Geant4
Digitization
Reconstruction

A step by step consideration

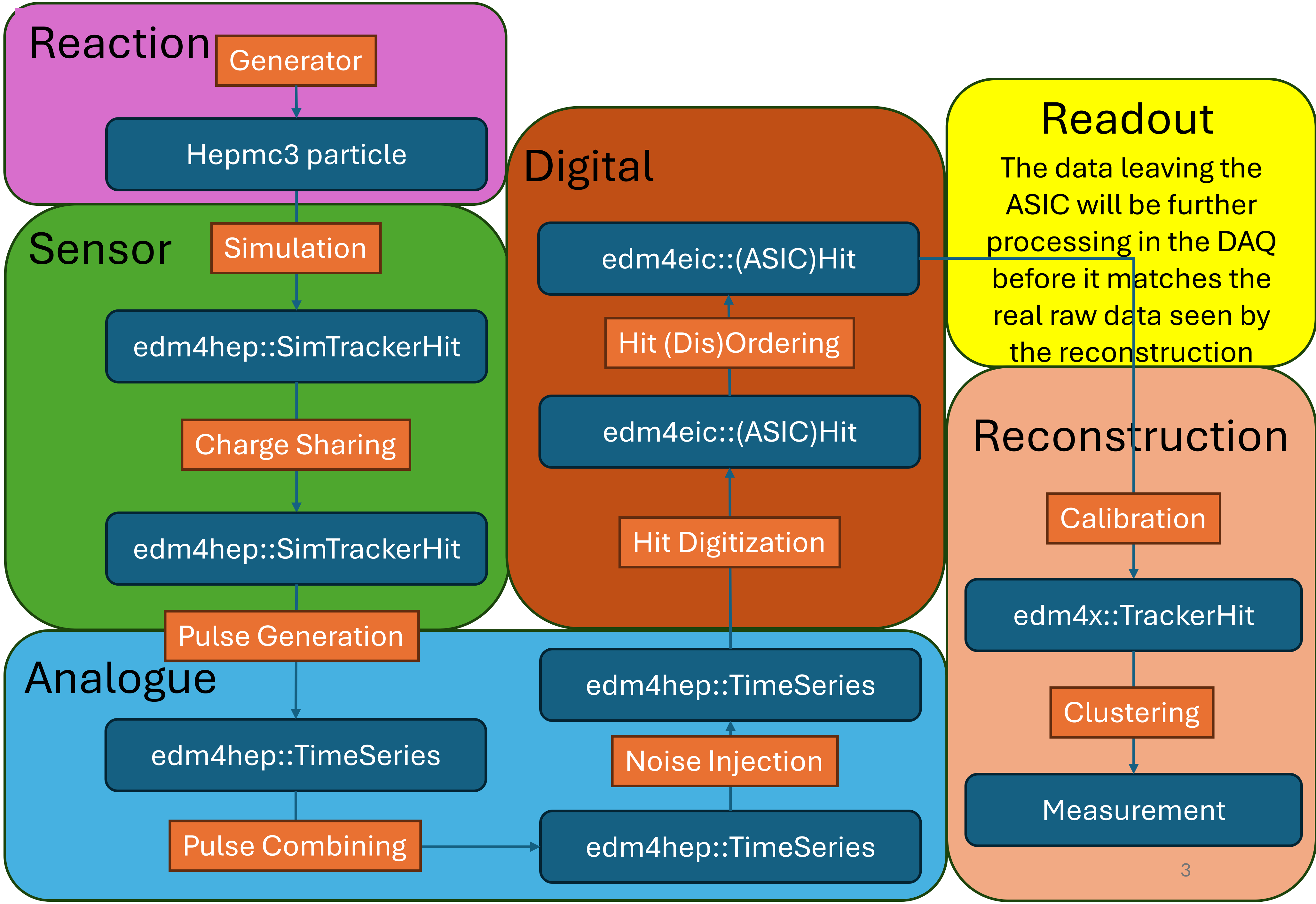
(From a Timepix4/Low-Q^2 Tagger perspective)

Simon Gardner

19/03/2025

Low-Q^2 Tagger and Timepix4

- A realistic digitization scheme and background model are perhaps more important for the Low-Q^2 Tagger than other detectors at this point.
    - Regions of phase space might be impossible to cover at certain beam conditions.
- High Bremsstrahlung rates increasing with Z^2 of ion.
    - Unavoidable overlap with Low-Q2 physics electrons.
- High Synchrotron backgrounds
    - Highly concerning but design of beampipe, magnets and exit window can mitigate the problem.
- **The Timepix4 ASIC already exists and is well understood so is a good example to build from.**
- Related work on MC/ML based digitization using Allpix2 is also being developed.



