Silicon Simulation and Digitization in the New Geometry Framework

Tim Nelson

Jeremy McCormick

SLAC

New Geometry Framework

- Based on concepts from GAUDI/LHCb and ATLAS
- Two hierarchies:
  - Geometry (LogicalVolume/PhysicalVolume) - all material in the detector
  - Functional (DetectorElements) - all functional elements of the detector

The functional hierarchy ties into the geometry hierarchy to for geometry information. Otherwise the two are distinct.

SiTrackerBarrel Description

- LogicalVolumes
    - Barrel
        - Layers
            - Modules
                - Carbon fiber
                - Rohacell
                - PEEK
                - KPiX
                - Cable
                - Sensor

- DetectorElements
    - Barrel
        - Layers
            - Modules
                - Sensors

Barrels, layers and modules are merely containers for sensors. All the action takes place in the sensor.

SiSim

- Charge deposition: from SimTrackerHits to electrons collected on electrodes
    - class SiSensor extends DetectorElement
        - class DopedSilicon (bulk)
        - class SiStrips implements SiSensorElectrodes (plan also SiPixels)
- Readout: addition of noise and digitization of charge
    - class Kpix implements interface ReadoutChip

Output of the readout chip and a fully qualified CellID are what we need to construct RawTrackerHits.

Creating SiSensors

From SiTrackerBarrelConverter:

```
String sensorPath = modulePath.toString() + "/" + pv.getName();
String sensorName = subdet.getName() + "_module" + moduleId + "_sensor" + sensorId;
SiSensor sensor = new SiSensor(sensorId, sensorName, module, sensorPath);
sensor.setIdentifier( id );

sensor.setElectrodes(ChargeCarrier.HOLE,new SiStrips(3679,0.025,1));
sensor.setElectrodeAngle(ChargeCarrier.HOLE,0.0);
sensor.setOrientation(Orientation.PSIDE_POSITIVE_Z);
sensor.initialize();
```

Driving Charge Deposition

From SimTrackerDigitizationTest

```
// set of sensors with hits
Set<SiSensor> hit_sensors = new HashSet<SiSensor>();

for (SimTrackerHit hit : eventHits)
{
    // Create track segment in global coordinates
    Hep3Vector midpoint = new BasicHep3Vector(hit.getPoint());
    Hep3Vector direction = VecOp.unit(new BasicHep3Vector(hit.getMomentum()));
    Hep3Vector half_length = VecOp.mult(hit.getPathLength()/2.0,direction);

    Hep3Vector p1 = VecOp.add(midpoint,VecOp.mult(-1.0,half_length));
    Hep3Vector p2 = VecOp.add(midpoint,half_length);
    double energy = hit.getdEdx();
    TrackSegment track_segment = new TrackSegment(p1,p2,energy);

    // Find the sensor for this hit, add to list of hit sensors
    SiSensor sensor = (SiSensor)tkr.findDetectorElement(midpoint);
    hit_sensors.add(sensor);

    // Transform track segment to sensor coordinates and assign to sensor
    ITransform3D global_to_sensor = sensor.getGeometry().getGlobalToLocal();
    track_segment.transform(global_to_sensor);
    sensor.addTrackSegment(track_segment);
}

// Loop over sensors and deposit charge
for (SiSensor sensor : hit_sensors)
{
    sensor.depositCharge();
}
```

Inside Charge Deposition

CDF/Padova model Includes many drift corrections

- dependence of electric field on position in depleted region (assumes constant space charge density)
  - arbitrary configurations could be added via ElectricFieldModel and might be needed in some special cases
- depth-dependent Lorentz angle (important only for electrons)
- lorentz corrections to diffusion (width of diffusion at surface scales roughly with $\tan^2\theta_L$
- dependence of mobility on temperature and carrier concentration
- ~~low-energy delta-rays~~ with small steps/range cuts in sensors GEANT4 does fine

