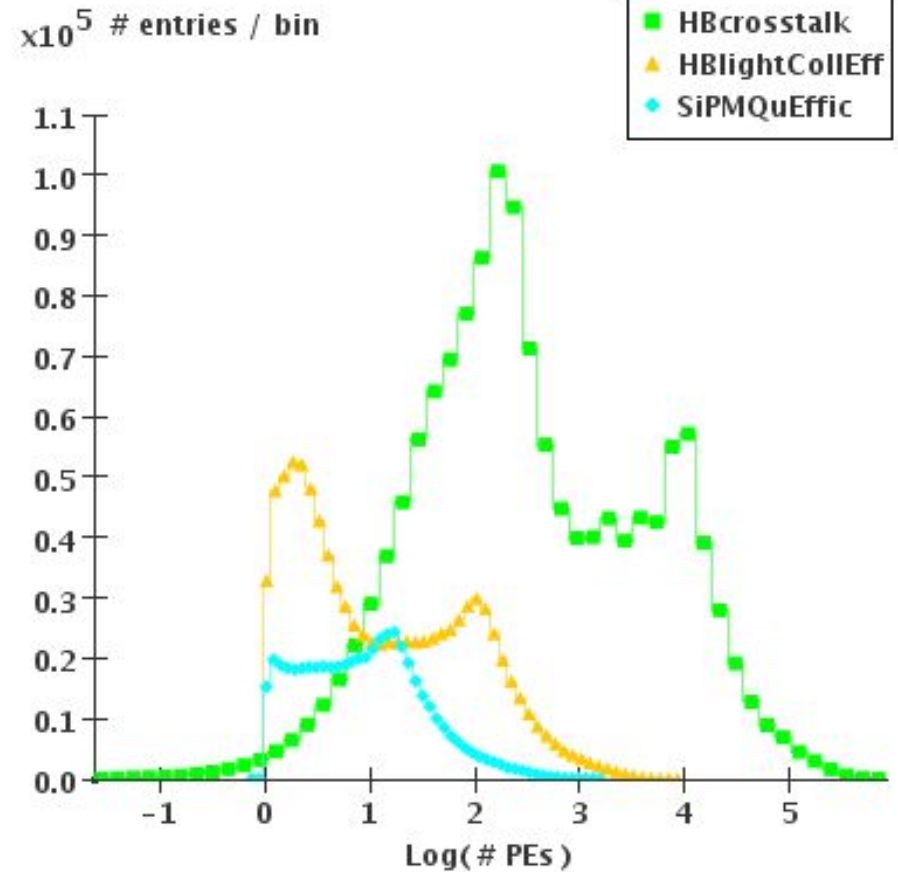


Photodetection

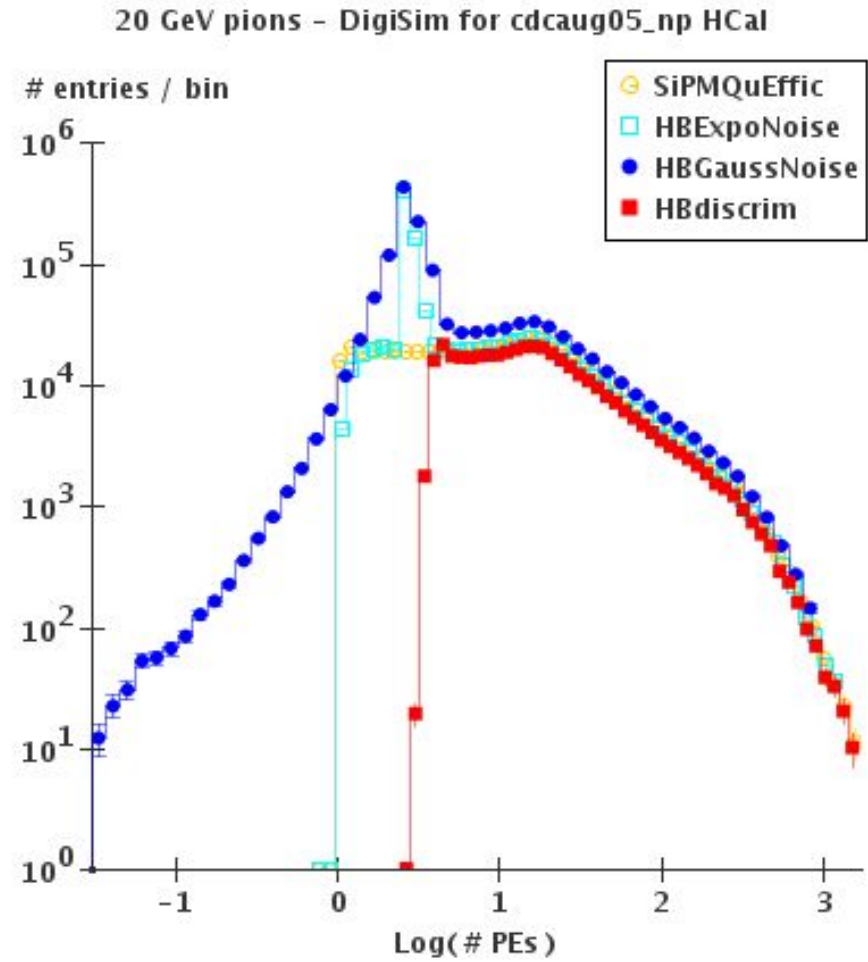
