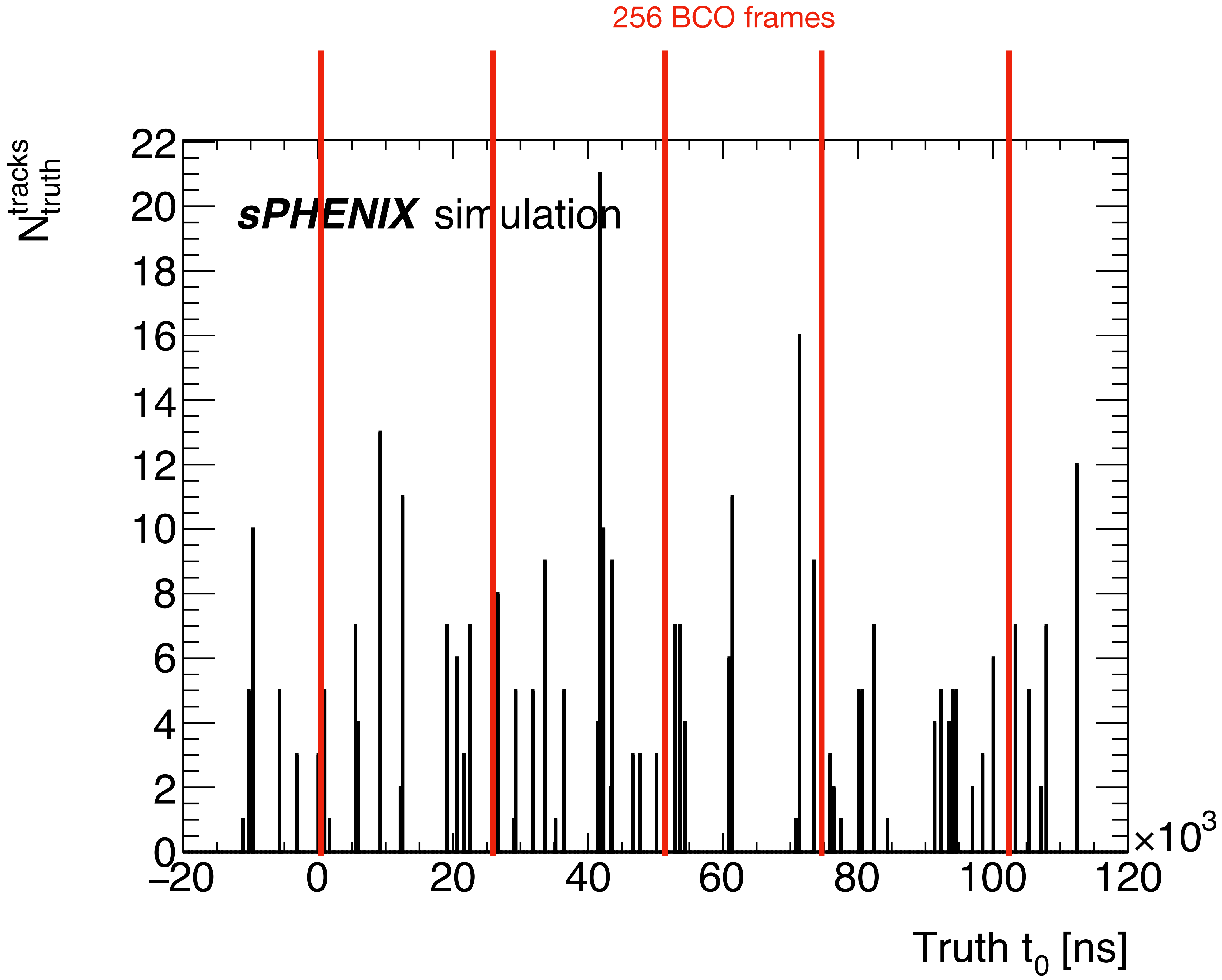
4D Tracking with Streaming Data

Joe Osborn

February 1, 2024

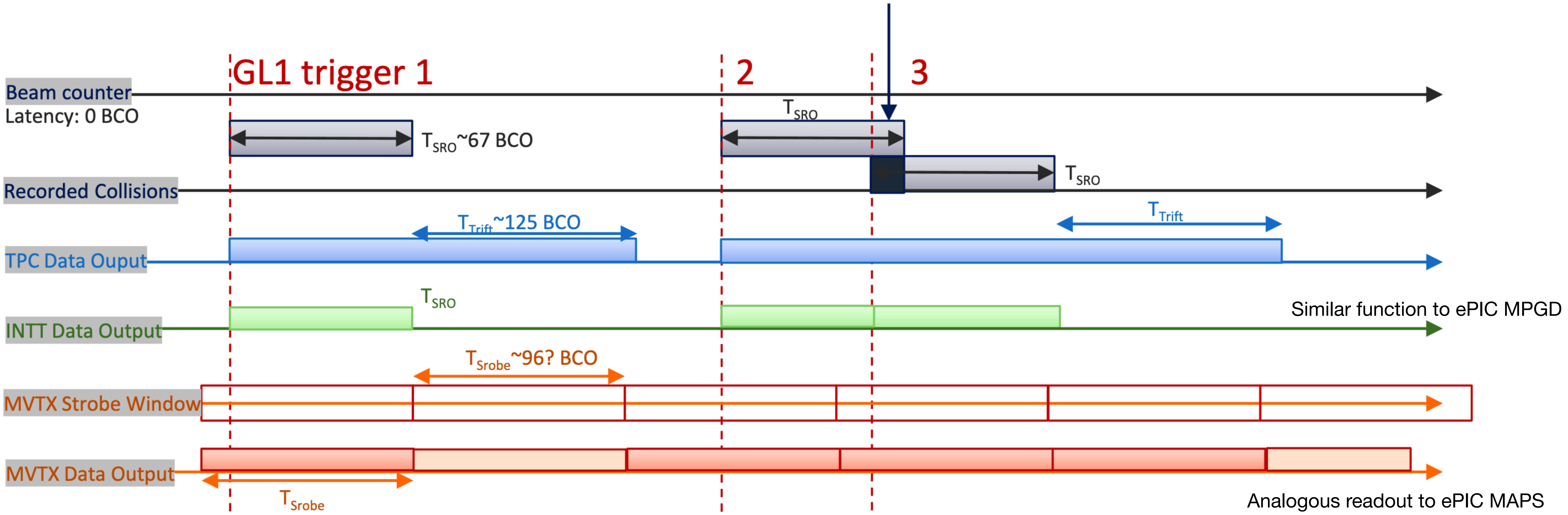
Streaming Tracking

- Redefine terms:
  - Event == a physics event corresponding to a bunch crossing where an interaction happened, producing particles
  - Trigger/time frame == a chunk of streamed data in time where reconstruction is performed
- Time frame building and reconstruction has to process data in time frames, considering data at time frame boundaries and any duplicated data
