Noise Hit Generation for ElCrecon

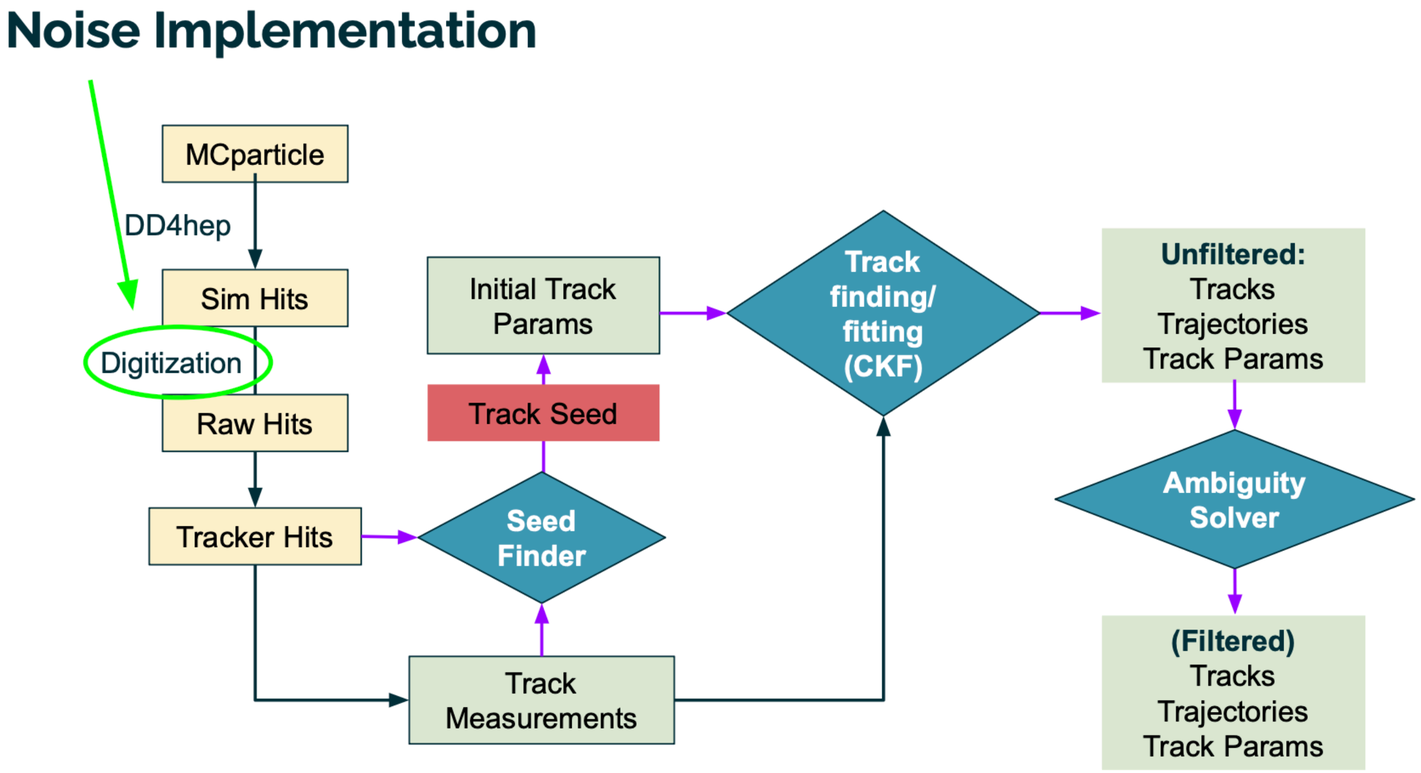
Minjung Kim (UC Berkeley)