Reaction

Generator

Hepmc3 particle

Sensor

Simulation

edm4hep::SimTrackerHit

Charge Sharing

edm4hep::SimTrackerHit

Pulse Generation

Analogue

edm4hep::TimeSeries

Pulse Combining

edm4hep::TimeSeries

Noise Injection

edm4hep::TimeSeries

Digital

Hit Digitization

edm4eic::(ASIC)Hit

Hit (Dis)Ordering

edm4eic::(ASIC)Hit

Readout

The data leaving the ASIC will be further processing in the DAQ before it matches the real raw data seen by the reconstruction

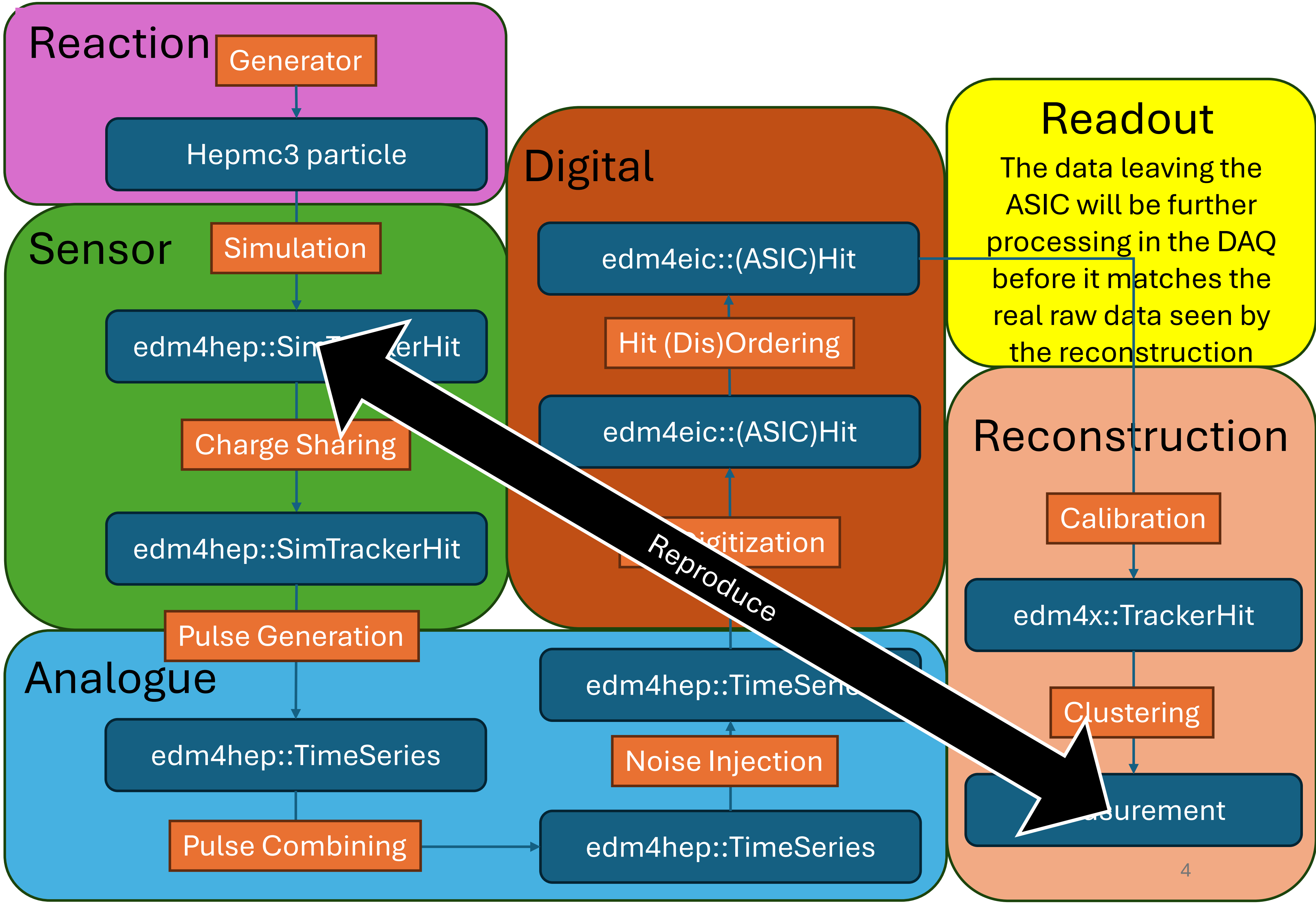
Reconstruction

Calibration

edm4x::TrackerHit

Clustering

Measurement



Reaction

Generator

Hepmc3 particle

Sensor

Simulation

edm4hep::SimT erHit

Charge Sharing

edm4hep::SimTrackerHit

Pulse Generation

Analogue

edm4hep::TimeSeries

Pulse Combining

edm4hep::TimeSeries

Noise Injection

edm4hep::TimeSeri

Digital

igitization

edm4eic::(ASIC)Hit

Hit (Dis)Ordering

edm4eic::(ASIC)Hit

Readout

The data leaving the ASIC will be further processing in the DAQ before it matches the real raw data seen by the reconstruction