- Output of reconstruction is tracks assigned to vertices



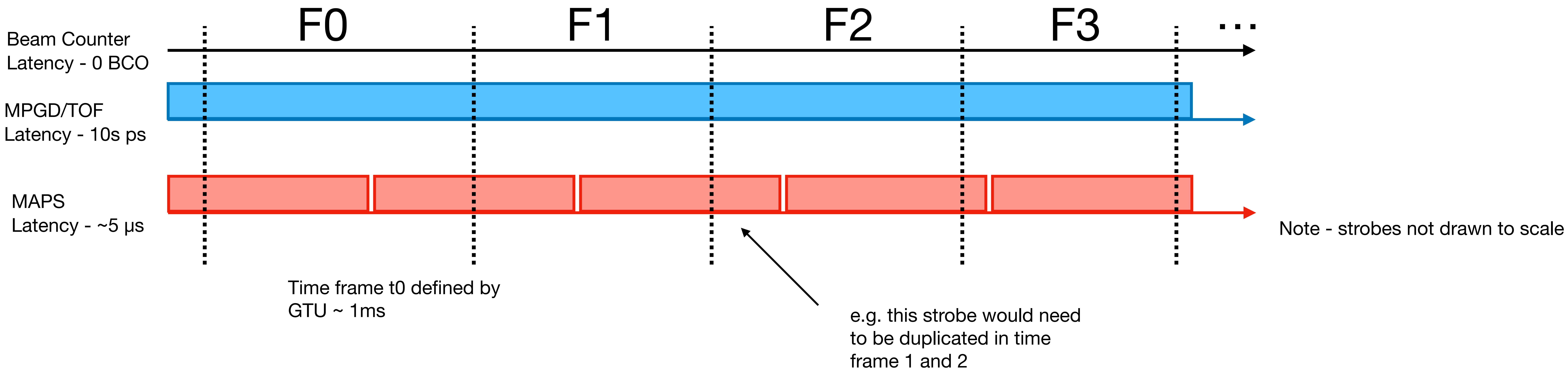
- Example of 100 μs free streaming in sPHENIX environment, to be divided into ~26 μs time frames
- ePIC SRO plans for building ~1 ms time frames sorted in data files

Streaming Tracking



- Necessary to synchronize all subsystems in time
- Reconstruction is presented with all tracks and vertices from a time/trigger frame
- Time/trigger frame is defined by various subsystem readouts (usually whichever is slowest)