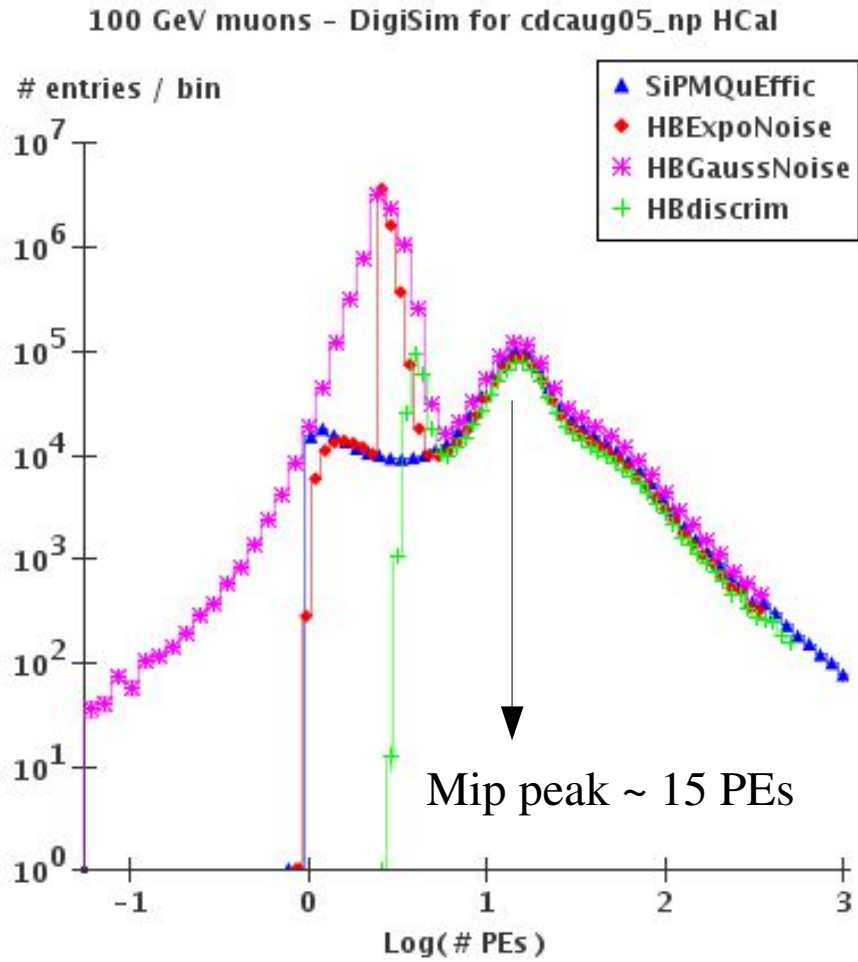
100 GeV muons - DigiSim for cdcaug05_np HCal