Reconstruction

Calibration

edm4x::TrackerHit

Clustering

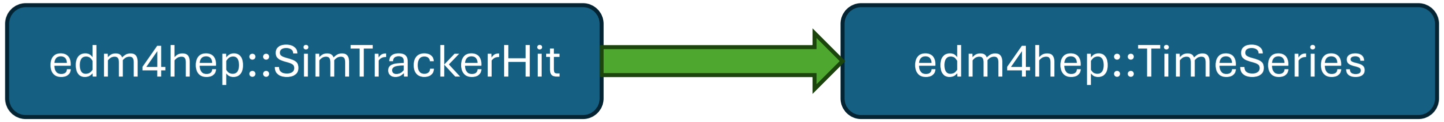
surement

Reproduce

What is a edm4hep::SimTrackerHit from dd4hep?

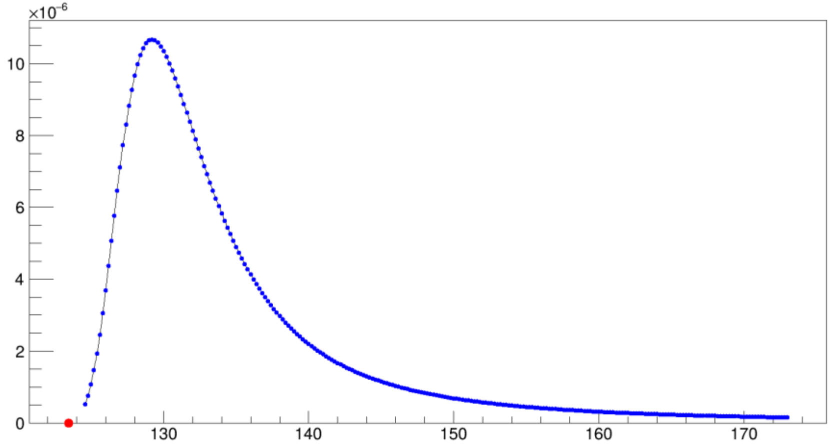
- A sim tracker hit is recorded at a weighted central point of particle steps through the sensitive detector.
- Usually only a single hit will be recorded.
- Multiple hits in the same cell will occur when another particle takes a step in the detector.
    - When secondaries created near/in the element summing of signals to reproduce a realistic size is important
- Some detectors with hot spots might see multiple hits in a time frame from physics or other backgrounds
    - Having a realistic digitization which will demonstrate the detectors' ability to separate the hits is important.