Streaming Tracking



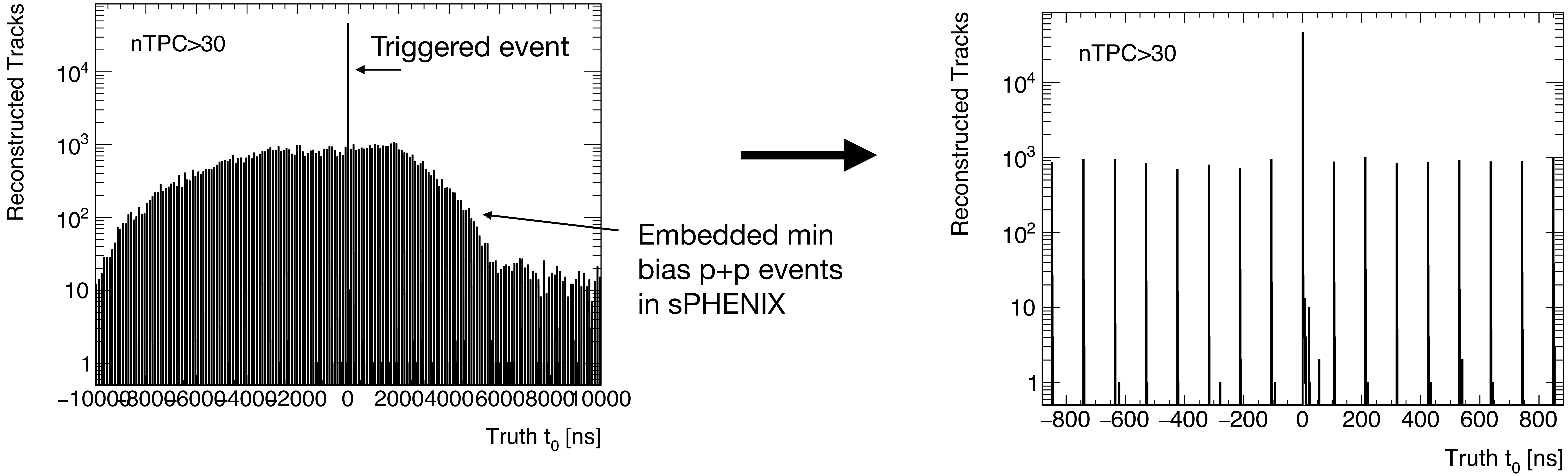
Note - strobes not drawn to scale

Time frame t0 defined by GTU ~ 1ms

e.g. this strobe would need to be duplicated in time frame 1 and 2

- ePIC will be 100% streaming, so we won’t have a hardware trigger + extended readout
    - Simpler case, because every frame can be treated the same
- Chunk data into ~1ms time frames, where readout windows will in general not coincide with time frames
- Requires duplication of some data at frame boundaries

