SiStrip Simulation overview

Claude Nuttens

claude.nuttens@uclouvain.be

Université catholique de Louvain
Center for Cosmology, Particle Physics and Phenomenology

Special meeting 13-05-2014

Outline

- Links & docs
- The job
- SiStrip Digitization steps overview
- SiStrip Digitizer software structure
- The first project
- Some Details on modules

Links & doc

- Your future kingdom the « SiStripDigitizer »
  https://github.com/cms-sw/cmssw/tree/CMSSW_7_0_X/SimTracker/SiStripDigitizer
- Subscribe to
  hn-cms-trackerperformance@cern.ch
- My personnal documentation:
  http://cp3.irmp.ucl.ac.be/~cnuttens/Docs/tmp/.tmp/SiStrip/
- SiStrip Simulation, LocalReconstruction and
  Calibration page:
  https://twiki.cern.ch/twiki/bin/viewauth/CMS/SiStripCalibration
- SiStripDigitization wiki:
  https://twiki.cern.ch/twiki/bin/view/CMSPublic/SWGuideStripDigitization?redirectedfrom=CMS.SWGuideStripDigitization

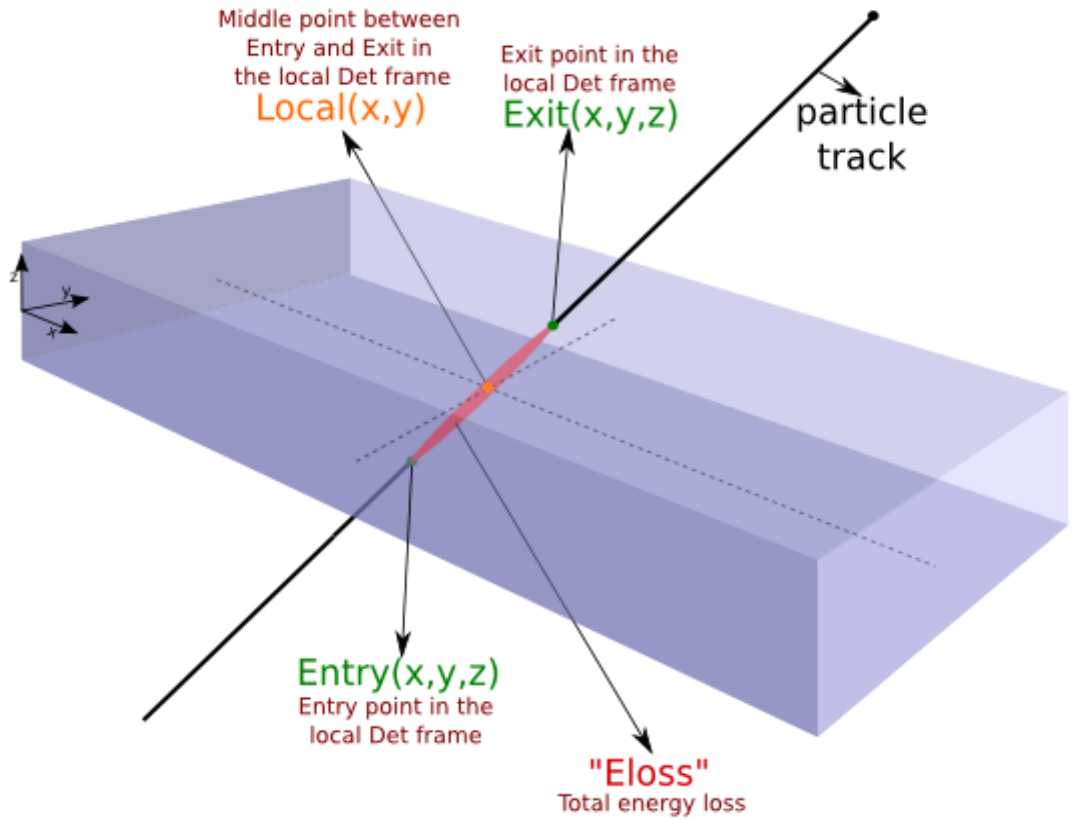
The Strip Sim contact Job

- Be aware of what happened to the SiStripDigitizer
- Be aware on things that could touch the sim in Strip
- Be the interface between the offline sim group and the Tracker group
- Keep an eye on the Relval on MC (not your job but check the status report)

Strip Digitizer Structure

From Geant4

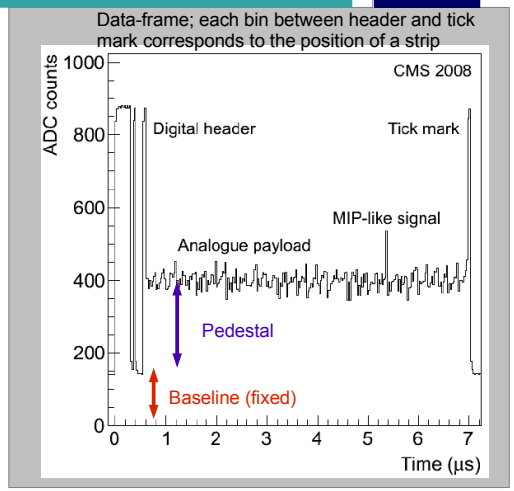
Variables in SimHits validation (Variables from (P)SimHits)



+Tof: ToF in nanoseconds from the primary interaction to the entry point. Always positive in a PSimHit, but may become negative in a SimHit due to bunch assignment

Strip signal simulation steps

- Standard mode of simulation is Zero Suppression (blue boxes)
- Heavy Ion run required Virgin Raw operation (blue+green)



Capacitive couplings tuned to cosmic data

OffDb

Probability tuned to test beam

Signal from Geant4 → Divide deposit → Cross-talk → Remove dead channels → APV Killer for HIP → Baseline shift → Baseline tilt

SiLinearCharge-Divider.cc

SiTrivialInduce-ChargeOnStrip.cc

SiStripDigitizerAlgorithm.cc

SiStripDigitizerAlgorithm.cc

SiGaussianTailNoiseAdder.cc

Shift and tilt being tuned on PbPb data in VR

ZS sim: N=a+b*length, with a,b fitted to data; VR sim: noise from data

CMN tuned to data in VR

Noise (N) → Gain (G); at APV level → Common mode noise → Pedestal → Digitization (incl. saturation) → To reco..