21 July 2025 (Tue.)

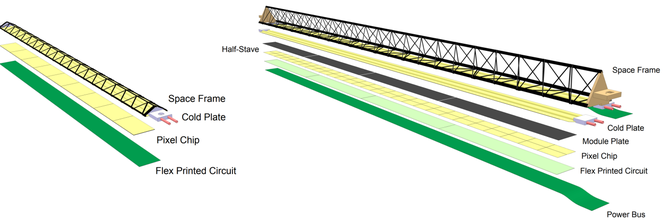
Noise Hit Generation for EICrecon

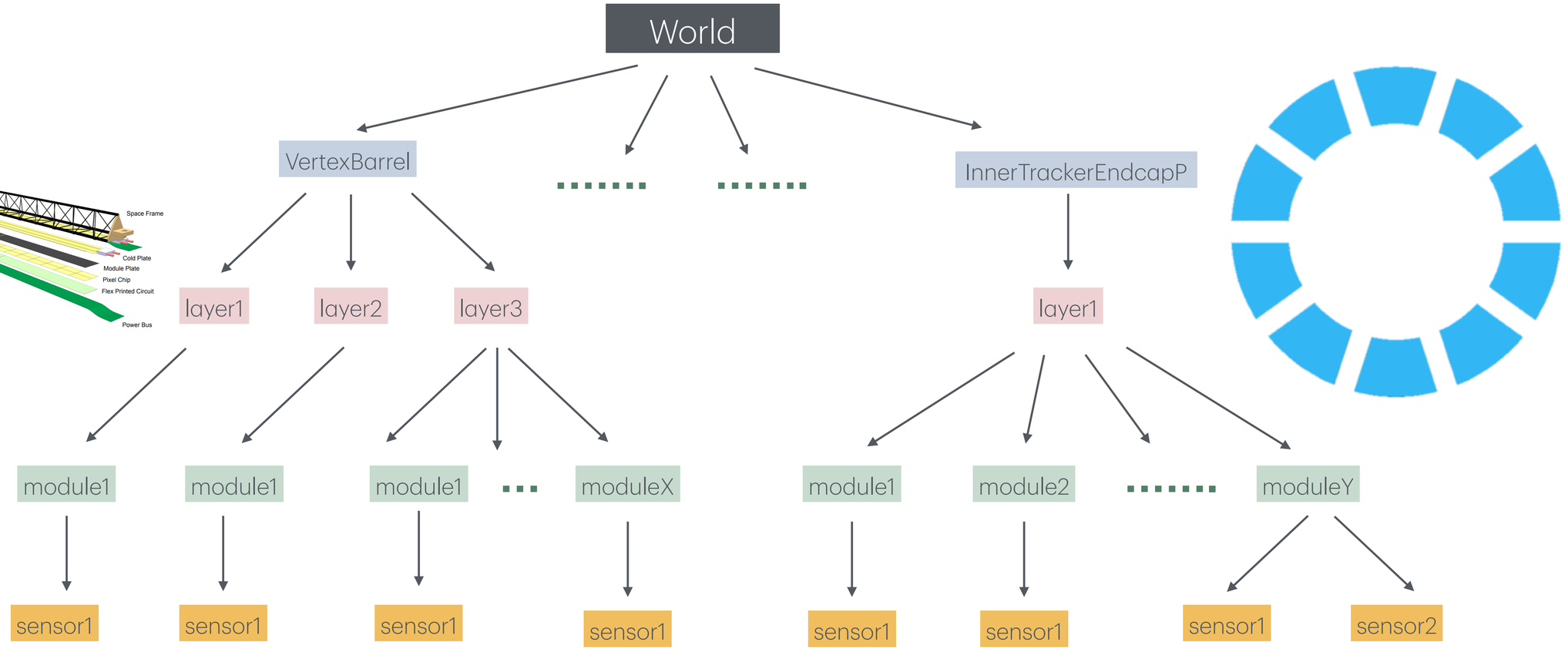
✦ Objective:

- Inject the noise hit of randomly selected sensor pixel
  → active readout unit
- Evaluate the noise effect (fake hit) or tracking performance