Zeroth Order Tracking with Timing



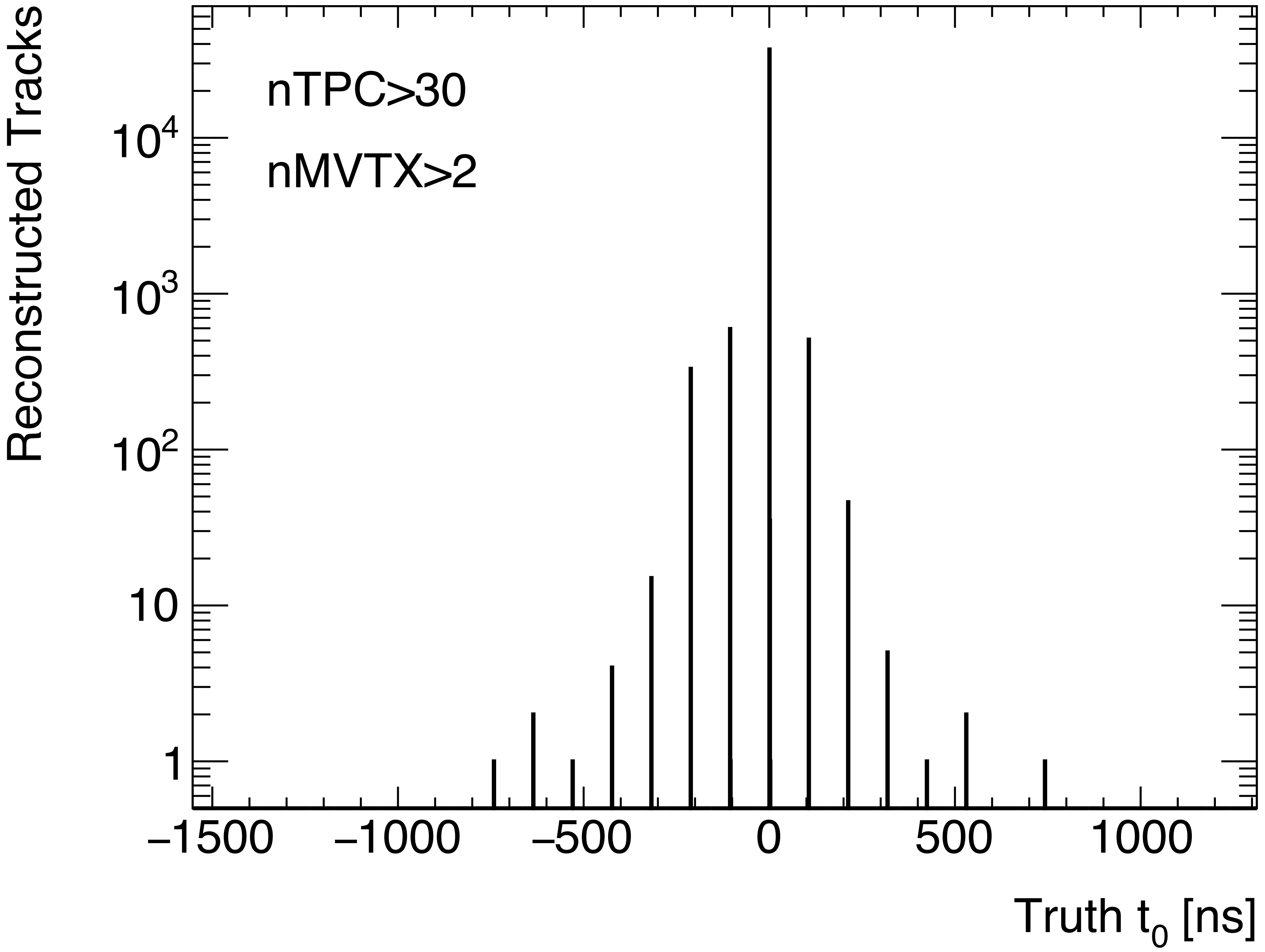
Triggered event

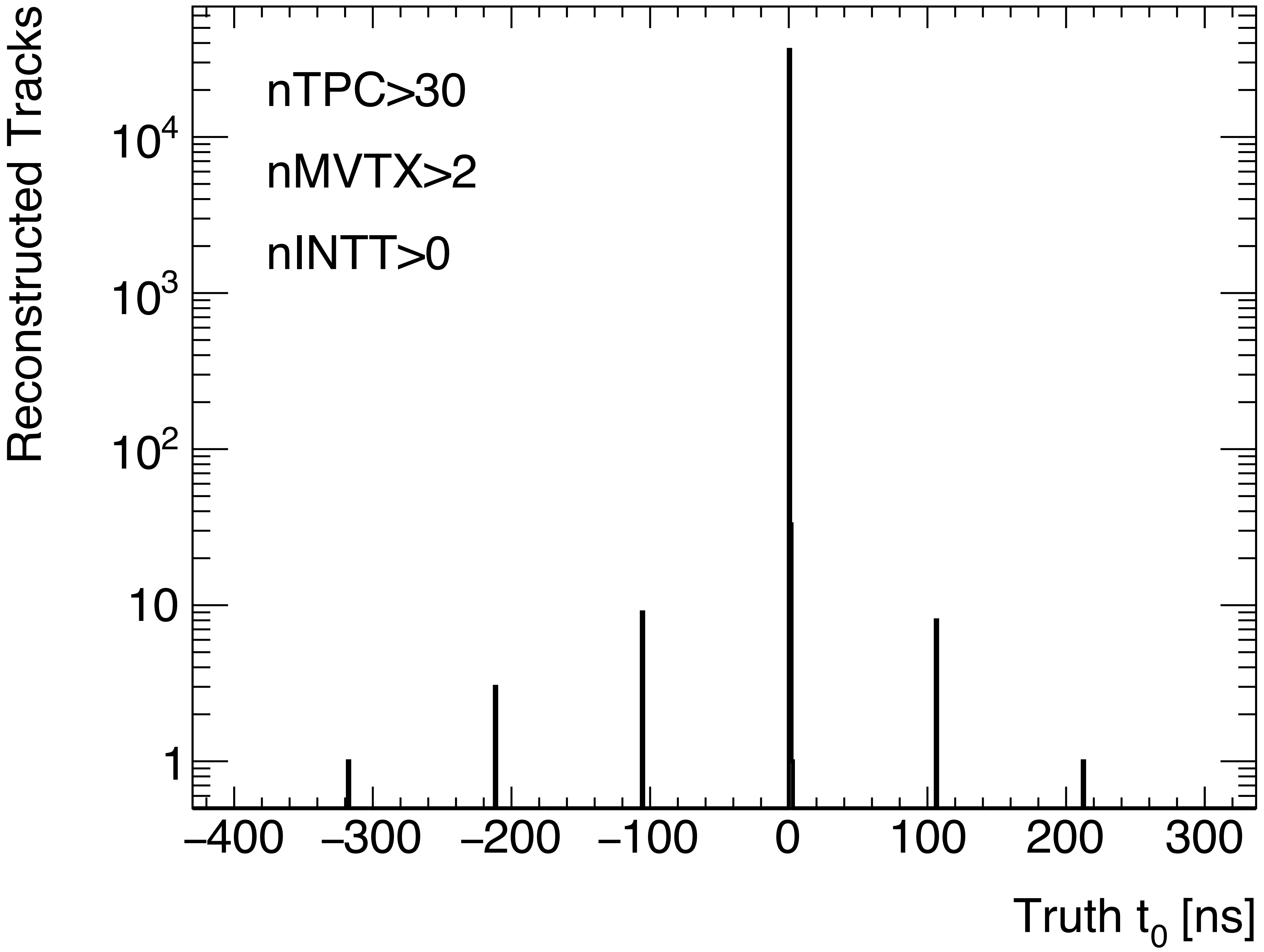
Embedded min bias p+p events in sPHENIX

- Looking at tracks with only TPC clusters is sensitive to full RHIC 106 ns bunch structure

