

Time Offset Issues in Simulation and Reconstruction

General Larsoft Meeting
Feb. 13, 2013

H. Greenlee

Processing Stages

- Generators.
 - Various modules (data product MCParticle).
- Geant4 simulation.
 - LArG4 module (data products MCParticle, SimChannel).
- Detector simulation.
 - SimWire module (data product RawDigit).
 - OpMCDigi module (data product OpDetPulse).
- Reconstruction.
 - CalWire module (data product Wire).
 - Optical reconstruction.

Generators

- All generators have fcl parameters that allow you to specify the time or time range of generated particles.
 - Every generator is different. There is no uniformity of parameter names or units among the various generators.
 - See “Simulation” article in “Documentation” section of larsoft wiki for a list of relevant parameters.

Geant4 Simulation

- LArG4 simulates:
 - Time-of-flight, and secondary particle production and decay
 - These processes usually introduce negligible additional time delays compared to electron drift (muon decay is a possible exception).
 - Electron drift (LArVoxelReadout).
 - Major component of time delay (maximum drift time 1.6 ms for microboone).

LArVoxelReadout Time-to-Distance

- LArVoxelReadout time-to-distance calculation:
 - $t = t_0 + t_s + x_s/v$.
 - t = Time when electrons reach readout wire.
 - t_0 = Time offset (plane-depenent).
 - t_s, x_s = Simulated time and position of ionization (not including electron drift). Mainly inherited from generator time.
 - v = drift velocity.
 - $t_0 = t_1 + t_2 + t_3$.
 - t_1 = drift time from $x=0$ to first readout plane using bulk drift velocity.
 - t_2 = drift time from first readout plane to current readout plane.
 - t_3 = trigger offset.

Reconstruction Time-to-Distance

- Reconstruction time-to-distance is implemented in DetectorProperties service, method ConvertTicksToX.
 - $x = v (t - t_0)$.
 - Inverse of LArVoxelReadout time-to-distance (assumes $t_s=0$).
 - Same t_0 as LArVoxelReadout (but calculated independently, therefore maintenance risk).
 - Currently no provision to add a reconstructed time offset (reconstructed equivalent of t_s , e.g. time of optical flash in beam gate).

Interplane Drift Time Offset

- The drift time from the first readout plane to the current readout plane (t_2) is currently calculated using the bulk electric field instead of the actual plane gap electric field.
 - Microboone bulk electric field $E = 500$ V/cm.
 - First gap electric field $E = 667$ V/cm.
 - Second gap electric field $E = 800$ V/cm.
- Time error from using wrong electric field.
 - First gap = 0.37 ticks.
 - First + second gaps = 0.96 ticks.
 - Not huge, but not negligible. Should be fixed.