Generic Pulse Generation

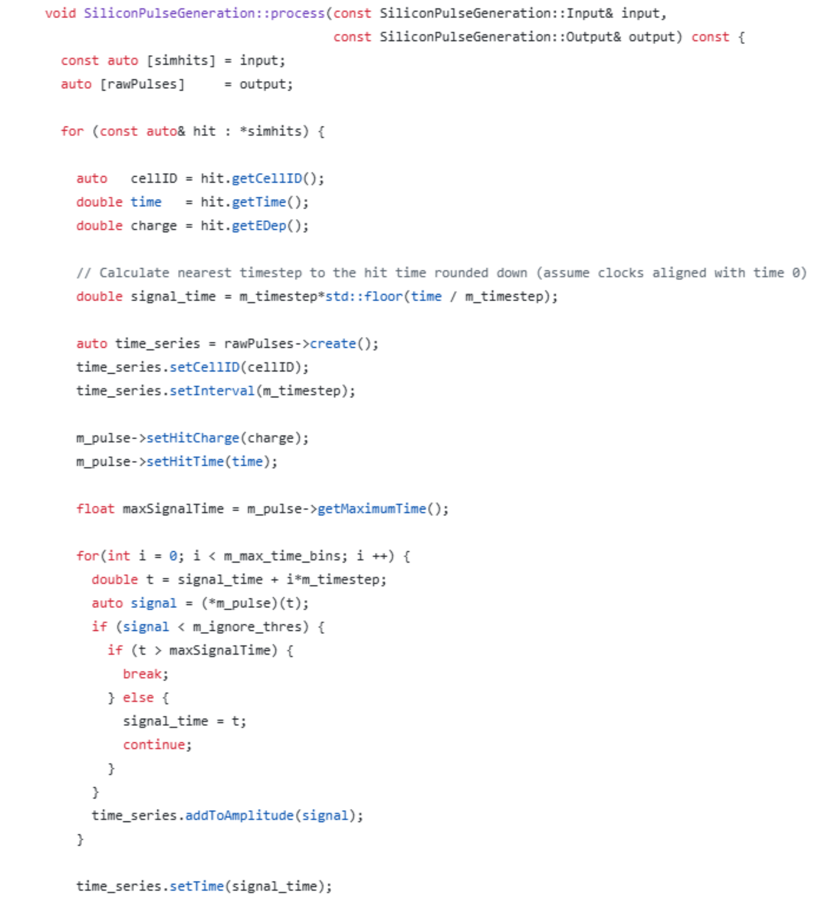

edm4hep::SimTrackerHit

edm4hep::TimeSeries

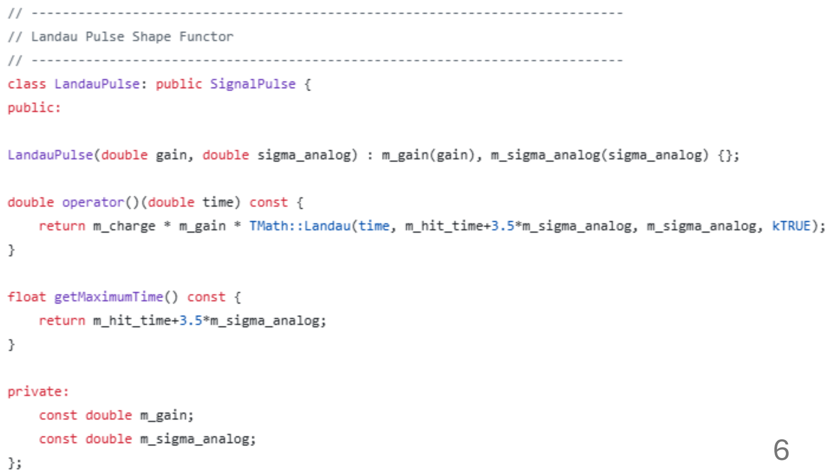
Hit (red point) time and pulse (blue).



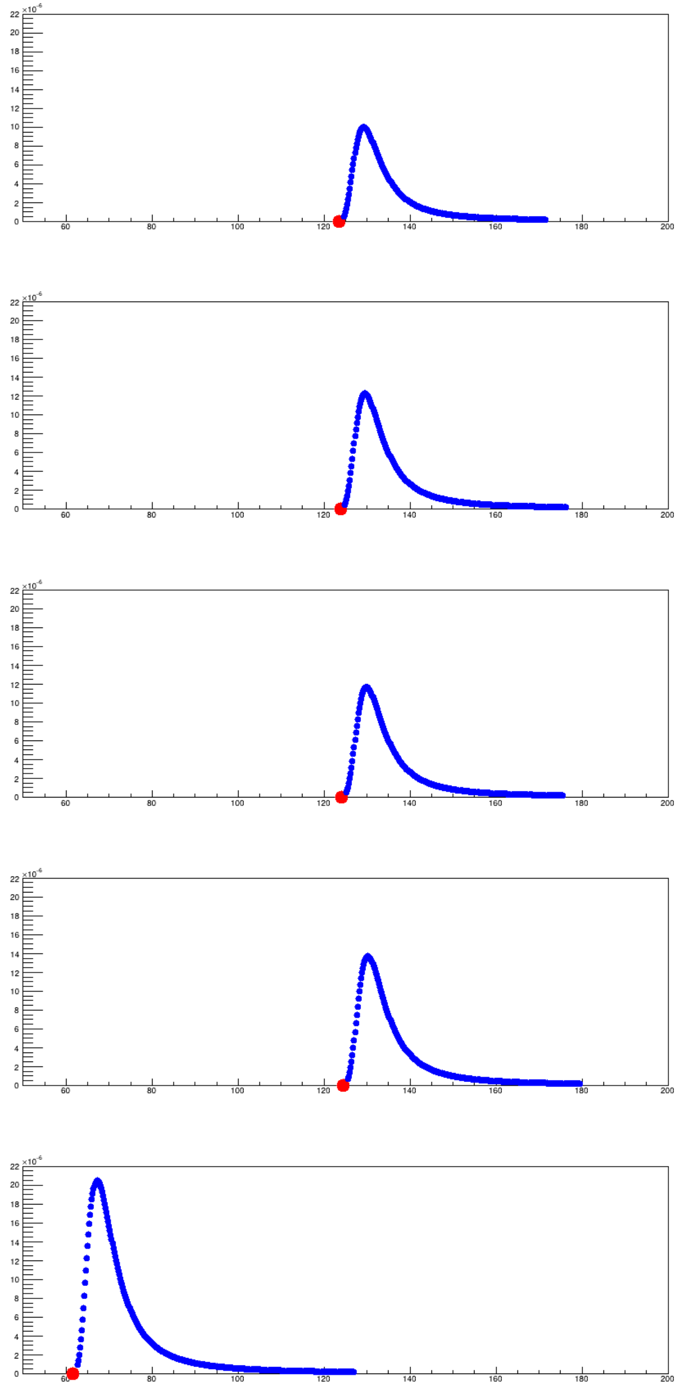
- Pulse shape parameterized by hit time and energy
- Pulse step needs to be equal to or smaller than fastest clock.
- Minimum pulse Threshold needs to be below any digitization threshold.
- More complex pulse functions could use position in cell and vector.
- [EICrecon/src/algorithms/digi/SiliconPulseGeneration.cc at Pulse-Noise · eic/EICrecon]