SiGaussianTailNoiseAdder.cc

SiTrivialDigitalConverter.cc

SiGaussianTailNoiseAdder.cc

SiGaussianTailNoiseAdder.cc

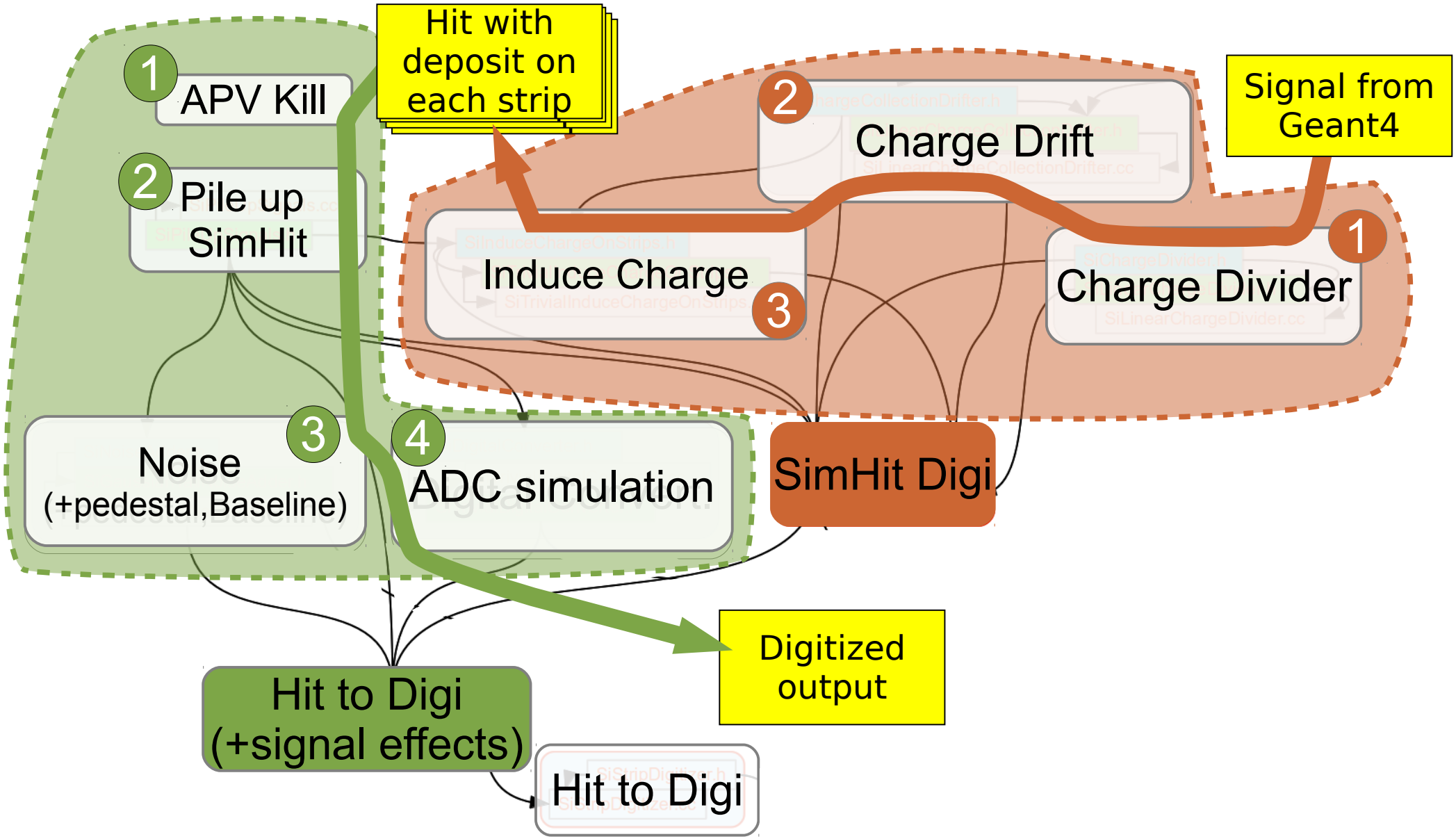
OffDb

OffDb

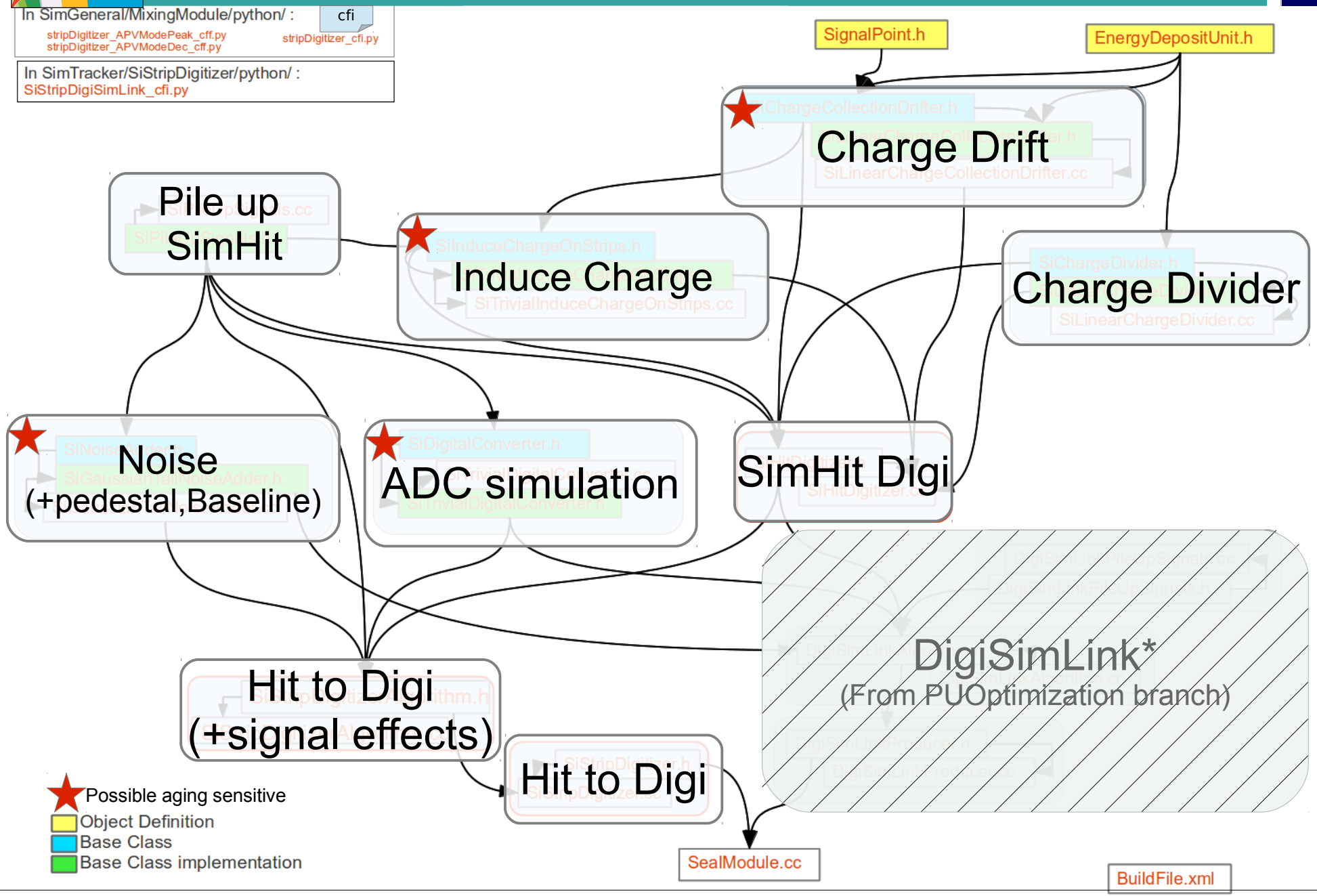
OffDb

*original diagram by Andrea Giammanco

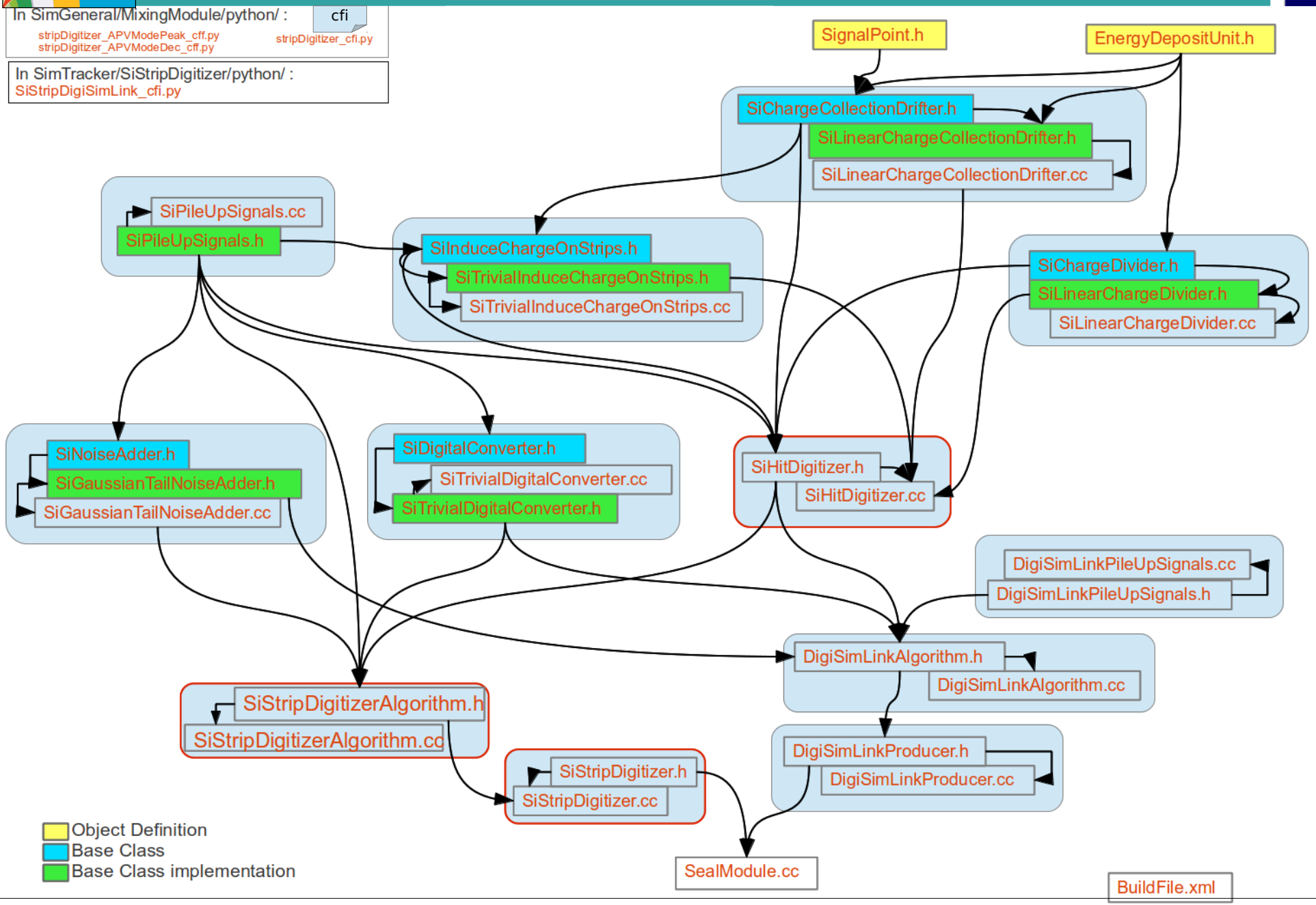
SiStrip Digitizer Steps



SiStrip Digitizer structure



SiStrip Digitizer structure



Strip Digitizer Parameters

stripDigitizer_cfi Parameters

In /SimGeneral/MixingModule/python/stripDigitizer_cfi.py

```
    accumulatorType = cms.string("SiStripDigitizer"),
    hitsProducer = cms.string('g4SimHits'),
```

SiLinearChargeDivider

```
    DeltaProductionCut      = cms.double(0.120425),
    APVpeakmode             = cms.bool(False),
             #also in SiStripDigitizerAlgorithm
    LandauFluctuations      = cms.bool(True),
    chargeDivisionsPerStrip = cms.int32(10),
    CosmicDelayShift        = cms.untracked.double(0.0),
             # also SiStripDigitizerAlgorithm
```

SiHitDigitizer

```
    DepletionVoltage        = cms.double(170.0),
    AppliedVoltage          = cms.double(300.0),
    ChargeMobility          = cms.double(310.0),
    Temperature             = cms.double(273.0),
    GevPerElectron          = cms.double(3.61e-09),
    ChargeDistributionRMS   = cms.double(6.5e-10),
    noDiffusion             = cms.bool(False),
```

SiTrivialInduceChargeOnStrips

```
    CouplingConstant[Mode][subdet]   (so 2*14 coupling constants)
       Mode is Dec or Peak
       Subdet { "IB1","IB2","OB1","OB2","W1a","W2a","W3a","W1b","W2b",
                "W3b", "W4","W5","W6","W7"}
```

SiStripDigitizer

```
    DigiModeList = cms.PSet(SCDigi = cms.string('ScopeMode'),
                            ZSDigi = cms.string('ZeroSuppressed'),
                            PRDigi = cms.string('ProcessedRaw'),
                            VRDigi = cms.string('VirginRaw')),
    ROUList = cms.vstring("TrackerHitsTIBLowTof","TrackerHitsTIBHighTof",
                          "TrackerHitsTIDLowTof","TrackerHitsTIDHighTof",
                          "TrackerHitsTOBLowTof","TrackerHitsTOBHighTof",