20 GeV pions - DigiSim for cdcaug05_np HCal



Noise and discrimination

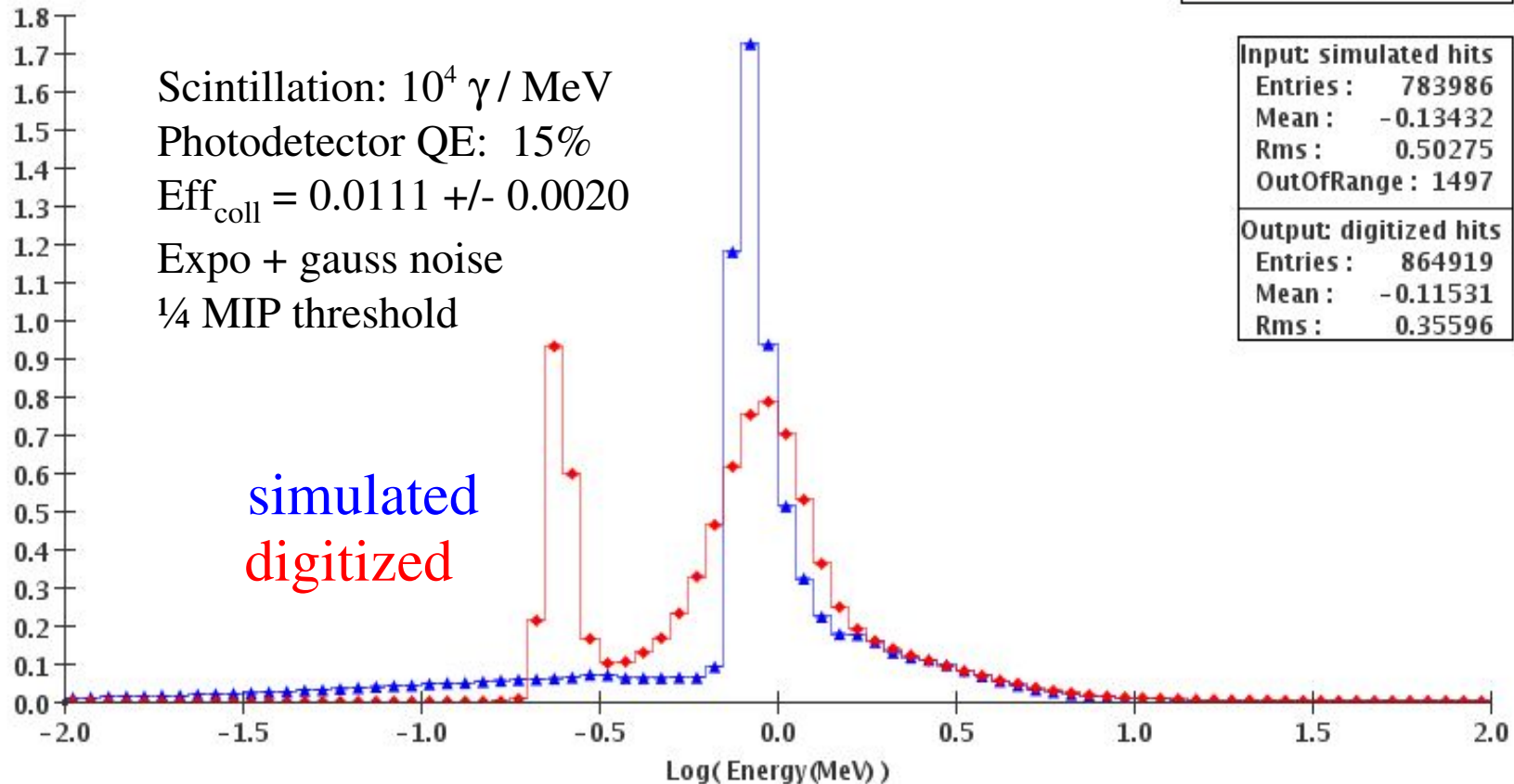


HCal scintillator digitization (preliminary)