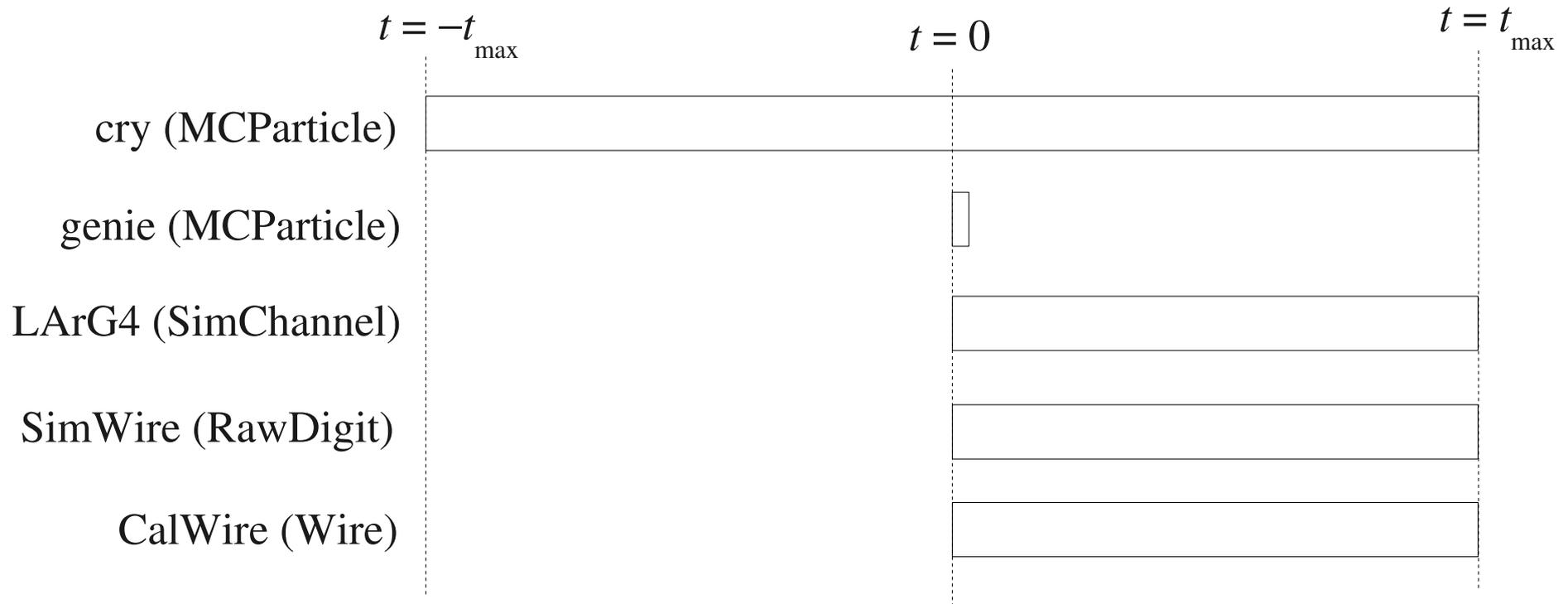
Simulating Multiple Readout Windows

- Real daq system will record three contiguous readout windows (pre-spill, in-time, and post-spill).
- Simulation may or may not simulate out-of-time (pre-spill and post-spill) readout windows.
- `DetectorProperties` has two parameters related to readout windows.
 - `ReadOutWindowSize`
 - Maximum drift time in ticks (3200 for microboone).
 - `NumberTimeSamples`
 - Can be 1x or 3x `ReadOutWindowSize`, depending on whether you want to simulate out-of-time windows or not.