Readout Unit & CellID





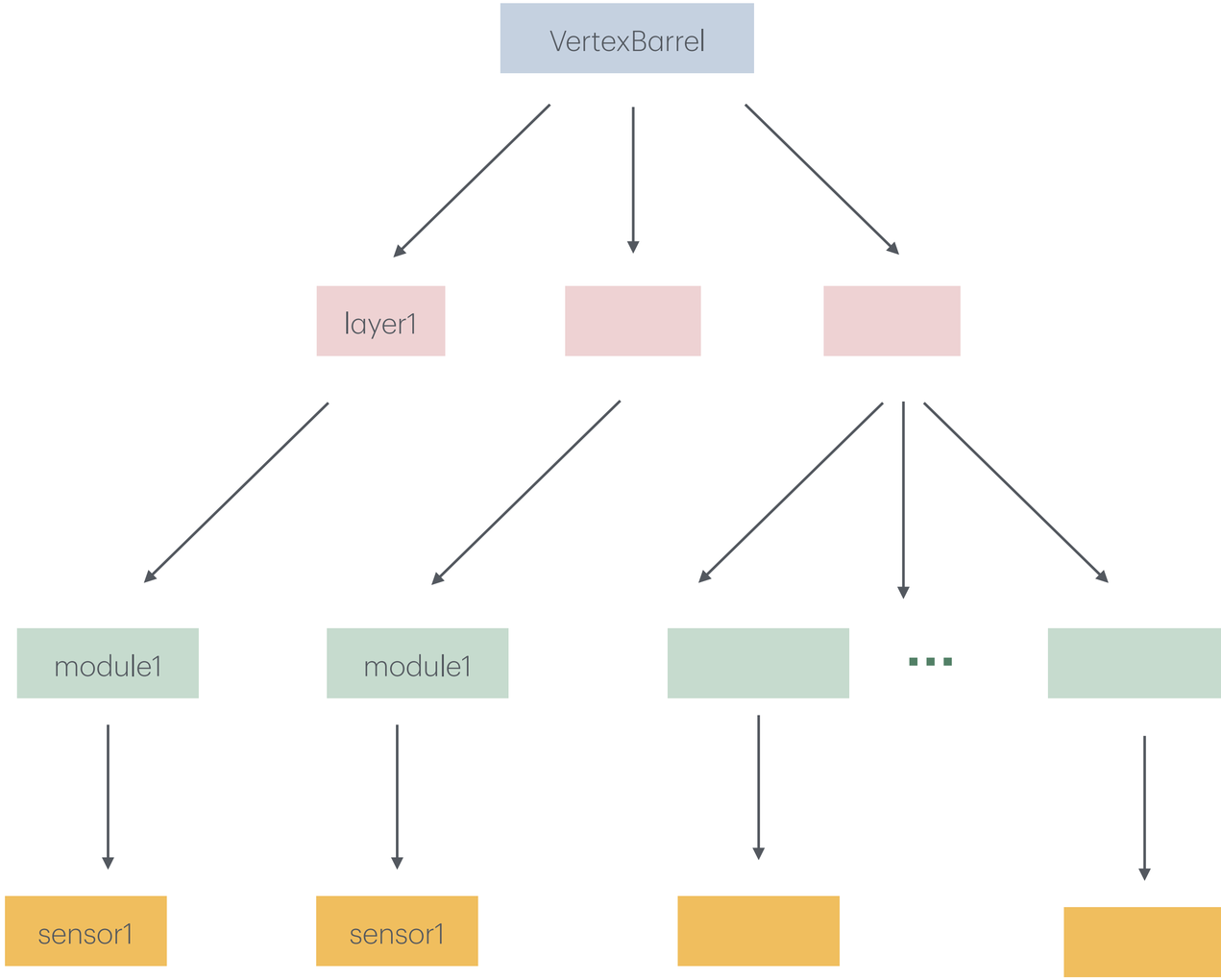
- **Cell: smallest unit of the system (detector component) matching to individual readout unit (segmentation)**
- **CellID: unique ID composed detector structure + readout structure given by 64bit**
  i.e. For silicon barrel detectors: system:8 layer:4 module:12 sensor:2 x:16 z:16

Gathering Geometry Information

- **Goal: gathering geometry information of each system with minimum hardcoding input**
- **Input:**
  - DD4Hep handler (detector info) accessible in EICrecon via const dd4hep::Detector* m_dd4hepGeo through Service<DD4hep_service> m_geoSvc
  - Hit collection (from simulation) created for each system
- **Starting from “system”, identify all unique paths to individual sensitive component via recursively exploring the detector geometry tree**
- **Substructure of detector dynamically determined; fields and their effective ranges**

system:8 layer:4 module:12 sensor:2 x:16 z:16

$2^{32} \rightarrow O(100\text{-}1000)$



Gathering Local Segmentation Fields

✦ **Goal: automatically discover the valid range of local coordinates for a single detector component**