100 GeV muons - DigiSim for cdcaug05_np HCal

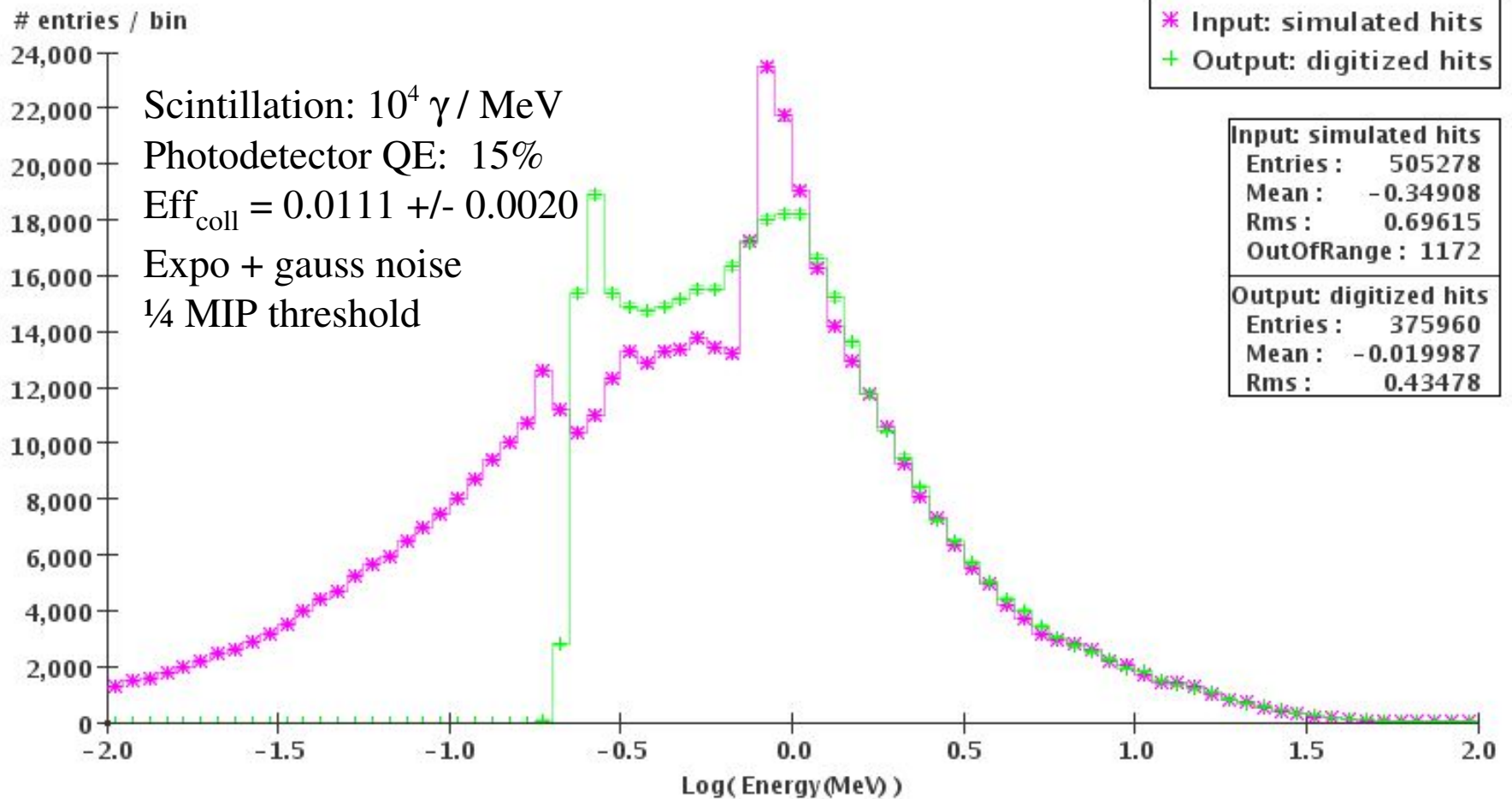
of entries / bin
 $\times 10^4$

▲ Input simulated hits
◆ Output digitized hits



HCal scintillator digitization (preliminary)

20 GeV pions - DigiSim for cdcaug95_np HCal



DigiSim usage instructions

- Download/install/build java 1.5, Maven, org.lcsim
(see <http://www.lcsim.org> for details)
- Drivers needed: (all available from org.lcsim.digisim)
CalHitMapDriver, DigiSimDriver and CalorimeterHitsDriver,
plus LCIODriver (for standalone run, saving output file)
or YourAnalysisDrivers (as an on-the-fly preprocessor)
- DigiSim configuration file stored on LCDetectors: digisim/digisim.steer
- Run it:
 - From command line: after setting the CLASSPATH (see docs for details)
java org.lcsim.digisim.DigiSimMain <inputfile>
an output file ./digisim.slcio will be produced, to be used for analysis or reconstruction
 - From inside JAS:
DigiSimExample is available from Examples -> org.lcsim examples

Running DigiSim inside JAS3

The image shows three overlapping screenshots of the JAS3 software interface, illustrating the steps to access the DigiSim example.