                          "TrackerHitsTECLowTof","TrackerHitsTECHighTof"),
    GeometryType                = cms.string('idealForDigi'),
    TrackerConfigurationFromDB = cms.bool(False),
    ZeroSuppression             = cms.bool(True),
    LorentzAngle                = cms.string(''),
    Gain                        = cms.string(''),
```

SiStripDigitizerAlgorithm

```
    NoiseSigmaThreshold         = cms.double(2.0),
    electronPerAdcDec           = cms.double(247.0),
             #tuned on collisions at 7 TeV
    electronPerAdcPeak          = cms.double(262.0),
             #tuned on craft08
    FedAlgorithm                = cms.int32(4),
    Noise                       = cms.bool(True),
```

Parameters valid only if Noise = True and ZeroSuppression = False

```
    *RealPedestals               = cms.bool(True),
    *SingleStripNoise            = cms.bool(True),
    CommonModeNoise             = cms.bool(True),
    BaselineShift               = cms.bool(True),
    APVSaturationFromHIP        = cms.bool(True),
    APVSaturationProb           = cms.double(0.001),
    cmnRMStib                   = cms.double(5.92),
    cmnRMStob                   = cms.double(1.08),
    cmnRMStid                   = cms.double(3.08),
    cmnRMStec                   = cms.double(2.44),
    PedestalsOffset             = cms.double(128),
```

* The pedestal and noise RMS for each stip is read from the Db. if False it is added to all the strips the cnetral strip pedestal value, noise RMS

```
    TOFCutForDeconvolution      = cms.double(50.0),
    TOFCutForPeak               = cms.double(100.0),
    Inefficiency                = cms.double(0.0)
```

Parameters potentially changed by aging

The first project

Impact of Temp. param.

- Study the impact of the Temperature parameter on the simulation
- Seen in previous slide :

```
Temperature                = cms.double(273.0),
```

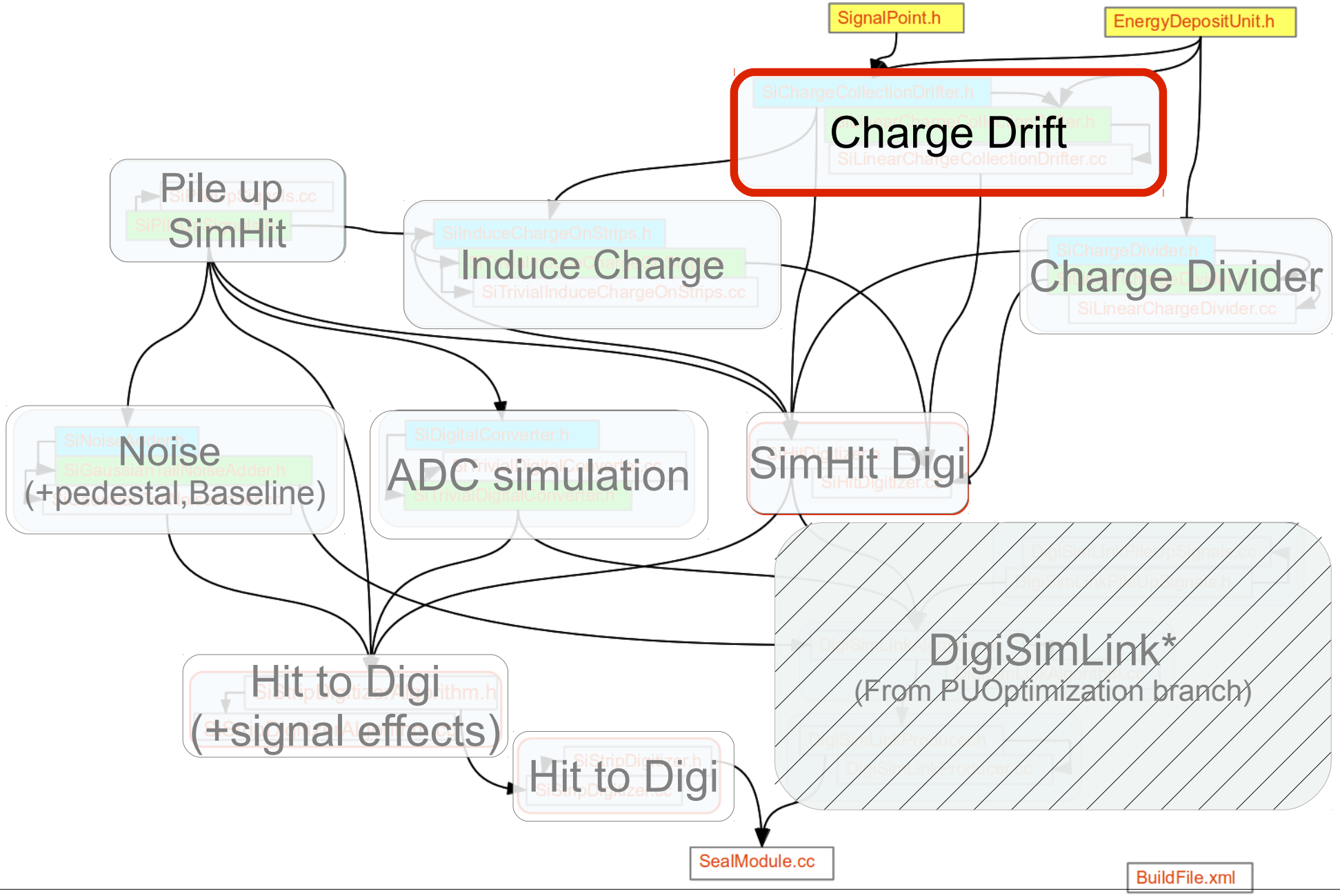
- This parameter is used in the diffusion constant computation required to compute the drift time of the charge in the sensor (we will see a bit further)
  - ➔ As seen before, the charge drift is one of the first step

