A digitization simulation package for the ILC and CALICE test beam

Guilherme Lima
for the CALICE group at NIU

CALICE Meeting at DESY
December 08, 2004
Talk Outline

- **DigiSim**: Purpose and requirements
- Package description
- Usage and configuration
- Developing new functionality
- Preliminary results
- Current status and summary
DigiSim

- Goal: a program to simulate the digitization process for the ILC detector simulation
- 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. Same reco / analysis software can be used for both
- 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-uniformities, etc.)
Requirements and choices

• Basic requirements:
  – Object oriented design to simplify maintenance and implementation of new functionality
  – Should be used within the CALICE test beam project, as a testbed for the reconstruction framework
  – All test beam code based on C++ and LCIO

• Gaede's Marlin chosen as the C++ framework (still using version available by ~Nov.20, 2004)
LCIO event model

**LCEvent**

- *LCRelation\[SimCalorimeterHit\]*
- *LCRelation\[RawCalorimeterHit\]*
- *LCRelation\[CalorimeterHit\]*

**SimCalorimeterHit**
- _cellID : int
- _energy : float
- _mcpvec : MCPartContVec

**RawCalorimeterHit**
- _channelID : int
- _amplitude : int
- _timeStamp : int

**CalorimeterHit**
- _cellID : int
- _energy : float
- _time : float
- _type : int
- _rawHit : LCOBJECT*

**MCParticleCont**
- Particle : MCParticle*
- Energy : float
- Time : float
- PDG : int

- **DigiSim**: SimCalorimeterHits --> RawCalorimeterHits
- **Reco**: RawCalorimeterHits --> CalorimeterHits
- **LCRelation** links not yet implemented
DigiSim class diagrams

- MARLIN
- Processor
- CalHitMapProcessor
  - CalHitMapMgr
- DigiSimProcessor
  - CalHitModifier
    - FunctionModifier
      - SmearedLinear
    - GainDiscrimination
      - CalDigiCrossTalk
  - TempCalHit

Marlin framework

Utility for optimal data access (singleton)

Function-based transformations

Transient class used by modifiers

Gain + threshold
DigiSim event loop

- Calorimeter hits are shown here, but the same structure could be used for trackers as well
- TempCalHits are both input and output to each modifier
- All processing is controlled by a DigiSimProcessor (one per subdetector)
- Modifiers are configured at run time, via the Marlin steering file
- New modifiers can be easily created for new functionality (more info later)
Steering file example

#######################################################
# 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 EMDigiSimProcessor
ActiveProcessors HADDigiSimProcessor
ActiveProcessors TCDigiSimProcessor
ActiveProcessors OutputProcessor

# limit the number of processed records (run+evt):
MaxRecordNumber 500
.end Global ---------------------------------------------
Configuring processors and modifiers

# A DigiSim processor. Instantiates one or more calorimeter hit modifiers, which together represent the full digitization process.

.begin EMDigiSimProcessor
  ProcessorType      DigiSimProcessor
  InputCollection    EMcalCollection
  OutputCollection   EMrawCollection
  CalorimeterType    EM

ModifierNames      EMFixedGain EMThreshOnly EMDigiIdent

# Parameters: type gainNom gainSig thresh thrSig
EMFixedGain        GainDiscrimination 1000000 0 0 0
EMThreshOnly       GainDiscrimination 1 0 30 0
EMGainThresh       GainDiscrimination 1000000 50000 30 1.5

# A function-based modifier
EMDigiIdent        SmearedLinear 1 0 1 0

.end

(As many as needed)
GainDiscrimination is a smeared linear transformation and threshold on energy.