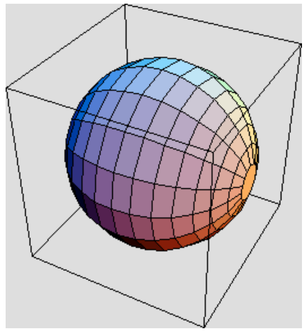
✦ **From minimum sensitive volume (assuming that system has single type of detector component), find its bounding box → physical x, y, z dimensions we need to scan**

✦ **Create a sample of valid cellIDs by generating points inside the sensor and checking that each point corresponds to a real cell →(min, max) range for each local segmentation field**

- Cartesian grid: straightforward case; evenly spaced 3D grid of test points inside the bounding box
- Optimized scanning strategy for “Cylindrical” and “Polar” grids
- Robustness: segmentation type is unknown or highly complex, it “throws" thousands of random points into the bounding box

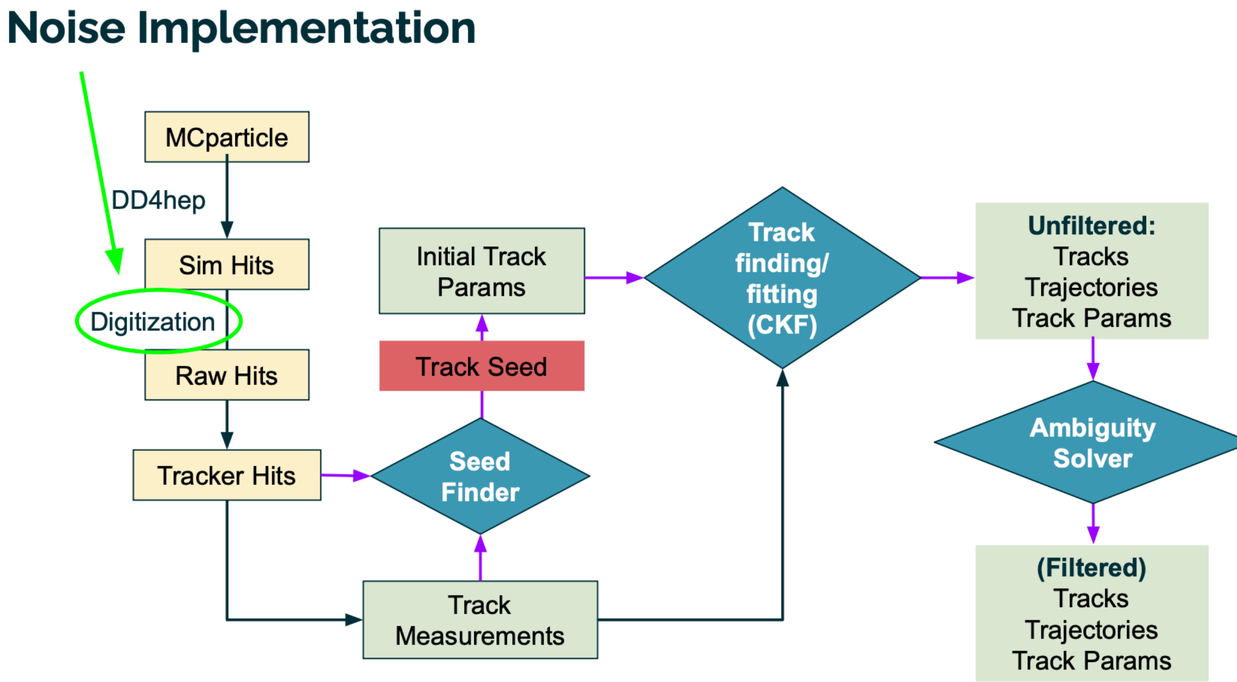
system:8 layer:4 module:12 sensor:2 **x:16 z:16**

$2^{32} \rightarrow O(2^{10\text{-}14})$



Inject Noise to Hit Collection

- **CellID Generation: pick a random detector component and local coordinate fields within the valid range**
- **CellID Validation:**
  - **Uniqueness Check**: verifies the cellID doesn't already exist in the event (either as a real hit or a previously generated noise hit)
  - **Position Check**: ensure the combination of random fields corresponds to a real, physical location in the detector
- **“Random Noise” Factory called after digitalization (SiliconTrackerDigi_factory), before hit reconstruction (TrackerHitReconstruction_factory)**