- In 2015 runs, the temperature is lowered to ~-20°C,
  - ➔ How does distributions are impacted by this change?
  - ➔ If you see impact, how a few degrees variations around this value impact the distributions
    - ➔ to see if keeping a global temperature is a fair approximation

- Note that it may not be the only temperature effect in the simulation, as other parameters used/defined are temperature dependent but not defined from this configuration temperature value.
  You can try to list temperature dependence in the sim.
  (not mandatory, we can see together when you have the first results)

Some details on modules

Depletion and applied Voltage



Depletion and applied Voltage

- In **SiHitDigitizer.cc** :

```
double timeNormalisation = (moduleThickness*moduleThickness)/(2.*depletionVoltage*chargeMobility);
```

$$t_{Norm} = \frac{Thickness_{mod}^{2}}{2.V_{Depl}.chargeMobility}$$

cfi

$$chargeMobility \propto \left[\frac{cm^2}{V.s}\right]$$

- In **SiLinearChargeCollectionDrifter** (drift time in the sensor) :

```
double driftTime = -timeNormalisation
                   *log(1.-2*depletionVoltage*thicknessFraction/(depletionVoltage+appliedVoltage))
                   +chargeDistributionRMS;
```

```
SignalPoint drift (EnergyDepositUnit edu, Localvector drift, moduleThickness, timeNormalisation)
```

- computes the fraction of the module the charge has to drift through,

$$depth = \frac{moduleThickness}{2} - energyDeposit.z()$$

$$thicknesFraction = \frac{depth}{moduleThickness}$$

- ensure thicknessFraction Є [0,1]

cfi

- Compute the drift time in the sensor

$$driftTime = -t_{Norm} * \log\left(1 - \frac{2.V_{Depl}}{V_{Depl}+V_{Appl.}}.thicknessFraction\right) + chargeDistributionRMS$$

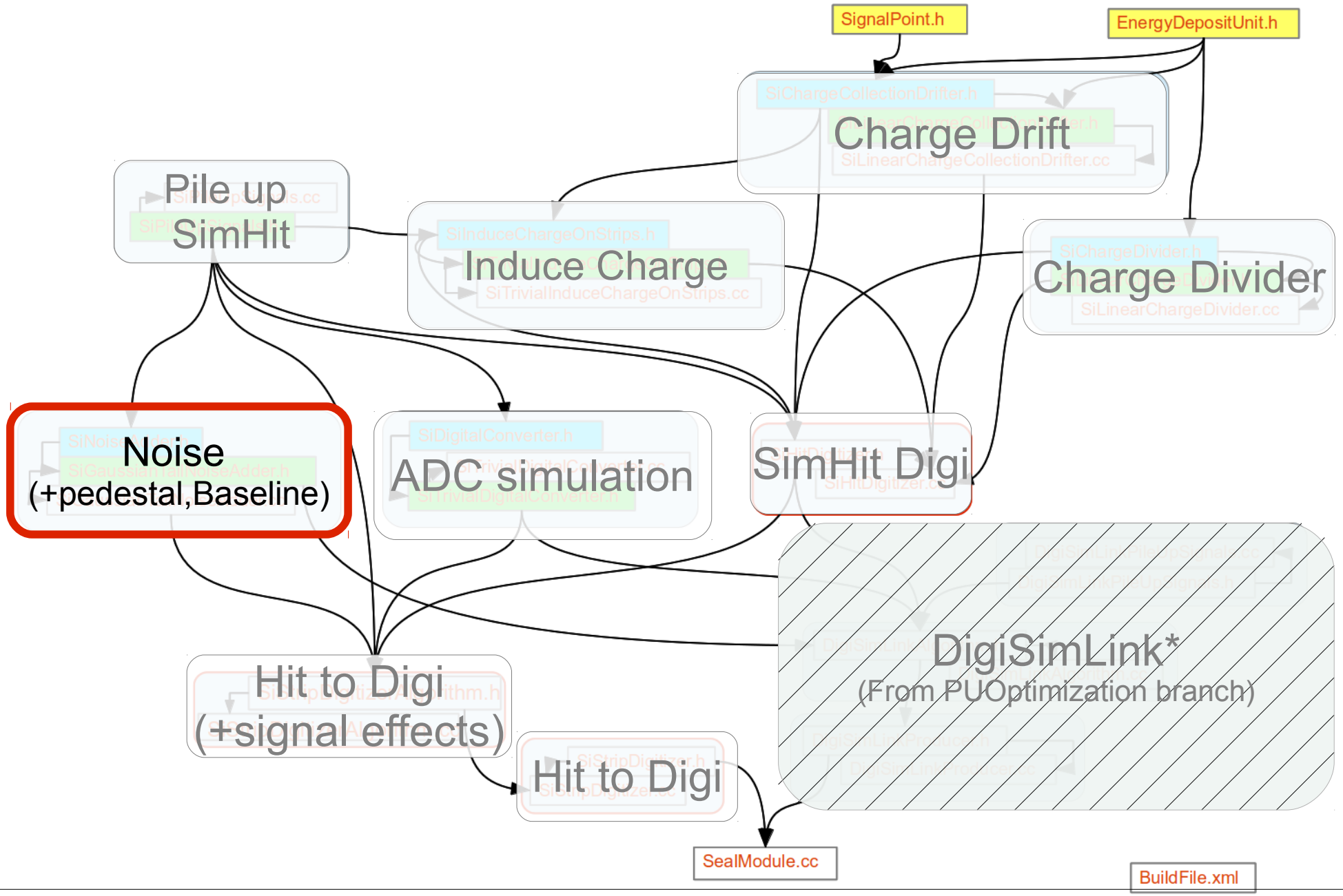
- Return the SignalPoint(x,y,sigma,amplitude): pos., an energy on the surface, with a size due to diffusion.

$$SignalPoint\left(edu.x() + \frac{depth * drift.x()}{drift.z()}, edu.y() + \frac{depth * drift.y()}{drift.z()}, \sqrt{2.*diffusionConstant*driftTime}, edu.energy()\right)$$

$$= \frac{C_{BOLTZ}}{e} * chargeMobility * Temp * \begin{bmatrix} 1.0\ if\ noDiffusion \\ 1.0e\text{-}3\ !\ noDiffusion \end{bmatrix}$$

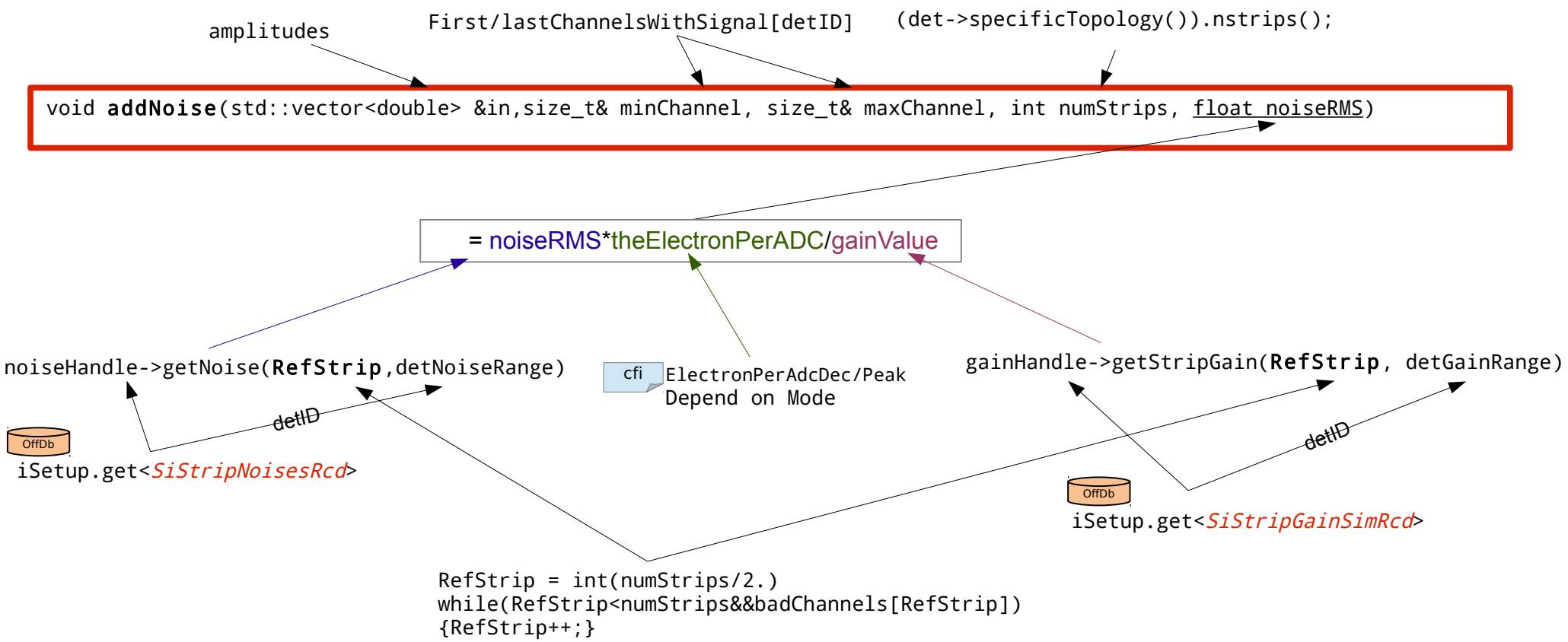
➔ **Effect on coupling wich use the chargeSpread**