- ePIC should see similar behavior except 1/98.5MHz ~ 10 ns

Zeroth Order Tracking with Timing



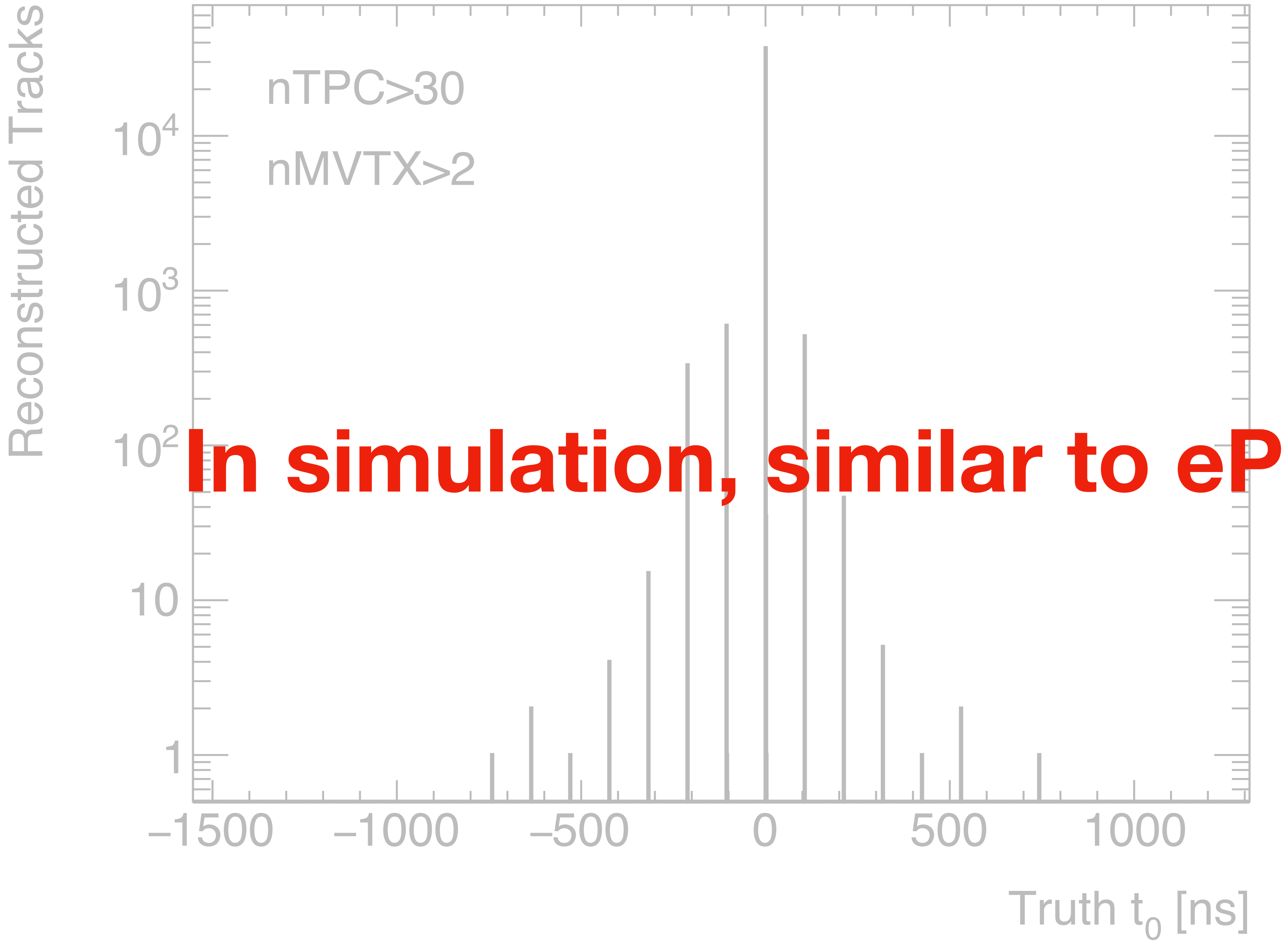


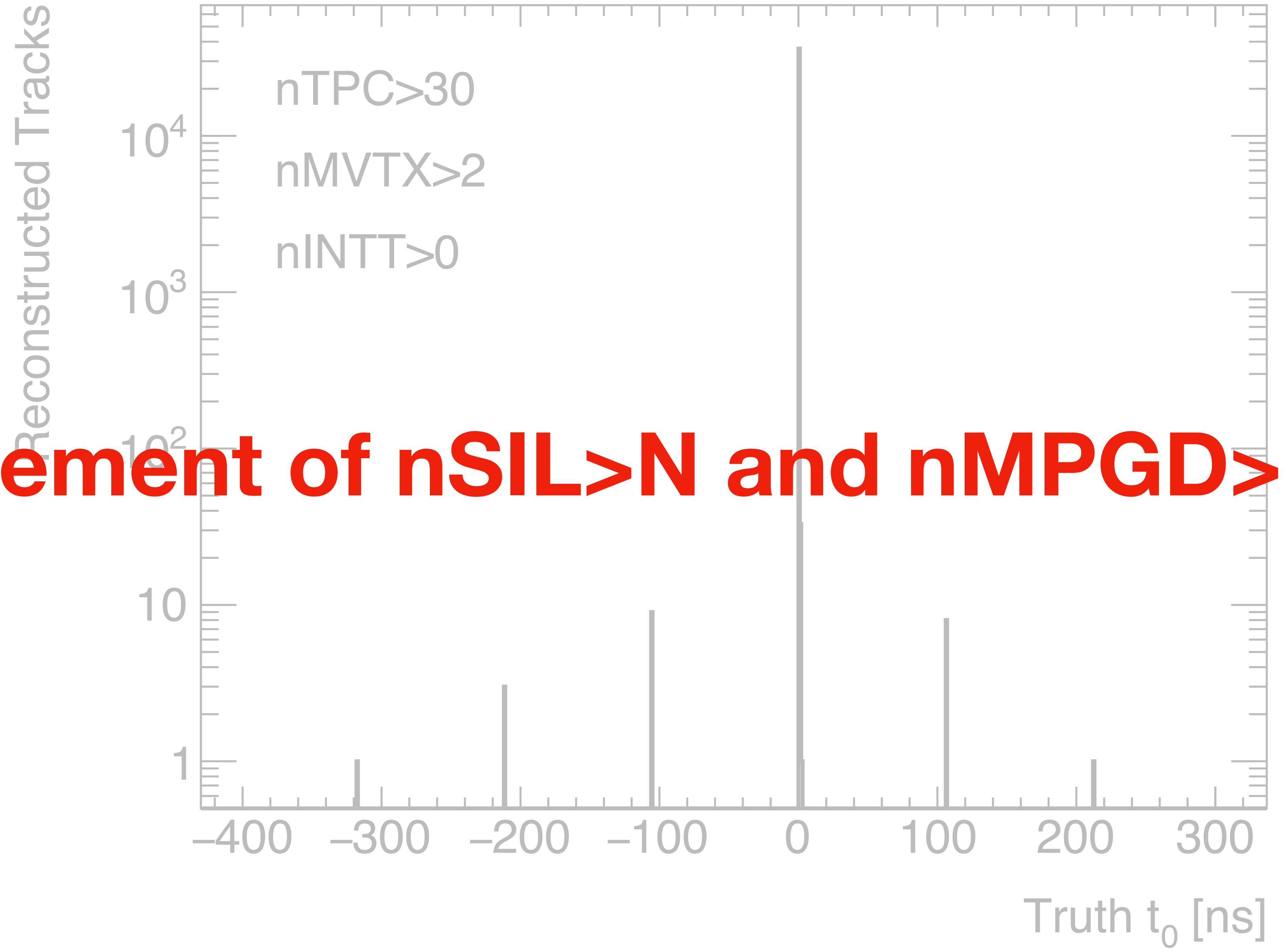
- Requiring clusters on track with smaller timing window limits to the triggered bunch crossing

- Simply a result of “artificially” matching timing windows of various subsystems

- This only gets you so far - e.g. doesn’t account for dead/hot areas (so timing in the hits/tracking is ultimately necessary). But in a simulation, it can already show pile up rejection power