Random Noise Factory

```
  // Digitization
  app->Add(new JOmniFactoryGeneratorT<SiliconTrackerDigi_factory>(
      "SiBarrelVertexRawHits", {"VertexBarrelHits"},
      {"SiBarrelVertexRawHits", "SiBarrelVertexRawHitAssociations"},
      {
          .threshold = 0.54 * dd4hep::keV,
      },
      app));
app->Add(new JOmniFactoryGeneratorT<RandomNoise_factory>(
        "NoisySiBarrelVertexRawHits",              // 1. The name of the plugin instance
        {"SiBarrelVertexRawHits"},         // 2. The input collection tag
        {"NoisySiBarrelVertexRawHits"},    // 3. The output collection tag
        {.addNoise = true, .n_noise_hits_per_system = 433, .readout_name = "VertexBarrelHits"},                                    app));

  // Convert raw digitized hits into hits with geometry info (ready for tracking)
  app->Add(new JOmniFactoryGeneratorT<TrackerHitReconstruction_factory>(
      "SiBarrelVertexRecHits", {"NoisySiBarrelVertexRawHits"}, {"SiBarrelVertexRecHits"},
      {}, // default config
      app));
```

3 input parameters: transparent when “addNoise” set to “false"

```
struct RandomNoiseConfig {
  // Enable or disable noise injection.
  bool addNoise = true;
  // Average number of noise hits to inject per system.
  // This will be used as the mean for a Poisson distribution.
  int n_noise_hits_per_system = 100;
  std::string readout_name="VertexBarrelHits";
}
```

Silicon Tracker Noise Rate (Shujie)

	A	B	C	D	E	F	G	H	I	J
1	noise rate	5.00E-07	per event per pixel							
2										
3		**dim 1 [mm]**	**dim 2 [mm]**	**area [mm2]**	**per RSU**					
4	pixel size	2.00E-02	2.00E-02	4.00E-04	1.00E+06					
5										
6		Use nominal geometry from https://eic.jlab.org/Geometry/Detector/Detector-20240515102931.html								
7	**Detector name**	**Barrel layers**	**R [mm]**	**L [mm]**	**Area [mm]**	**# of pixels**	**# of noise hits**	**Total noise hits per parts**	**This work**	**from Mito**
8	VertexBarrel	L0	36	270	6.11E+04	1.53E+08	76	Inner Barrels	433	433
9		L1	48	270	8.14E+04	2.04E+08	102			
10		L2	120	270	2.04E+05	5.09E+08	254			
11	SagittaSiBarrel	L3	270	540	9.16E+05	2.29E+09	1,145	Outer Barrels	3,784	3,920
12	OuterSiBarrel	L4	420	800	2.11E+06	5.28E+09	2,639			
13										
14	**Detector name**	**Disks**	**R1 [mm]**	**R2 [mm]**	**Area [mm]**	**# of pixels**	**# of noise hits**			
15	InnerTrackerEndcapN	E-Disk1	36.76	230	3.24E+05	8.10E+08	405	Electron disks (Negative)	6,163	5,910
16	MiddleTrackerEndcapN	E-Disk2	36.76	430	1.15E+06	2.88E+09	1,442			
17	OuterTrackerEndcapN	E-Disk3	36.76	430	1.15E+06	2.88E+09	1,442			
18		E-Disk4	40.06	430	1.15E+06	2.88E+09	1,440			
19		E-Disk5	46.35	430	1.15E+06	2.87E+09	1,435			
20	InnerTrackerEndcapP	H-Disk1	36.76	230	3.24E+05	8.10E+08	405	Hadron disks (Positive)	6,130	5,910
21	MiddleTrackerEndcapP	H-Disk2	36.76	430	1.15E+06	2.88E+09	1,442			
22	OuterTrackerEndcapP	H-Disk3	38.42	430	1.15E+06	2.88E+09	1,441			
23		H-Disk4	54.43	430	1.14E+06	2.86E+09	1,429			
24		H-Disk5	70.14	430	1.13E+06	2.83E+09	1,414			

Summary and outlook

- ✦ New factory “Random Noise” generation is implemented in EICrecon
- ✦ PR: Random Noise injection factory with its usage in tracker classes #1988
  https://github.com/eic/EICrecon/pull/1988
- ✦ For silicon tracker, noise rate calculated by Shujie is implemented
- ✦ Further study - impact of noise on tracking performance - can be done by comparing standard EICrecon output with “addNoise” on and off