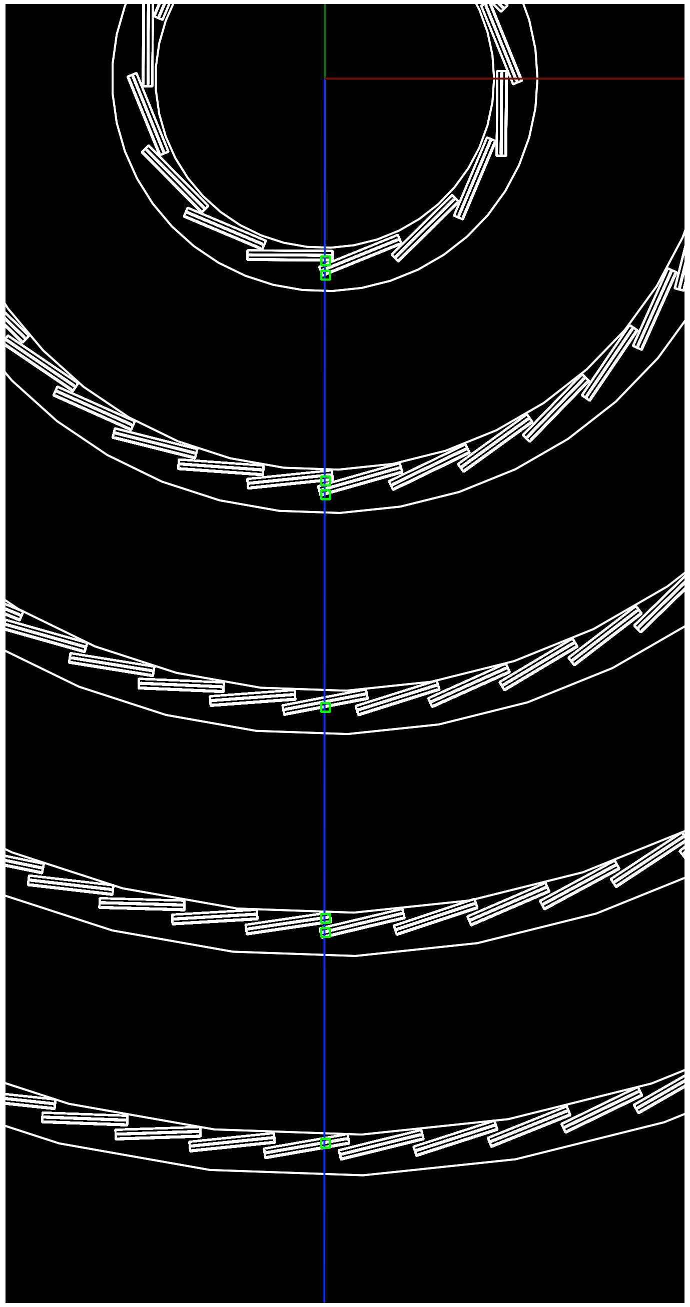
CMS model offers some improvements

- studied up to field of 9T!
- incorporating these ideas into an alternate model of DopedSilicon

Readout

```
// readout chip
ReadoutChip readout_chip = new Kpix();

// readout
for (ChargeCarrier carrier : ChargeCarrier.values())
{
    if (sensor.hasElectrodesOnSide(carrier))
    {
        Map<Integer,Integer> raw_hits = readout_chip.readout
        (sensor.getElectrodes(carrier));
    }
}
```

```
Sensor: SiTrackerBarrel_module80_sensor0
Charge Map for HOLE: {116=1, 117=18326, 118=1740}
Readout Map for HOLE: {117=58, 118=5}


Sensor: SiTrackerBarrel_module3474_sensor0
Charge Map for HOLE: {1801=2440, 1802=17285, 1803=7194}
Readout Map for HOLE: {1801=7, 1802=55, 1803=23}


Sensor: SiTrackerBarrel_module3445_sensor0
Charge Map for HOLE: {69=9401, 70=17639}
Readout Map for HOLE: {69=30, 70=56}


Sensor: SiTrackerBarrel_module524_sensor0
```



Creating RawTrackerHits

- Need ID: this is currently in progress (a first implementation was checked in late yesterday by Jeremy)
- Need to adapt RawTrackerHit to new geometry framework?

Interface RawTrackerHit

```
public interface RawTrackerHit
{
   /**
    *  Returns a time measurement associated with the adc values.
    *  E.g. the t0 of the spectrum for the TPC. Subdetector dependent.
    */
    int getTime();
    /**
     * Returns the detector specific cell id.
     */
    long getCellID();
    /**
     * Returns the array of ADCValues.
     * The array may be of length 1 if this detector only reads out a single value per cell.
     * The value may also need decoding (for example the KPiX chip uses one bit as a
     * range indicator).
     */
    short[] getADCValues();
    /** Returns the IDDecoder associated with this hit */
    IDDecoder getIDDecoder();
    /** Using the IDDecoder, returns the Subdetector associated with this hit */
    Subdetector getSubdetector();
    /** Returns the associated SimTrackerHit. Note this may be <code>null</code>
     * if there is no associated SimTrackerHit (for example because this is a noise
     * hit, or because there is no MC information.
     */
    // FixMe: This should return an array of tracker hits
    SimTrackerHit getSimTrackerHit();
}
```

Summary

- After sitting unused in cvs for many months, SiSim is being adapted into a framework for simulation and reconstruction of silicon hits for org.lcsim using the new geometry framework. Some pieces still need work:
  - noise modeling (looking at digisim for components to re-use)
  - ID infrastructure, Hit classes and retracing of MC truth information
- As much or more work has gone into making the framework modular and extensible as has gone into the charge deposition and digitization.

The intent is to use this framework as a place to hang a lot of very good work done on silicon simulation by many people.