Zeroth Order Tracking with Timing





In simulation, similar to ePIC requirement of nSIL>N and nMPGD>0

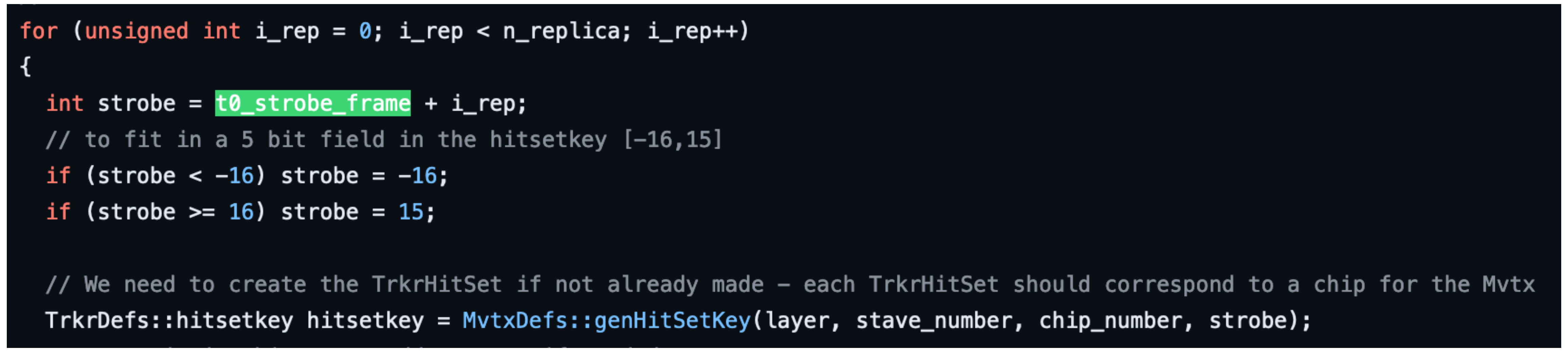
- Requiring clusters on track with smaller timing window limits to the triggered bunch crossing
- Simply a result of “artificially” matching timing windows of various subsystems
- This only gets you so far - e.g. doesn’t account for dead/hot areas (so timing in the hits/tracking is ultimately necessary). But in a simulation, it can already show pile up rejection power