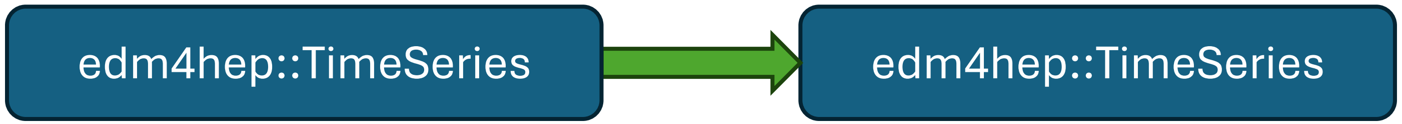
```
void SiliconPulseGeneration::process(const SiliconPulseGeneration::Input& input,
                                     const SiliconPulseGeneration::Output& output) const {
  const auto [simhits] = input;
  auto [rawPulses]     = output;

  for (const auto& hit : *simhits) {

    auto   cellID = hit.getCellID();
    double time   = hit.getTime();
    double charge = hit.getEDep();

    // Calculate nearest timestep to the hit time rounded down (assume clocks aligned with time 0)
    double signal_time = m_timestep*std::floor(time / m_timestep);

    auto time_series = rawPulses->create();
    time_series.setCellID(cellID);
    time_series.setInterval(m_timestep);

    m_pulse->setHitCharge(charge);
    m_pulse->setHitTime(time);

    float maxSignalTime = m_pulse->getMaximumTime();

    for(int i = 0; i < m_max_time_bins; i ++) {
      double t = signal_time + i*m_timestep;
      auto signal = (*m_pulse)(t);
      if (signal < m_ignore_thres) {
        if (t > maxSignalTime) {
          break;
        } else {
          signal_time = t;
          continue;
        }
      }
      time_series.addToAmplitude(signal);
    }

    time_series.setTime(signal_time);
```



```
// ---------------------------------------------------------------------------
// Landau Pulse Shape Functor
// ---------------------------------------------------------------------------
class LandauPulse: public SignalPulse {
public:

LandauPulse(double gain, double sigma_analog) : m_gain(gain), m_sigma_analog(sigma_analog) {};

double operator()(double time) const {
    return m_charge * m_gain * TMath::Landau(time, m_hit_time+3.5*m_sigma_analog, m_sigma_analog, kTRUE);
}

float getMaximumTime() const {
    return m_hit_time+3.5*m_sigma_analog;
}

private:
    const double m_gain;
    const double m_sigma_analog;
};
```