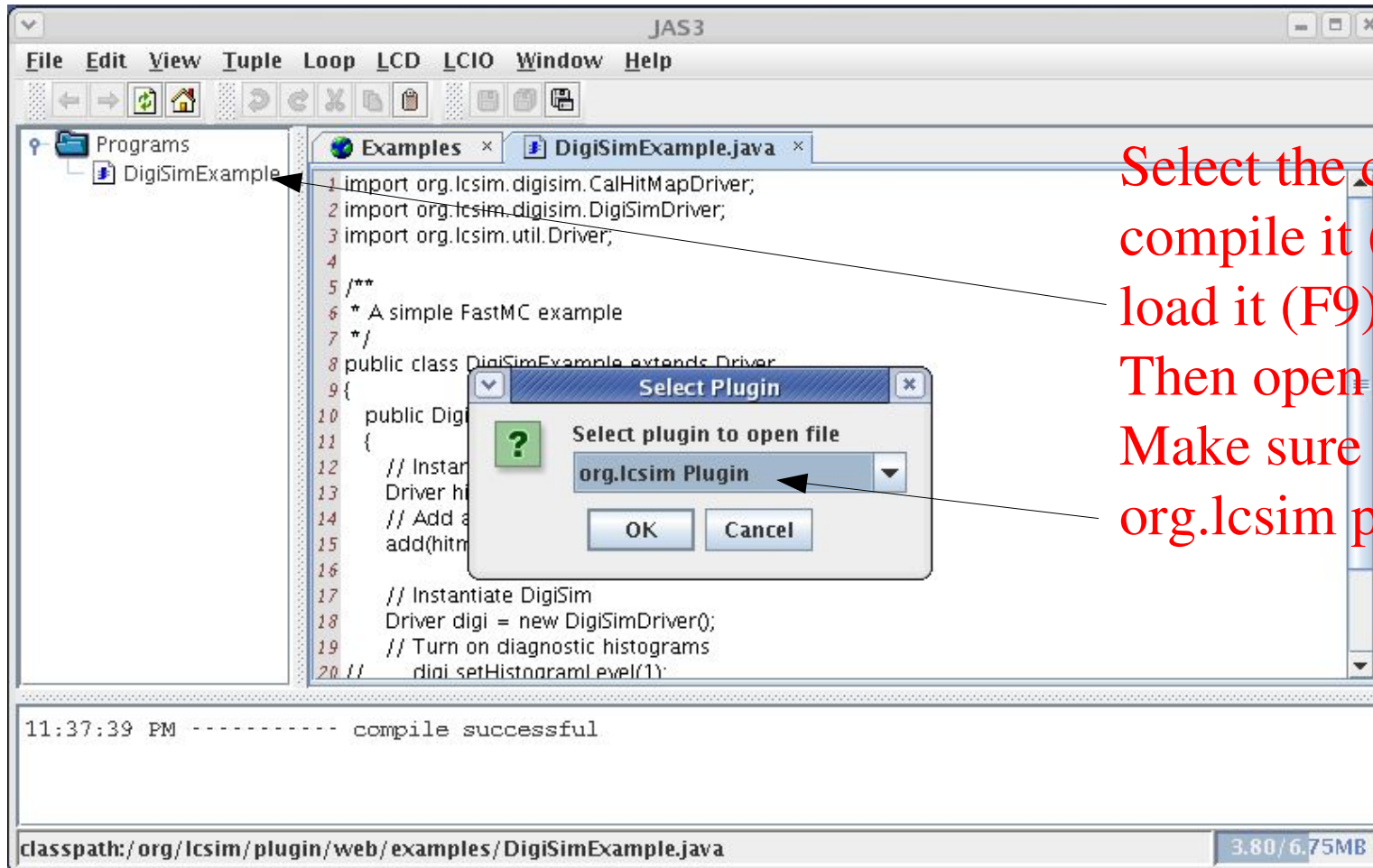
- Top Screenshot:** Shows the "Welcome" dialog box. The text reads: "Welcome to JAS3! JAS3 v0.8.1 (build 1274 on March 16 2005) Welcome to this release of JAS3 -- an AIDA compliant data analysis system. See the [release notes](#) for recent changes. For a quick start with scripting and programming see the [examples](#)". An arrow points from the "examples" link to the next screenshot.
- Middle Screenshot:** Shows the "Examples" dialog box. The text reads: "JAS3 examples The following example sets are installed: [Python Examples](#) AIDA examples written in Python. [Java Examples](#) AIDA examples written in Java. [Pnuts Examples](#) AIDA examples written in Pnuts. [org.lcsim Examples](#) Examples of using the org.lcsim software. Most of these examples can also be run in batch mode. See the release notes at <http://java.freehep.org/jas3/>". An arrow points from the "org.lcsim Examples" link to the final screenshot.
- Bottom Screenshot:** Shows the "org.lcsim examples" dialog box. The text reads: "org.lcsim examples These examples are written using the Java language. After opening them you need to compile and load them, and then use feed data to them using the Run menu." Below this is a table of examples:

SimpleGenerator.java	Simple diagnostic event generator.
Simple.java	Simply prints the event header of each event analyzed.
Analysis101.java	Intro to analysis with AIDA.
SimpleFastMC.java	Running the Fast MC.
SimpleOutput.java	Example of writing LCIO output
JetFinding.java	Using the Jet Finder
ClusterFinding.java	Cluster Finding example
DigiSimExample.java	Digitization example

Below the table, it says: "org.lcsim Jython examples for advanced users These examples are written in Jython. They have to be executed from within". An arrow points from the "DigiSimExample.java" link to the text "DigiSimExample.java" in the main text of the slide.

From starting page, click on examples
--> org.lcsim examples
--> DigiSimExample.java

Loading driver(s) and data



Select the code window,
compile it (F2) and
load it (F9).
Then open an LCIO file.
Make sure you select the
org.lcsim plugin.

Looking at raw hits...

The screenshot shows the JAS3 software interface. The main window displays the following information:

- Menu Bar:** File, Edit, View, Tuple, Loop, LCD, LCIO, Window, Help
- Toolbar:** Navigation and playback controls.
- File Explorer:** Shows a tree structure with folders: Programs, DigiSimExample, DataSets, and a sub-folder aida40080aida.
- Event Tree:** A tree view under 'Run:0 Event:0' showing various hit collections. The selected collection is 'HcalBarrRawHits'.
- Data Table:** A table with columns: CellID, Amplitude, and TimeStamp. The collection is identified as 'Collection: HcalBarrRawHits size:267 flags:28000000'.

CellID	Amplitude	TimeStamp
52918747320549765	43559	-90
285052684468619	129328	5
-720547249997738...	43239	262
-844317555949137	42692	136
27303132870476174	55524	64
-653014515675295...	63300	77
17735801160860075	-47417	53
-402483747540168...	57165	-62
19987330391605664	49677	-114
-678354006677909...	-42588	199
-363089019011068...	-51179	33
12950498923512232	48461	89
284764921659778	56520	5
284885180744070	277069	5
46163584102695307	48640	94
- Status Bar:** Shows 'Analyzed 1 records in 5812ms' and '6.56/8.27MB'.

...and raw -> sim links

The screenshot shows the JAS3 software interface. The main window displays a file tree on the left, a central event viewer, and a table of simulation links. The event viewer shows 'Run:0 Event: 3' and a collection of 'EcalBarrRaw2sim' with size 117 and flags 80000000. The table below lists the links between raw hits and simulated hits.

From	To	Weight
EcalBarrRawHits[0]	EcalBarrHits[31]	1.0000
EcalBarrRawHits[1]	EcalBarrHits[17]	1.0000
EcalBarrRawHits[2]	EcalBarrHits[104]	1.0000
EcalBarrRawHits[3]	EcalBarrHits[109]	1.0000
EcalBarrRawHits[4]	EcalBarrHits[8]	1.0000
EcalBarrRawHits[5]	EcalBarrHits[32]	1.0000
EcalBarrRawHits[6]	EcalBarrHits[56]	1.0000
EcalBarrRawHits[7]	EcalBarrHits[11]	1.0000
EcalBarrRawHits[8]	EcalBarrHits[10]	1.0000
EcalBarrRawHits[9]	EcalBarrHits[92]	1.0000
EcalBarrRawHits[10]	EcalBarrHits[20]	1.0000
EcalBarrRawHits[11]	EcalBarrHits[3]	1.0000
EcalBarrRawHits[12]	EcalBarrHits[38]	1.0000
EcalBarrRawHits[13]	EcalBarrHits[66]	1.0000

11:46:12 PM ----- compile successful

Compiler x Record Loop x

Analyzed 1 records in 327ms 8.15/12.8MB

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

Thanks!

A common transformation