Integrating Timing to Tracking

- Upon hit creation, construct unique surface identifier that carries both spatial (channel) and timing (crossing) information

- Examples:

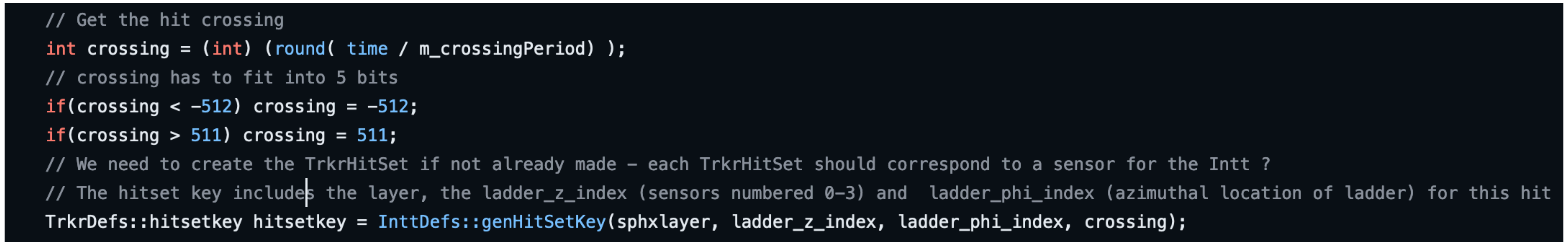
- MVTX digitization - determine strobe window relative to triggered crossing



```
for (unsigned int i_rep = 0; i_rep < n_replica; i_rep++)
{
  int strobe = t0_strobe_frame + i_rep;
  // to fit in a 5 bit field in the hitsetkey [-16,15]
  if (strobe < -16) strobe = -16;
  if (strobe >= 16) strobe = 15;

  // We need to create the TrkrHitSet if not already made - each TrkrHitSet should correspond to a chip for the Mvtx
  TrkrDefs::hitsetkey hitsetkey = MvtxDefs::genHitSetKey(layer, stave_number, chip_number, strobe);
```

- INTT digitization - determine crossing value based on rate and time G4hit was created



```
// Get the hit crossing
int crossing = (int) (round( time / m_crossingPeriod) );
// crossing has to fit into 5 bits
if(crossing < -512) crossing = -512;
if(crossing > 511) crossing = 511;
// We need to create the TrkrHitSet if not already made - each TrkrHitSet should correspond to a sensor for the Intt ?
// The hitset key includes the layer, the ladder_z_index (sensors numbered 0-3) and  ladder_phi_index (azimuthal location of ladder) for this hit
TrkrDefs::hitsetkey hitsetkey = InttDefs::genHitSetKey(sphxlayer, ladder_z_index, ladder_phi_index, crossing);
```