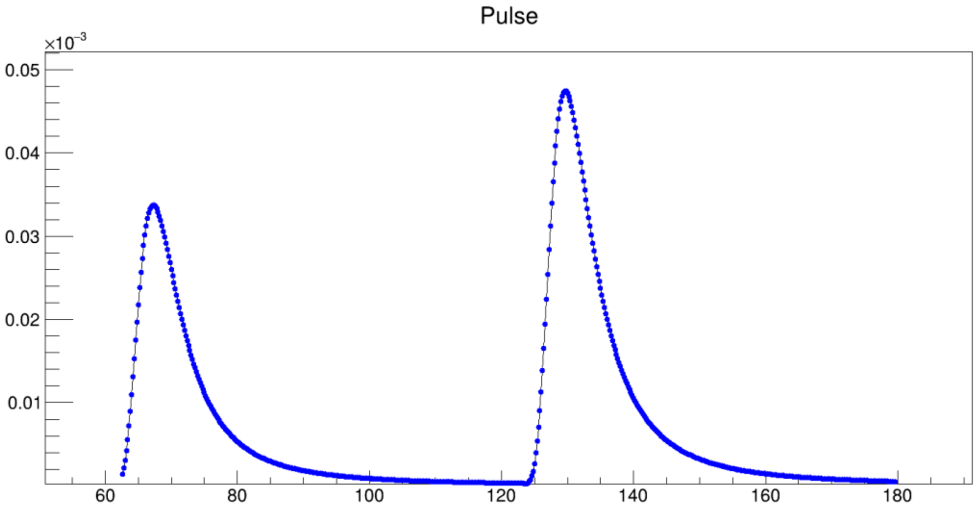


Time Series Merging



edm4hep::TimeSeries

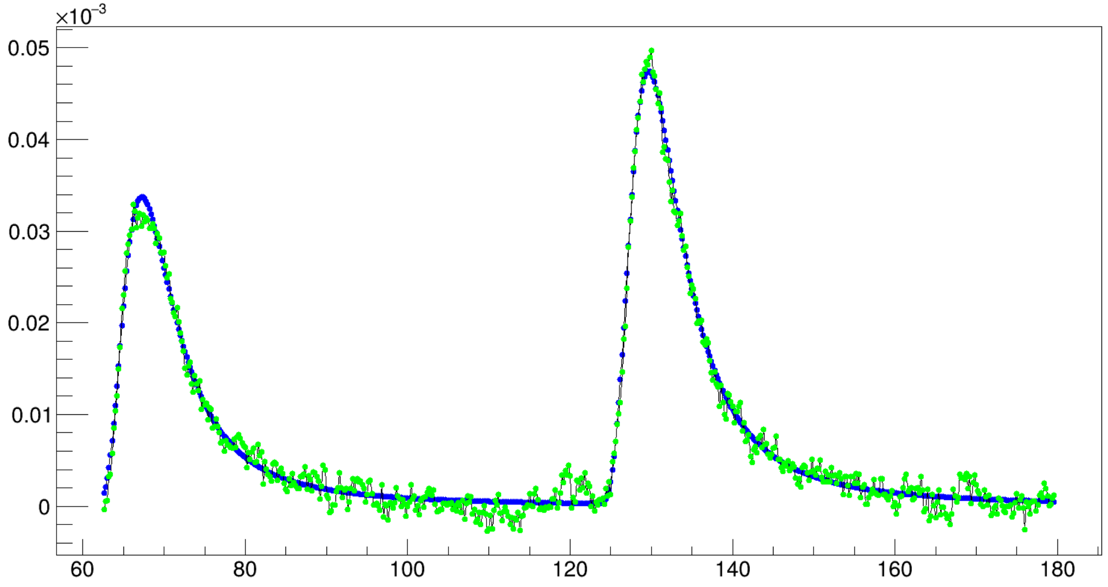
edm4hep::TimeSeries

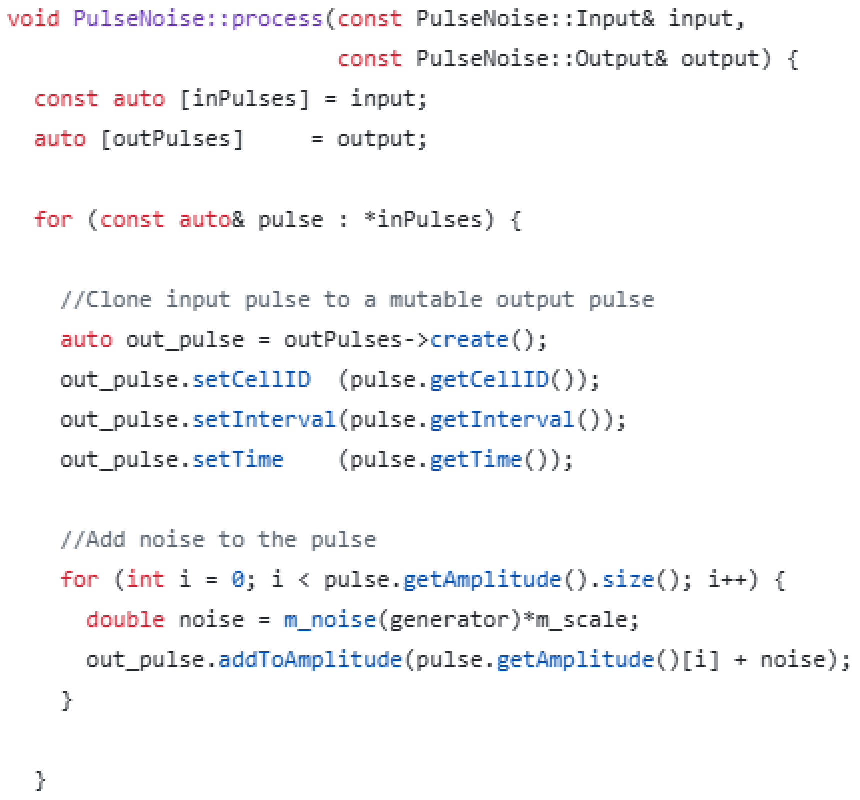


- Time series summed together when the end of one pulse is within x time of the start of the next.
- x needs to be the maximum time a signal can influence another
  - This is guided by the slower clocks responsible for readout of the data.
- Linear summing of pulses - could extend to include non-linear responses.
- EICrecon/src/algorithms/digi/PulseCombiner.cc at Combine-pulses · eic/EICrecon
- Branch ready for PR...