Simulating Multiple Readout Windows

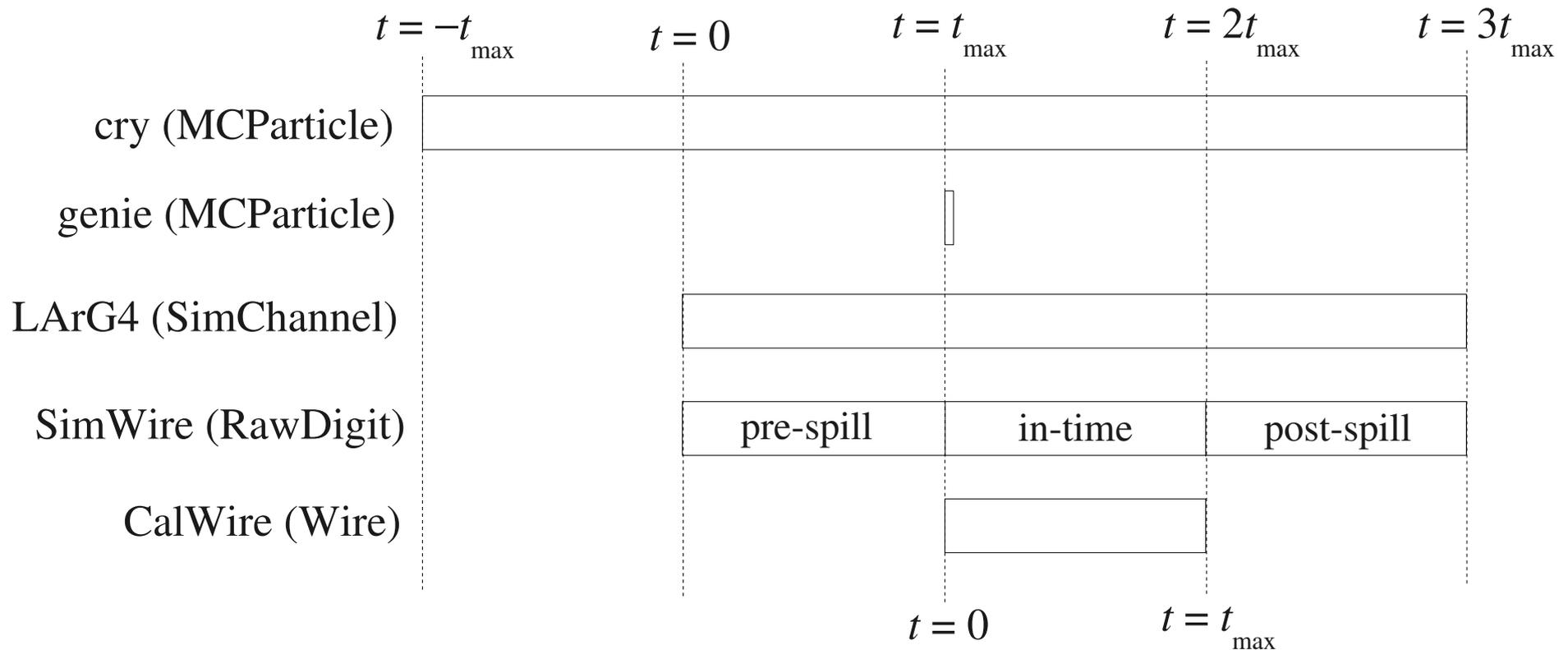
- Effect of setting readout window parameters on simulation and reconstruction chain:
 - No effect on generators.
 - No effect on LArG4.
 - Major effect on detector simulation.
 - SimWire will make one or three RawDigit data products depending on value of NumberTimeSamples and ReadOutWindowSize.
 - OpMCDigi ignores currently (this is a bug).
 - Standard reconstruction only uses maximum drift time (ReadOutWindowSize), does not know about NumberTimeSamples.
 - Above will not apply to cosmic rejection, obviously.
 - Analysis that includes back-tracking the simulation to the geant or generator level needs to know whether out-of-time readout windows were simulated (see following slides).

Configuring One-Window Simulation



- Cosmic ray generator should start at $t = -t_{\max}$ and have an integration time of twice the maximum drift time.
- Beam-related generators (genie) should start at $t = 0$, with optional random offset up to the beam gate length.

Configuring Three-Window Simulation



- Cosmic ray generator should start at $t = -t_{\max}$ and have an integration time of four times the maximum drift time.
- Beam-related generators (genie) should start at $t = t_{\max}$, with optional random offset up to the beam gate length.

Simulation Time Offset

- In the case of one-window simulation (like most simulation we do now), reconstructed hits are in-time with simulated charge objects (SimChannel).
- In the case of three-window simulation, there is a time offset of the full drift time between reconstructed hits and simulated charge.
- To successfully back-track three-window simulated events, two things have to be done.
 - NumberTimeSamples parameter must be set correctly in DetectorProperties service.
 - BackTracker service must be taught to take into account simulation time offset.
 - It doesn't do this now (this is a bug).

Suggested Code Changes

- **DetectorProperties service.**
 - Add methods to convert between reconstructed time (ticks) and geant time (tdc counts).
 - `double DetectorProperties::ConvertTicksToTDC(double).`
 - `double DetectorProperties::ConvertTDCToTicks(double).`
 - Add `preProcessEvent` callback to auto-sense proper value of `NumberTimeSamples` parameter (so users won't have to change configuration depending on number of simulated readout windows).
- **BackTracker service.**
 - Invoke `reco vs. geant` time conversion at appropriate places.
- Above changes implemented and tested in my private test release, but not committed to svn yet.