Noises, pedestal and baseline



Noise in ZeroSuppression (1)

Noise in ZeroSuppression inputs:



Noise in ZeroSuppression (2)

Noise in ZeroSuppression addNoise implementation

- Produce noise using *GaussianTailNoiseGenerator*

```
std::vector<std::pair<int,float> > generatedNoise;
genNoise->generate(numStrips, threshold, noiseRMS, generatedNoise);
```

```
genNoise(new GaussianTailNoiseGenerator(rndEngine))
```

cfi `"NoiseSigmaThreshold"`

Generation of random noise on "numberOfChannels" channels with a given threshold.
The generated noise:
- corresponds to a Gaussian distribution of RMS = "noiseRMS"
- is larger than threshold*noiseRMS.

- Gaussian Noise **on strips with signal** with *CLHEP::RandGaussQ(rndEngine)*:
  - Loop on channels and add noise if not null

```
in[iChannel] += gaussDistribution->fire(0.,noiseRMS);
```

```
gaussDistribution(new CLHEP::RandGaussQ(rndEngine))
```

- Noise **on the other strips**
  - Loop on generatedNoise vector pair generated at top
    - Add to channel the corresponding noise (and check that value ==0)

```
in[(*p).first] += (*p).second;
```

VirginRaw Noises, Pedestal, baseline

(in Si*NoiseAdder & SiDigiAlgo)

```
void addBaselineShift(std::vector<double> &in, std::vector<bool> &badChannels)
```

```
    size_t nAPVs = in.size()/128;
      std::vector<float> vShift;
      double apvCharge, apvMult;

      size_t iChannel;
      for(size_t APVn =0; APVn < nAPVs; ++APVn){
            apvMult=0; apvCharge=0;
            for(iChannel=APVn*128; iChannel!=APVn*128+128; ++iChannel) {
               if(in[iChannel]>0){
                     ++apvMult;
                      apvCharge+= in[iChannel];
               }
               if(apvMult==0) vShift.push_back(0);
               else vShift.push_back(apvCharge/apvMult);
            }
      }
      for (iChannel=0; iChannel!=in.size(); ++iChannel) {
         if(!badChannels[iChannel]) in[iChannel] -= vShift[(int)(iChannel/128)];
      }
```

Are aging studies needed in VirginRaw mode?

```
void addNoiseVR(std::vector<double> &in, std::vector<float> &noiseRMS)
```

```
    Add noise
      // Full Gaussian noise is added everywhere
         if(noiseRMS[iChannel] > 0.) in[iChannel] += gaussDistribution->fire(0.,noiseRMS[iChannel]);
```

```
void addCMNoise(std::vector<double> &in, float cmnRMS, std::vector<bool> &badChannels)
```

```
  int nAPVs = in.size()/128;
  std::vector<float> CMNv;
  for(int APVn =0; APVn < nAPVs; ++APVn) CMNv.push_back(gaussDistribution->fire(0.,cmnRMS));
  for (size_t iChannel=0; iChannel!=in.size(); iChannel++) {
     if(!badChannels[iChannel]) in[iChannel] += CMNv[(int)(iChannel/128)];
  }
```

```
void addPedestals(std::vector<double> &in,std::vector<float> & ped)
```

```
        for (size_t iChannel=0; iChannel!=in.size(); iChannel++) {
          if(ped[iChannel]>0.) in[iChannel] += ped[iChannel];   } }
```

VirginRaw : APV killer

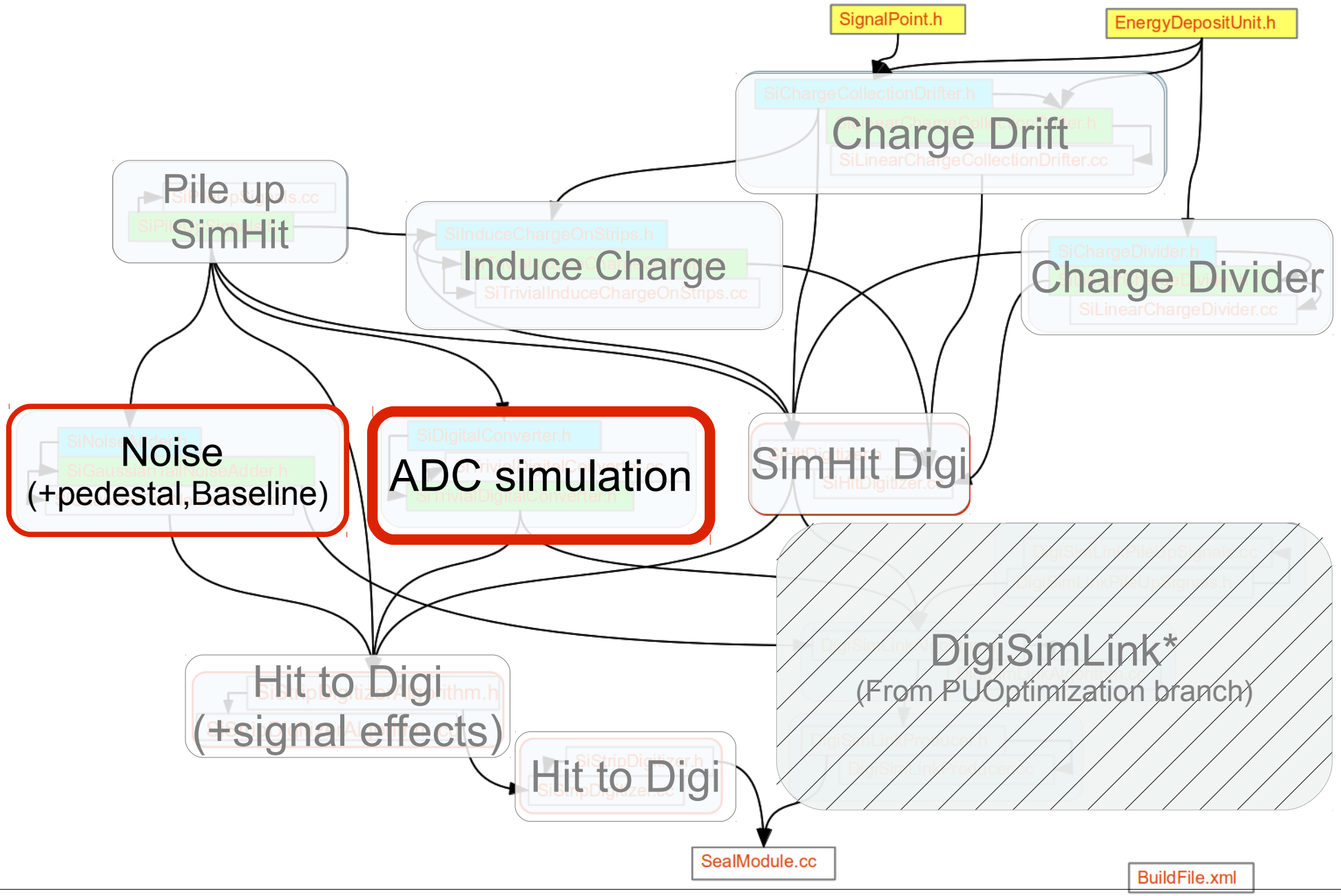
(in Si*NoiseAdder & SiDigiAlgo)