Noise Injection

edm4hep::TimeSeries
edm4hep::TimeSeries

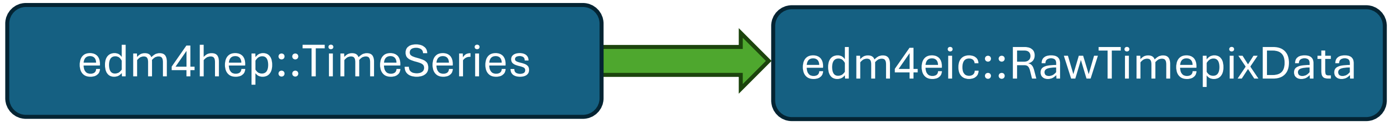




```
void PulseNoise::process(const PulseNoise::Input& input,
                         const PulseNoise::Output& output) {
  const auto [inPulses] = input;
  auto [outPulses]      = output;

  for (const auto& pulse : *inPulses) {

    //Clone input pulse to a mutable output pulse
    auto out_pulse = outPulses->create();
    out_pulse.setCellID  (pulse.getCellID());
    out_pulse.setInterval(pulse.getInterval());
    out_pulse.setTime    (pulse.getTime());

    //Add noise to the pulse
    for (int i = 0; i < pulse.getAmplitude().size(); i++) {
      double noise = m_noise(generator)*m_scale;
      out_pulse.addToAmplitude(pulse.getAmplitude()[i] + noise);
    }

  }
```

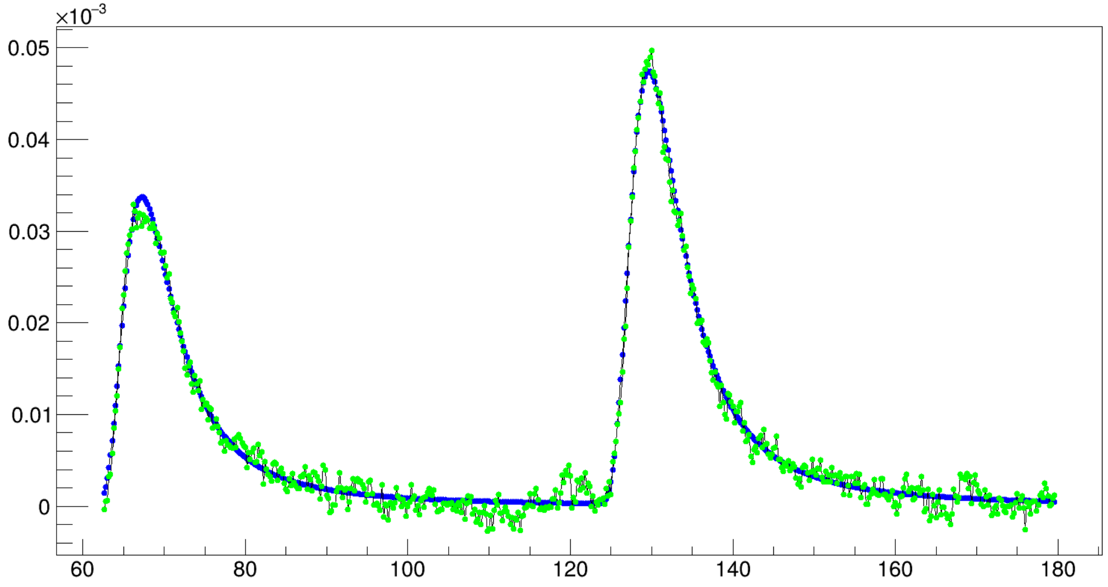
- Noise injection into the pulse across frequencies, provided by DDDigi: DD4hep/DDDigi/src/noise/FalphaNoise.cpp at master · AIDASoft/DD4hep
- Will not add independent noise hits.

ASIC specific digitization (Timepix4)



edm4hep::TimeSeries

edm4eic::RawTimepixData



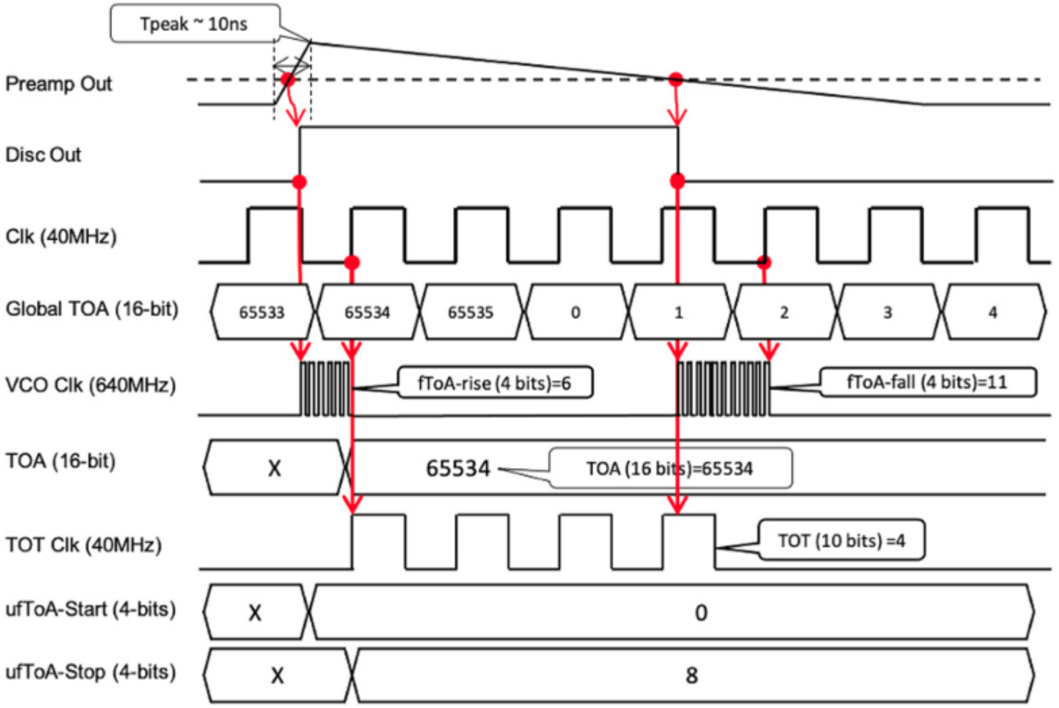
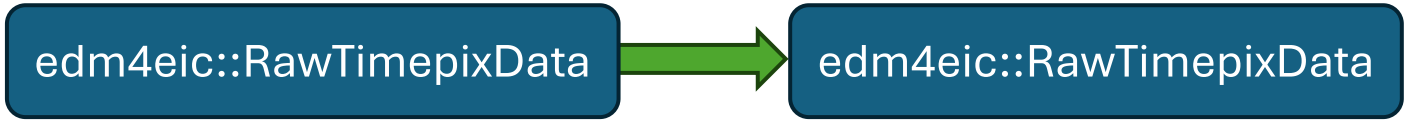