How to Auto-Sense NumberTimeSamples

- It is usually possible to determine whether events were simulated with one or three windows by looking at various data products. However, there is no ideal way to check now (that I know of).
 - One can look whether there are one or three RawDigit data products.
 - This is the method that I implemented in my private test release. I believe this is currently the best way.
 - Will not work if RawDigits are dropped in future.
 - One can look at MCParticle or SimChannel object times.
 - Not necessarily foolproof (nothing prevents times from spilling across readout window boundaries).
 - More complicated, less efficient (may have to look at many objects).
 - Introduces circular dependency between Utilities and Simulation.

Summary of T0 Issues

- Plane gap drift time offset bug should be fixed (in LArVoxelReadout and DetectorProperties).
 - It would be better to collect common aspects of time offset in one place (i.e. DetectorProperties).
- Most cosmic ray generator fcl files are wrongly configured (including ones used for microboone MC challenge).
 - That is why there are too few cosmic ray tracks in MC challenge files.
 - Cosmic ray fcl files in ubooneoffline repository are fixed (also added 3-window cosmic ray fcl files and cosmic overlay fcl files).
- OpMCDigi.
 - Can not handle negative time photons (crashes).
 - Does not handle three-window time offset correctly.

Summary of T0 Issues (cont.)

- BackTracker
 - Does not handle three-window time offset correctly.
- DetectorProperties.
 - Add common t_0 calculation.
 - A mechanism should be added to feed back a reconstructed global time offset into time reconstructed time-to-distance calculation.
 - Auto-sense proper value of NumberTimeSamples parameter.
- When using three-window simulation, people need to be aware that they need to modify generator fcl files (applies to all generators).
 - In general, standard three-window versions of genie, cry, and other generator fcl files do not exist in larsoft.

Summary of T0 Issues (cont.)

- All experiments are currently inheriting argoneut trigger offset of 60 ticks.
 - This value probably makes no sense for any experiment except argoneut.
 - Doesn't affect reco vs. mc truth position agreement, but it shifts the in-time window acceptance.
 - I assume this was added to make simulated argoneut data look like real data.
 - Daq experts should determine the correct value for microboone.
 - My personal guess is that the correct value is zero, or at least it should be zero until we know different.

The Configuration Problem

- There are currently several parameters (e.g. NumberTimeSamples, TriggerOffset) that are owned by services (DetectorProperties, LArProperties, Geometry), that are required to be set independently via fcl for each step in processing chain.
 - No easily accessible record of what these parameters were is saved in the data.
 - Puts an unreasonably large burden on analyzer.
 - Source of these parameters may change in future (e.g. they may come from database instead of fcl parameter).
- For above reasons, it will be very convenient to have a record in the data of what some of these configuration parameters were for each processing stage, for both simulation and real data.

DetectorProperties Parameters

- Sampling rate.
- Trigger offset.
- Gain (Electrons to ADC).
- Number of time samples.
- Readout window size.
- Plane-dependent time offsets.

LArProperties

- These parameters should be saved.
 - Electric field in bulk and gaps.
 - LAr temperature.
 - Drift velocity.
 - Electron lifetime.
- Physical and ionization parameters do not need to be saved.
 - Density.
 - Radiation length.
 - Bethe-Bloch (dE/dx) parameters.
- Optical parameters also do not need to be saved.

Geometry Parameters

- SurfaceY (fcl parameter).
 - Not sure if this parameter is actually used anywhere.
- Geometry name.
 - Name of gdml and root file.

Configuration Data Product

- I suggest to add a data product which will store the values of the previously mentioned parameters (about 15 numbers and one string) in each event.
- In future, services can add `preProcessEvent` callback to update any or all of these parameters from configuration data product (as an alternative to current fcl parameters, databases, etc.).
 - Configuration data products from different processing stages can be distinguished by module label or instance name.