- In SiStripDigitizerAlgorithm
- Aim is to simulate APV « killed » by high « signal » on many strips so no clear signal can be extracted

Are aging studies needed in VirginRaw mode?

```
if(APVSaturationFromHIP&&!zeroSuppression){
                    int pdg_id = simHitIter->particleType();
                        particle = pdt->particle(pdg_id);
                        if(particle != NULL){
                                float charge = particle->charge();
                                bool isHadron = particle->isHadron();
                            if(charge!=0 && isHadron){
                                        if(theFlatDistribution->fire()<APVSaturationProb){
                                                int FirstAPV = localFirstChannel/128;
                                                int LastAPV = localLastChannel/128;
                                                std::cout << "-------------------HIP--------------" << std::endl;
                                                std::cout<<"Killing APVs "<<FirstAPV<< " - " <<LastAPV<< " " <<detID <<std::end
                                                for(int strip = FirstAPV*128; strip < LastAPV*128 +128; ++strip) {
                                                        badChannels[strip] = true;
                                                }
                                                //doing like that I remove the signal information only after the
                                                //stip that got the HIP but it remains the signal of the previous
                                                //one. I'll make a further loop to remove all signal
                                        }
                                }
                        }
```

Gains


SignalPoint.h
EnergyDepositUnit.h
Charge Drift
Pile up SimHit
Induce Charge
Charge Divider
Noise (+pedestal,Baseline)
ADC simulation
SimHit Digi
DigiSimLink*
(From PUOptimization branch)
Hit to Digi (+signal effects)
Hit to Digi
SealModule.cc
BuildFile.xml

Gains

- When adding noise (see before)
- When converting to ADC

```
DigitalVecType convert(const std::vector<double>& analogSignal, edm::ESHandle<SiStripGain> & gainHandle, unsigned int detid)
```

```
gainHandle->getStripGain(RefStrip, detGainRange)
```

detID

OffDb

```
iSetup.get<SiStripGainSimRcd>
```

```
RefStrip = int(numStrips/2.)
while(RefStrip<numStrips&&badChannels[RefStrip])
{RefStrip++;}
```

- Check if gainHandle is valid
  - Loop on analogsignal vector and if >0 (else set 0)
    - Convert analog to digital

```
int adc = convert( (gainHandle->getStripGain(i, detGainRange))*(analogSignal[i]) );
```

    - Add to to digitalrectype if >0

- If gainHandle not valid :
  - Same but don't multiply by a gain

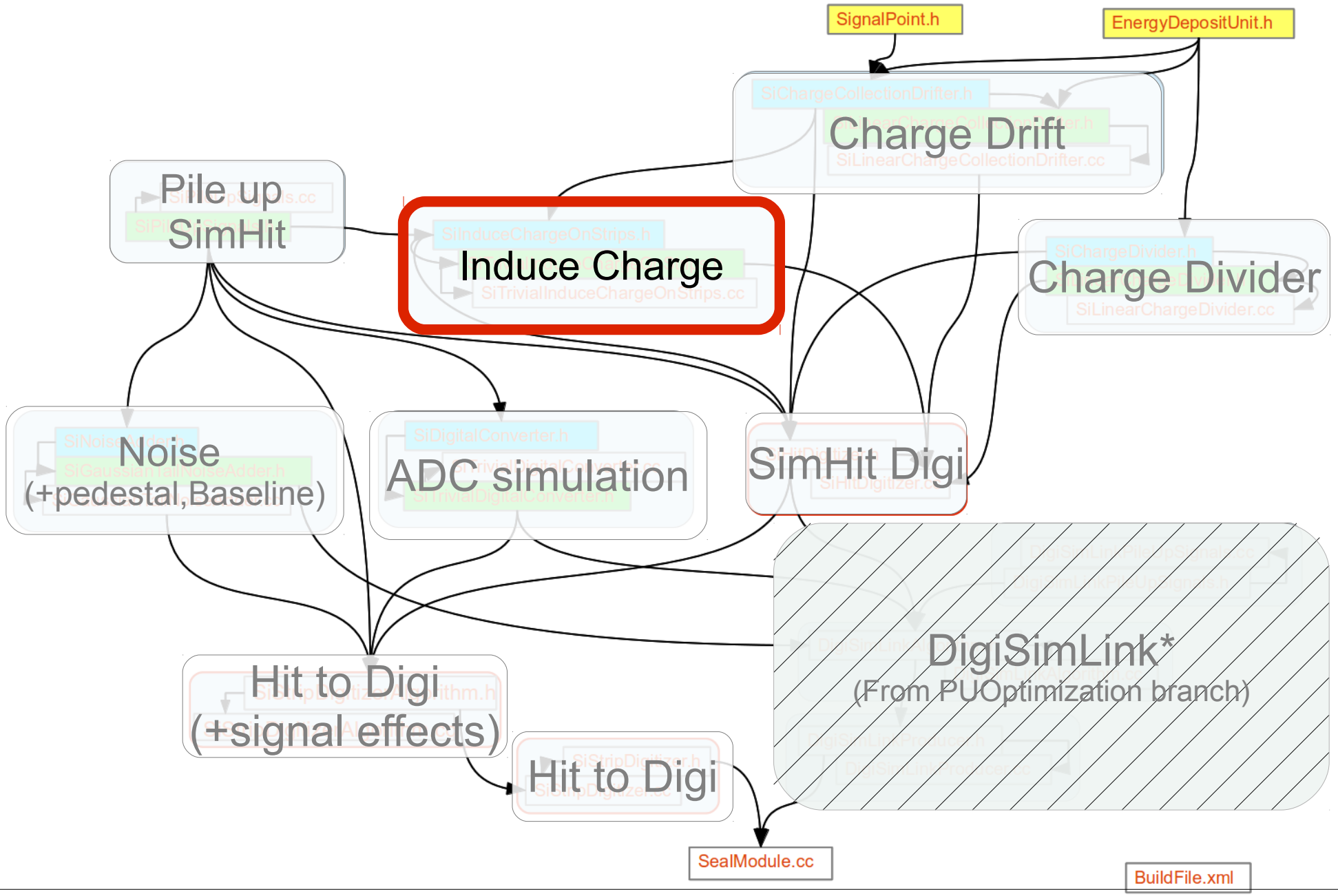
```
adc = convert( analogSignal[i] );
```

int convert(float in) {return truncate(in/electronperADC);}

Truncate :

```
If (raw charge < 254)          -> adc unchanged
If (254  <= raw charge < 1023) -> adc=254
If (raw charge >= 1023)        -> adc=255
```

Couplings


SignalPoint.h
EnergyDepositUnit.h
Charge Drift
Pile up SimHit
Induce Charge
Charge Divider
Noise (+pedestal,Baseline)
ADC simulation
SimHit Digi
DigiSimLink*
(From PUOptimization branch)
Hit to Digi (+signal effects)
Hit to Digi
SealModule.cc
BuildFile.xml

Couplings (1)

```
type[Ntypes] = { "IB1", "IB2","OB1","OB2","W1a","W2a","W3a","W1b","W2b","W3b","W4","W5","W6","W7"};
```

```
double chargeDeposited(size_t strip,size_t Nstrips,double amplitude,double chargeSpread,double chargePosition)
```

- Determine integral and fraction of signal
  - integralUpToStrip = (strip == 0) ? 0. : (normal_cdf(strip, **chargeSpread**, chargePosition));
  - integralUpToNext = (strip+1 == Nstrips) ? 1. : (normal_cdf(strip+1, **chargeSpread**, chargePosition));
  - percentOfSignal = integralUpToNext - integralUpToStrip;
- return percentOfSignal * amplitude / geVperElectron;

cfi

```
void induce(collection_type collection_points,StripGeomDetUnit det,localAmplitudes,
            size_t recordMinAffectedStrip, size_t recordMaxAffectedStrip,TrackerTopology *tTopo)
```

- Get the coupling ; Get the topology ; The number of strip in this topology
- Loop on signal points in collection :
  - Define :
    - chargePosition=from signal point ; chargeSpread : $chargeSpread = \frac{signalPoint \rightarrow sigma()}{localPitch}$
    - firtStrip and untilStrip from chargePosition±Nsigma*chargeSpread
  - Loop on strips :
    - Compute charge deposited on the Strip =chargeDeposited(strip, Nstrips, signalpoint->amplitude(), chargeSpread, chargePosition)
    - Strip range affected by deposit (affectedFromStrip, affectedUntilStrip)

      using Nsigma, coupling, position
    - Loop on affectedStrip to apply induce charge :
      - $affectedStrip_{localAmplitude}$ += chargeDepositedOnStrip * coupling.at(abs(affectedStrip - strip))

Depend on V_{depl} V_{Appl}

cfi `CouplingConstant[Mode][subdet]`

Back up slides

SiChargeDivider : why ?