SmearedLinear is a func-based smeared linear transformation on energy and/or timing.
Creating new modifiers

CalHitModifier

- setDebug(int) : void
- newInstance() : CalHitModifier* = 0
- init(floats) : void = 0
- processHits(TempCalHits) : void = 0
- print() : void = 0

FunctionModifier

- _par : floats
- init(floats) : void
- processHits(TempCalHits) : void
- print() : void
- transformEnergy(hit) : double = 0
- transformTime(hit) : double = 0

GainDiscrimination

- _par : floats
- newInstance() : CalHitModifier* = 0
- init(floats) : void
- processHits(TempCalHits) : void
- print() : void
- energyToADC(double) : double
- isBelowThreshold(double) : bool

CalDigiIdentity

- newInstance() : CalDigiIdentity* = 0
- transformEnergy(hit) : double = 0
- transformTime(hit) : double = 0

YourClassHere

Start by copying either one of the existing classes, then implement what is needed.
10 GeV ± on EM: gain + threshold

~140keV ⇒ 140 counts

“gain” = 1000000
threshold = 125

Fixed gain, no threshold (FGNT)
Gaussian gain, no threshold (GGNT)
Gaussian gain, fixed threshold (GGFT)
Gaussian gain, gaussian threshold (GGGT)
10 GeV $\pm$ on EM: gain + threshold

same plot as before, now with log-x
10 GeV $\pm$ on EM: time stamps

No time smearing in this example
$10 \text{ GeV} \pm \text{ on HAD: gain + threshold}$

$\sim 1.7 \text{MeV} \Rightarrow 1700 \text{ ADC counts}$

$\text{“gain”} = 1000000$

$\text{threshold} = 1750$
10 GeV ± on HAD: gain + threshold

Comparing HAD energies - 10 GeV pions - SDJan03

- Fixed gain - No threshold
- Gauss gain - No threshold
- Gauss gain - Fixed threshold
- Gauss gain - Gauss threshold

FGNT
GGNT
GGFT
GGGT

Entries:
Mean:
Rms:
OutOfRange:

Entries:
Mean:
Rms:
OutOfRange:

Entries:
Mean:
Rms:
OutOfRange:

Entries:
Mean:
Rms:
OutOfRange:
10 GeV ± on HAD: time stamps
Status

• A first version of DigiSim (proof of concept) is implemented
  
  - Two real modifiers implemented: GainDiscrimination and function-based SmearedLinear
  
  - LCRelation: to be implemented (soon?), example code at hand (thanks to F.Gaede)

• Output LCIO files contain EM and HAD RawCalorimeterHit collections, while keeping simulation collections untouched.

• Analysis code for plotting raw hits, plots confirm expected behavior.

• Creation of new (simple) modifiers is quite easy, by just copying one of the existing modifier classes and implementing the desired transformation.

• Some complicated ones: crosstalk and cell ganging require neighborhood information, from geometry-aware classes. NonProj code exists (java and C++), but projective geometry only available in java.
Summary

- A first version of a digitization simulation program, DigiSim, has been developed at NIU, aiming at full digitization of ILC detector simulations.
- DigiSim is object-oriented, powerful, extensible, yet very simple implementation using C++.
- Digitization of tracking detectors can probably be implemented as easily as the calorimeter hits.
- CALICE test beam can use DigiSim as a testbed, for evaluation and improvement suggestions. Please use it for your digitization studies.
- Send any questions or comments to me: lima at nicadd.niu.edu.
- Documentation available from http://nicadd.niu.edu/~lima/digisim, including download and building instructions.
Danke!
10 GeV ± on EM: time stamps

Same plot as before,
now in log scale

x~1, + log scale
= binning effect
10 GeV ± on HAD: time stamps

Comparing HAD time stamps – 10 GeV pions – SDJan03

- Fixed gain – No threshold
  - Entries: 450265
  - Mean: 0.88796
  - Rms: 0.45312
  - OutOfRange: 6

- Gauss gain – No threshold
  - Entries: 450265
  - Mean: 0.88796
  - Rms: 0.45312
  - OutOfRange: 6

- Gauss gain – Fixed threshold
  - Entries: 117267
  - Mean: 0.80961
  - Rms: 0.29871

- Gauss gain – Gauss threshold
  - Entries: 116450
  - Mean: 0.81192
  - Rms: 0.29984
  - OutOfRange: 1