Figure 4. Timing diagram for the Timepix4 pixel cell in data driven mode.

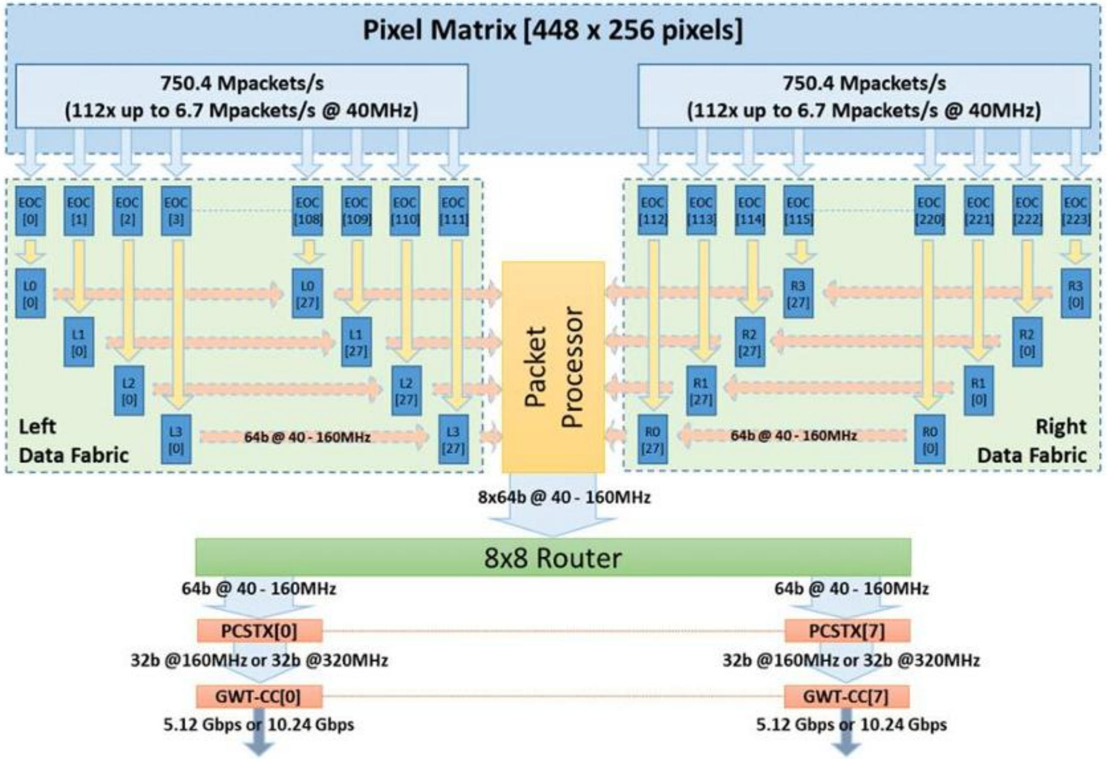
Timepix4 - X. Llopart et al 2022

- Implementation not ready but relatively simple
- Basically overlays clocks onto the pulse to determine ToA, ToT, fToA, ufToA.
- Should encode into Timepix4 64bit readout

ASIC specific Hit ordering



- The order of the hits in the datastream will not be ordered by ToA
  - E.g. Hits with a longer ToT will take longer to be processed
  - Hits in different areas of the ASIC take preference.
  - Sorting has to be done to some degree at some point in the DAQ so understanding how the events are disordered is important.,



Timepix4 - X. Llopart et al 2022