- In Geant4 :
  - the energy deposited on subsections of silicon sensitive section are simulated but not recorded

    ⇒ Only the total charge is recorded (E_{tot})
- In the SimTracker/SiStripDigitizer :
  - The pathlength is divided in n subsections
  - $E_{subsection} = \frac{E_{tot}}{n}$
  - Fluctuation δE_i on $E_{subsection,i}$ using « `SampleFluctuations` » routine from Geant4
  - Rescale factor ε in order to have

$$\varepsilon . \sum_i \left(E_{subsection,i} + \delta E_i \right) = E_{tot}$$

SiChargeDivider : why ?

- Can pulse and decoshape be modified by aging ?
  - Becauses hardcoded in the DividerPart

```
inline float TimeResponse( const PSimHit* hit, const StripGeomDetUnit& det) {
  return (peakMode ? PeakShape(hit,det) : DeconvolutionShape(hit,det));
}
```

```
// pulse shape in peak mode
float PeakShape(const PSimHit*, const StripGeomDetUnit& det);
// pulse shape in deconvolution mode
float DeconvolutionShape( const PSimHit*, const StripGeomDetUnit& det);
```

```
// data table for pulse shape in peak mode
static float peakValues[921];
// data table for pulse shape in deconvolution mode
static float decoValues[651];
```

SiHitDigitizer

```
SiHitDigitizer::SiHitDigitizer(const edm::ParameterSet& conf, CLHEP::HepRandomEngine& eng) :
  depletionVoltage(conf.getParameter<double>("DepletionVoltage")),
  chargeMobility(conf.getParameter<double>("ChargeMobility")),
  theSiChargeDivider(new SiLinearChargeDivider(conf, eng)),
  theSiChargeCollectionDrifter(new SiLinearChargeCollectionDrifter(
         CBOLTZ_over_e_SI * chargeMobility * conf.getParameter<double>("Temperature") *
(conf.getParameter<bool>("noDiffusion") ? noDiffusionMultiplier : 1.0),
         conf.getParameter<double>("ChargeDistributionRMS"),
         depletionVoltage,
         conf.getParameter<double>("AppliedVoltage"))),
  theSiInduceChargeOnStrips(new SiTrivialInduceChargeOnStrips(conf, conf.getParameter<double>("GevPerElectron"))) {
}


processHit(const PSimHit* hit, const StripGeomDetUnit& det, GlobalVector bfield,float langle,
                           std::vector<double>& locAmpl, size_t& firstChannelWithSignal, size_t& lastChannelWithSignal,
                           const TrackerTopology *tTopo){

  // Compute the drift direction for this det
  double moduleThickness = det.specificSurface().bounds().thickness(); // active detector thicness
  double timeNormalisation = (moduleThickness*moduleThickness)/(2.*depletionVoltage*chargeMobility);
  LocalVector driftDir = DriftDirection(&det,bfield,langle);

  // Fully process one SimHit
  theSiInduceChargeOnStrips->induce(
      theSiChargeCollectionDrifter->drift(
          theSiChargeDivider->divide(hit, driftDir, moduleThickness, det),
          driftDir,moduleThickness,timeNormalisation),
      det,locAmpl,firstChannelWithSignal,lastChannelWithSignal,tTopo);
}
```

SiLinearChargeCollectionDrifter

```
Drifts the charges linearly.
 * Drift each energy deposits in the bulk to the surface.
 * The resulting position depends on the Lorentz angle, and a sigma is computed that describes the diffusion.
```

```
SignalPoint drift (EnergyDepositUnit edu, Localvector drift, moduleThickness, timeNormalisation)
```

- computes the fraction of the module the charge has to drift through,

$$depth = \frac{moduleThickness}{2} - energyDeposit.z()$$

$$thicknesFraction = \frac{depth}{moduleThickness}$$

- ensure thicknessFraction Є [0,1]

- Compute the drift time in the sensor

$$driftTime = -timeNormalisation * \log\left(1. - \frac{2 * depletionVoltage * thicknessFraction}{depletionVoltage + appliedVoltage}\right) + chargeDistributionRMS$$

- Return the signal: an energy on the surface, with a size due to diffusion.

$$SignalPoint\left(edu.x() + \frac{depth * drift.x()}{drift.z()}, edu.y() + \frac{depth * drift.y()}{drift.z()}, \sqrt{2. * diffusionConstant * driftTime}, edu.energy()\right)$$

```
SiChargeCollectionDrifter::collection_type drift(ionization_type ion,LocalVector driftDir,double mt, double tn)
```

- Call the previous drift method for each deposit
  _temp[i] = drift(ion[i], driftDir, mt, tn);

- Return a collection_type

SiTrivialInduceChargeOnStrip (1)

```
* Given a SignalPoint, computes the charge on each strip, using also the coupling between strips
```

```
type[Ntypes] = { "IB1", "IB2","OB1","OB2","W1a","W2a","W3a","W1b","W2b","W3b","W4","W5","W6","W7"};
```

```
std::vector<std::vector<double>> fillSignalCoupling(edm::ParameterSet& conf, nTypes, string typeArray)
```

- Define mode from *APVPeakmode*
- For each type, fill vector with corresponding coupling constant
- Return vector of coupling constant

```
int indexOf(const std::string& t)
```

- return index of subdetector type in type vector

```
int typeOf(StripGeomDetUnit& det, TrackerTopology *tTopo)
```

- return the corresponding index of subdetector type from the detunit and Topo

```
SiTrivialInduceChargeOnStrips(const edm::ParameterSet& conf,double g)
  : signalCoupling(fillSignalCoupling(conf, Ntypes, type)), Nsigma(3.), geVperElectron(g)  {}
```

- Constructor

```
double chargeDeposited(size_t strip,size_t Nstrips,double amplitude,double chargeSpread,double chargePosition)
```

- Determine integral and fraction of signal
  - integralUpToStrip = (strip == 0) ? 0. : (normal_cdf(strip, chargeSpread, chargePosition));
  - integralUpToNext = (strip+1 == Nstrips) ? 1. : (normal_cdf(strip+1, chargeSpread, chargePosition));
  - percentOfSignal = integralUpToNext - integralUpToStrip;
- return percentOfSignal * amplitude / geVperElectron;

SiTrivialInduceChargeOnStrip (2)

```
void induce(collection_type collection_points,StripGeomDetUnit det,localAmplitudes,
            size_t recordMinAffectedStrip, size_t recordMaxAffectedStrip,TrackerTopology *tTopo)
```

- Get the coupling
- Get the topology
- The number of strip in this topology
- Loop on signal points in collection :
  - Define :
    - chargePosition=from signal point
    - chargeSpread : $chargeSpread = \frac{signalPoint \rightarrow sigma()}{localPitch}$
    - firtStrip and untilStrip from chargePosition±Nsigma*chargeSpread
  - Loop on strips :
    - Compute charge deposited on the Strip

      =chargeDeposited(strip, Nstrips, signalpoint->amplitude(), chargeSpread, chargePosition)
    - Strip range affected by deposit (affectedFromStrip, affectedUntilStrip)

      using Nsigma, coupling, position
    - Loop on affectedStrip to apply induce charge :
      - $affectedStrip_{localAmplitude}$ += chargeDepositedOnStrip * coupling.at(abs(affectedStrip - strip))

SiStripDigitizer (1)

```
* SiStripDigitizer to convert hits to digis
```

```
void finalizeEvent (edm::Event &e, edm::EventSetup const &c)
```

- LOOP on StripGeomDetUnit
  - apply the cable map _before_ digitization: consider only the detis that are connected
  - Apply digitization (noise, etc) (using theDigiAlgo->digitize();)
  - Fill digivector (zerosuppression or not)
  - Case zerosuppression or VR
    - create output collection with digivector
    - write output to file

```
void initializeEvent (edm::Event const &e, edm::EventSetup const &c)
```

- Get Inputs
- Initialize noise, etc (using theDigiAlgo->initializeEvent(iSetup))
- clear detectorUnits
- Loop on detector units
  - (re)fill
  - Init them (using theDigiAlgo->initializeDetUnit())

SiStripDigitizer (2)

```
* SiStripDigitizer to convert hits to digis
```

```
void accumulateStripHits (edm::Handle< std::vector< PSimHit > >, const TrackerTopology *tTopo)
```