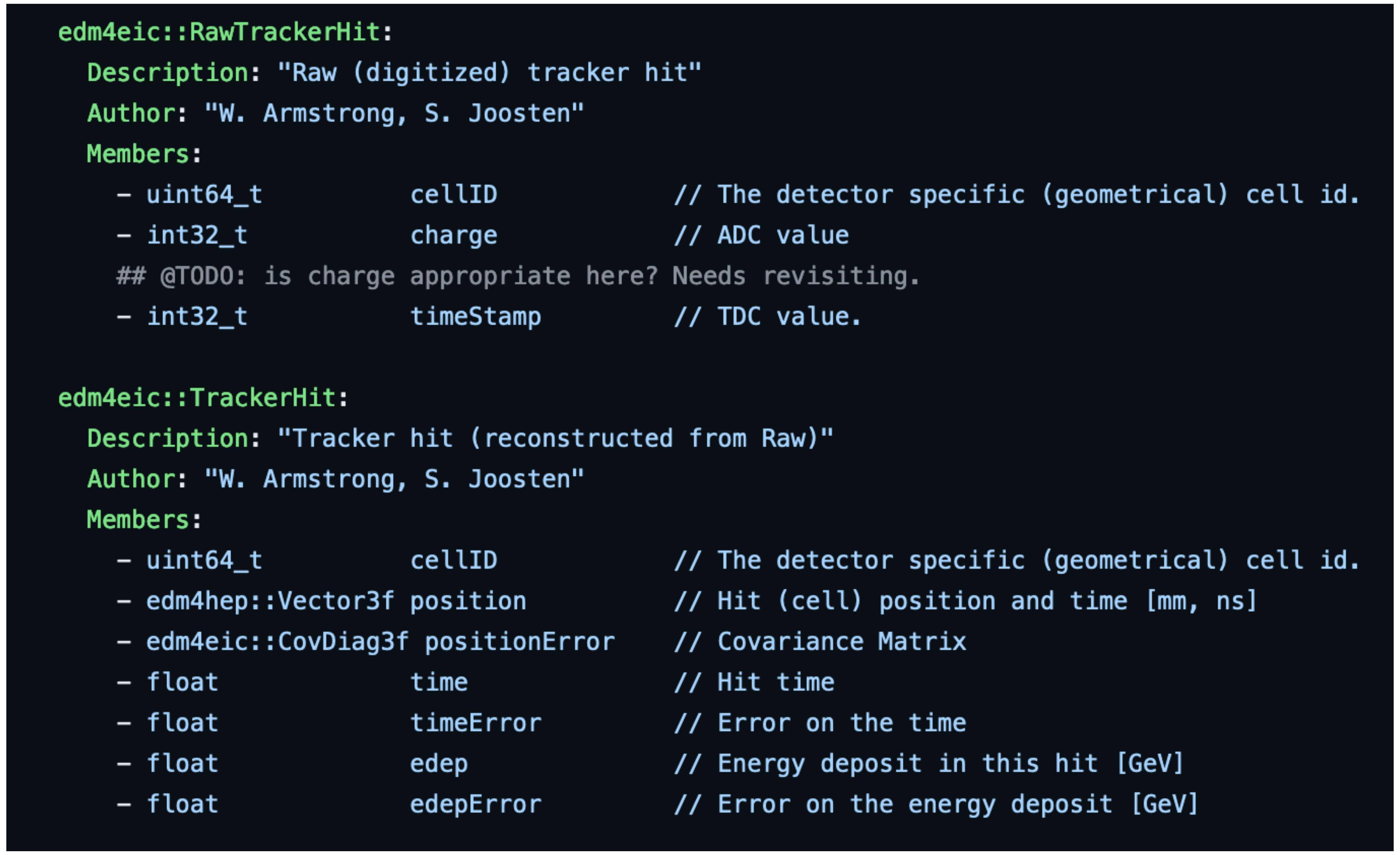
Integrating Timing into Tracking

- Currently hits have a time component, so the information could be stored in digitization

- As far as I could tell a smeared time is already stored in the hit digitization

- Question for discussion

    - For ePIC, does each hit need their own time, or do we want to group hits into structures defined by the readout (channel + timing window)?



```
edm4eic::RawTrackerHit:
  Description: "Raw (digitized) tracker hit"
  Author: "W. Armstrong, S. Joosten"
  Members:
    - uint64_t          cellID            // The detector specific (geometrical) cell id.
    - int32_t           charge            // ADC value
    ## @TODO: is charge appropriate here? Needs revisiting.
    - int32_t           timeStamp         // TDC value.

edm4eic::TrackerHit:
  Description: "Tracker hit (reconstructed from Raw)"
  Author: "W. Armstrong, S. Joosten"
  Members:
    - uint64_t          cellID            // The detector specific (geometrical) cell id.
    - edm4hep::Vector3f position          // Hit (cell) position and time [mm, ns]
    - edm4eic::CovDiag3f positionError    // Covariance Matrix
    - float             time              // Hit time
    - float             timeError         // Error on the time
    - float             edep              // Energy deposit in this hit [GeV]
    - float             edepError         // Error on the energy deposit [GeV]
```

Track Reconstruction with Timing

- Need some global timing with which to correlate hits to each other
- For ePIC this will likely be the start of the time frame given by the GTU
  - Can assign MAPS hits to relative strobe with respect to GTU GL1
  - Can assign MPGD/TOF hits to relative crossing number with respect to GTU GL1
- Assign tracks an estimated crossing number based on their composition

Track Reconstruction with Timing

- In sPHENIX we match TPC and silicon tracklets in eta/phi/PCA_{xy}
- From those matches, determine whether or not they match in crossing and PCA_z based on the TPC drift velocity
- In ePIC could imagine doing something similar with the MPGD/TOF
    - It may be enough to look at χ^2 contribution given by MPGD hits to overall track. For out of time silicon+MPGD matches, the χ^2 contribution should be nominally much larger for the MPGD clusters than the silicon clusters (if out of time match)
    - The Acts determined χ^2 is a very good discriminator for identifying which track match is “self-consistent”
    - e.g. χ^2 in some dummy tests increases by factor of 5-10 for +/-2 crossings from the correctly matched crossing

Final Thoughts

- Not possible to separate reconstruction from time frame
- Not uniformly possible to build real physics events, depending on what type of event you are looking at
    - Example - is some track a highly displaced track from in time primary vertex or a primary track from a primary vertex 3 beam crossings away?
        - Needs to be dealt with at analysis level and not necessarily reconstruction - depends on whether or not looking for some HF decay or not
- Streaming with MAPS involves some data duplication - no way around this since hits in a strobe window can belong to either time frame when the time frame boundary slices a strobe window
- TPC is inherently different because time == z coordinate, so buys some timing discrimination "for free"
    - To get the same discriminating power out of Acts, we will need to include time in Kalman Filter

1. Reject background by requiring fast timing detector cluster on track
2. Include time bin in offline channel ID and build tracks with a crossing estimate, associating a track with a beam crossing relative to some GTU global time (frame start)
3. Include time in Acts track fit - should make χ^2 discrimination even more obvious - not sure if anyone in Acts community has actually done this yet