- Check if hit vector is valid
- Loop on hits
  - Check if the insert succeeded, so this detector element has not yet been processed
  - access to magnetic field in global coordinates
  - Accumulate simhits (using theDigiAlgo->accumulateSimHits(it, itEnd, stripdet, bfield, tTopo))

```
void accumulate (edm::Event const &e, edm::EventSetup const &c)
```

- Retrieve tracker topology from geometry
- Loop on trackerContainers
  - Get inputs
  - accumulateStripHits(simHits,tTopo);

```
void accumulate (PileUpEventPrincipal const &e, edm::EventSetup const &c)
```

- Same as before

SiStripDigitizerAlgorithm (1)

```
* SiStripDigitizerAlgorithm to convert hits to digis
```

```
void initializeDetUnit (StripGeomDetUnit *det, const edm::EventSetup &iSetup)
```

- storing the bad strip of the the module. the module is not removed but just signal put to 0

```
void initializeEvent (const edm::EventSetup &iSetup)
```

- get gain noise pedestal lorentzAngle from ES handle

```
void setParticleDataTable (const ParticleDataTable *pardt)
```

- Set particle datatable (using theSiHitDigitizer->setParticleDataTable(pardt))

```
SiStripDigitizerAlgorithm (const edm::ParameterSet &conf, CLHEP::HepRandomEngine &)
```

- constructor

```
void accumulateSimHits (const std::vector< PSimHit >::const_iterator inputBegin, const std::vector< PSimHit
>::const_iterator inputEnd, const StripGeomDetUnit *stripdet, const GlobalVector &bfield)
```

- loop on the SimHits
  - skip hits not in this detector.
  - check TOF and energydeposit
  - process the hit (from sihitdigitizer)
  - APV Killer to simulate HIP effect
- Pileup the hits (using theSiPileUpSignals->add())

SiStripDigitizerAlgorithm (2)

```
* SiStripDigitizerAlgorithm to convert hits to digis
```

```
void digitize (edm::DetSet< SiStripDigi > &outDigis, edm::DetSet< SiStripRawDigi > &outRawDigis, const
StripGeomDetUnit *stripdet, edm::ESHandle< SiStripGain > &, edm::ESHandle< SiStripThreshold > &, edm::ESHandle<
SiStripNoises > &, edm::ESHandle< SiStripPedestals > &)
```

- removing signal from the dead (and HIP effected) strips
- Zerosuppression :
  - Noise by refstrip
- !zerosuppression
  - calculating the charge deposited on each APV and subtracting the shift
    (BaseLineShift using theSiNoiseAdder->addBaselineShift())
  - Adding the strip noise
    - Either SingleStripNoise
    - Or by refstrip
    - Add noise to signal (using theSiNoiseAdder->addNoiseVR())
  - Adding the Common Mode Noise
    - Get cmnRMS value for detector and multiply by electronPerADC
    - Add noise to signal (using theSiNoiseAdder->addCMNoise)
  - Adding the pedestals
    - Real pedestal by strip
    - Using pedoffset
    - Add to signal (using theSiNoiseAdder->addPedestals())
  - Produce rawdigis (using theSiDigitalConverter->convertRaw(detAmpl, gainHandle, detID))

SiPileUpSignals

```
* Class which takes the responses from each SimHit and piles-up them, within a given module.
* More precisely, it keeps for each strip the link to each individual measurement.
```

```
const SignalMapType* getSignal(uint32_t detID)
```

```
auto where = signal_.find(detID);
    if(where == signal_.end()) {
      return 0;
    }
    return &where->second;
  }
```

```
void resetSignals()
```

- Clear the signal

```
void add(uint32_t detID, vector<double>& locAmpl, size_t firstChannelWithSignal, size_t lastChannelWithSignal)
```

- Loop on channels within signals bounds
  - If signal!=0
    - If channel present :
      - Create pair channe- signal amplitude
    - Else
      - Add signal value to previous piled up values for this channel

SiHiDigitizer

```
* Digitizes the response for a single SimHit.
```

```
void processHit (const PSimHit *, const StripGeomDetUnit &, GlobalVector, float, std::vector< double > &,
size_t &, size_t &)
```

- LOOP on StripGeomDetUnit
  - apply the cable map _before_ digitization: consider only the detis that are connected
  - Apply digitization (noise, etc) (using theDigiAlgo->digitize();)
  - Fill digivector (zerosuppression or not)
  - Case zerosuppression or VR
    - create output collection with digivector
    - write output to file

```
void setChargeCollectionDrifter (SiChargeCollectionDrifter *cd)
```

- Get Inputs

```
void setChargeCollectionDrifter (SiChargeCollectionDrifter *cd)
```

- Get Inputs

```
void setChargeCollectionDrifter (SiChargeCollectionDrifter *cd)
```

- Get Inputs

Noise and Gain

```
SiStripDigitizerAlgorithm.cc:91:  //get gain noise pedestal lorentzAngle from ES handle
SiStripDigitizerAlgorithm.cc:179:                                edm::ESHandle<SiStripGain> & gainHandle,
SiStripDigitizerAlgorithm.cc:200:  SiStripApvGain::Range detGainRange = gainHandle->getRange(detID);
SiStripDigitizerAlgorithm.cc:216:                    float gainValue = gainHandle->getStripGain(RefStrip, detGainRange);
SiStripDigitizerAlgorithm.cc:217:                    theSiNoiseAdder-
>addNoise(detAmpl,firstChannelWithSignal,lastChannelWithSignal,numStrips,noiseRMS*theElectronPerADC/gainValue);
SiStripDigitizerAlgorithm.cc:221:    theSiZeroSuppress->suppress(theSiDigitalConverter->convert(detAmpl, gainHandle, detID), digis,
detID,noiseHandle,thresholdHandle);
SiStripDigitizerAlgorithm.cc:321:    DigitalRawVecType rawdigis = theSiDigitalConverter->convertRaw(detAmpl, gainHandle, detID);
SiStripDigitizer.cc:56:  gainLabel(conf.getParameter<std::string>("Gain")),
SiStripDigitizer.cc:184:  edm::ESHandle<SiStripGain> gainHandle;
SiStripDigitizer.cc:188:  iSetup.get<SiStripGainSimRcd>().get(gainLabel,gainHandle);
SiStripDigitizer.cc:212:                             gainHandle,thresholdHandle,noiseHandle,pedestalHandle);
SiTrivialDigitalConverter.cc:12:SiTrivialDigitalConverter::convert(const std::vector<double>& analogSignal, edm::ESHandle<SiStripGain> & gainHandle,
unsigned int detid){
SiTrivialDigitalConverter.cc:16:  if(gainHandle.isValid()) {
SiTrivialDigitalConverter.cc:17:    SiStripApvGain::Range detGainRange = gainHandle->getRange(detid);
SiTrivialDigitalConverter.cc:21:      int adc = convert( (gainHandle->getStripGain(i, detGainRange))*(analogSignal[i]) );
SiTrivialDigitalConverter.cc:36:SiTrivialDigitalConverter::convertRaw(const std::vector<double>& analogSignal, edm::ESHandle<SiStripGain> &
gainHandle, unsigned int detid){
SiTrivialDigitalConverter.cc:40:  if(gainHandle.isValid()) {
SiTrivialDigitalConverter.cc:41:    SiStripApvGain::Range detGainRange = gainHandle->getRange(detid);
SiTrivialDigitalConverter.cc:45:      int adc = convertRaw( (gainHandle->getStripGain(i, detGainRange))*(analogSignal[i]));
SiDigitalConverter.h:6:#include "CalibFormats/SiStripObjects/interface/SiStripGain.h"
SiDigitalConverter.h:18:  virtual DigitalVecType    convert(const std::vector<double> &,  edm::ESHandle<SiStripGain>& ,unsigned int detid) = 0;
SiDigitalConverter.h:19:  virtual DigitalRawVecType convertRaw(const std::vector<double> &,  edm::ESHandle<SiStripGain>& ,unsigned int detid) = 0;
SiStripDigitizerAlgorithm.h:24:#include "CalibFormats/SiStripObjects/interface/SiStripGain.h"
SiStripDigitizerAlgorithm.h:77:                edm::ESHandle<SiStripGain>&,
SiStripDigitizer.h:57:  const std::string gainLabel;
SiTrivialDigitalConverter.h:13:  DigitalVecType    convert(const std::vector<double>&,  edm::ESHandle<SiStripGain>& ,unsigned int detid);
SiTrivialDigitalConverter.h:14:  DigitalRawVecType convertRaw(const std::vector<double>&,  edm::ESHandle<SiStripGain>& ,unsigned int detid);
```