Simulation of pixilated sensor digitization in LCSIM framework

Nick Sinev, University of Oregon

Topics:

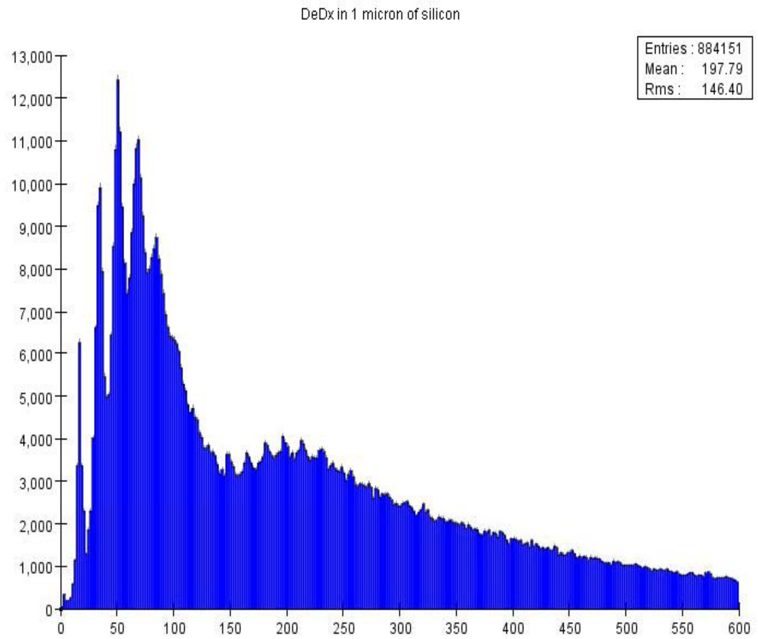
- 1. Goals of sensor digitization simulation
- 2. What and how was simulated
- 3. Sensor configuration
- 4. Simulation outputs
- 5. Simulation speed
- 6. Using lookup tables
- 7. Some examples
- 8. What is not completed yet
- 9. Conclusions

Goals of sensor digitization simulation

- Digitization simulation means detailed simulation of ionization generation in the sensitive layer of the sensor, charge collection, signals formation and signal processing.
- Such simulation has 2 major goals:
  - Provide realistic estimation of the real detector performance.
  - Provide tool for optimizing sensor parameters and comparison of different sensors from the point of view of achieving physics goals

What and how was simulated- ionization process

- For the energy loss of the charged particle in the sensor I have used single collision energy loss tables, calculated by Hans Bichsel ionization loss package. I have simulated every ionizing collision, as there are only 4 such collisions per 1 micron path in silicon in average. Our sensors are few microns thick, so usage of Landau distributions would not be correct.



Example of energy loss distribution
For 1 micron thick silicon – output from my energy loss simulation

What and how was simulated- charge collection

- In principle, I have simulated the movement of every charge carrier, generated in the ionization process through sensitive layer of silicon to the point where it is collected by charge collection electrode. (Of course in CCD pixel charge is not collected by electrode, so in that case I have simulated process of charge collection in the CCD buried channel, which is, in that case “collector”).

Illustration - charge movement in chronopixel device .

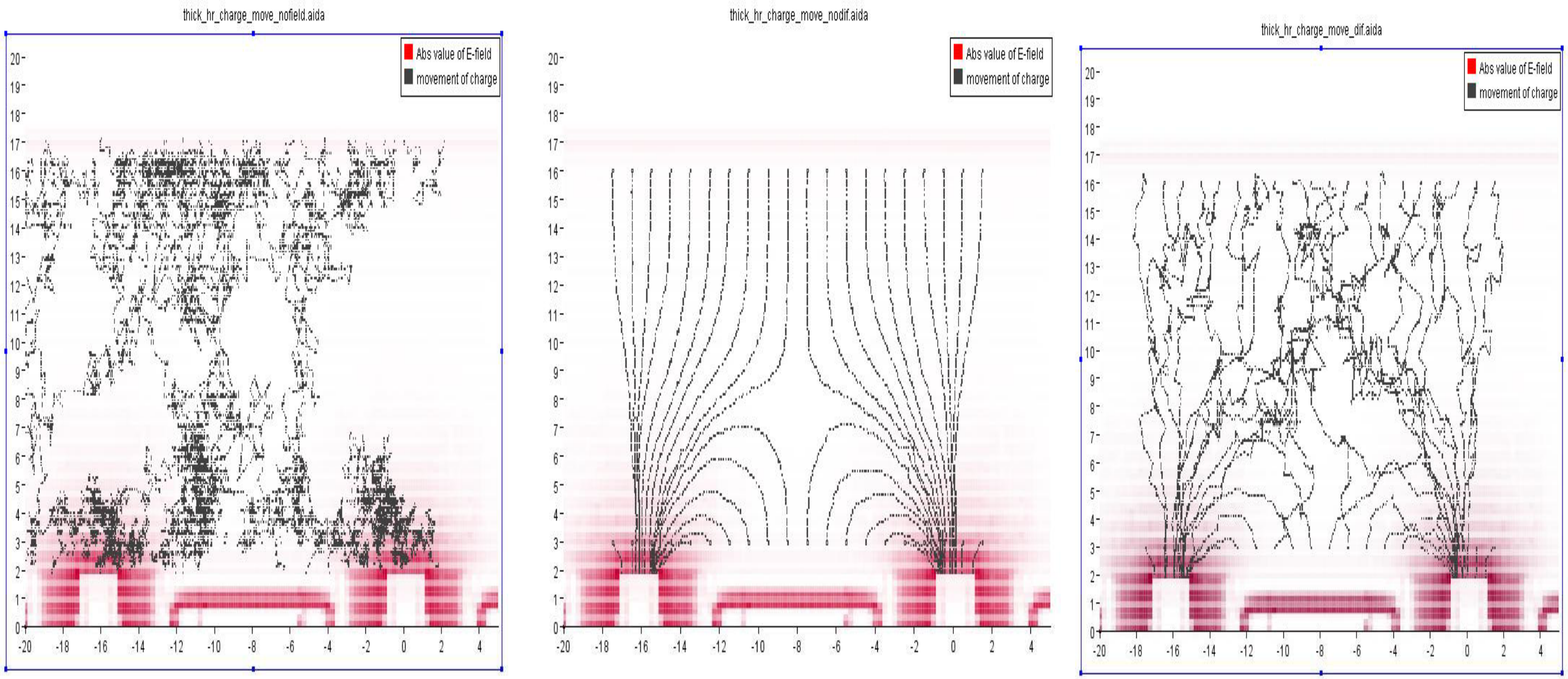


Illustration of the electric field effect on the charge collection in silicon sensor:
On the left picture only diffusion is simulated, in the middle charge is moving only by electric forces, and the right picture shows how it moved in our simulations

Pixel configuration

- PixelConfiguration class defines everything needed to simulate charge carrier movement:
  - List of collecting, absorbing and reflecting regions, which can be boxes, cylinders spheres of cones.
  - Properties of silicon (doping, mobility, diffusion length and so on)
  - Electric and magnetic fields

Simulation outputs

- Currently software is developed to the point, where simulation of every track segment in the sensitive layers produces a list of ActivePixel objects. Each such object contains following information:
  - Row and column offset relative to pixel to each center of segment is projected.
  - Total charge collected by this pixel
  - Array of values of charge collected in time bins (allows to simulate signal shape)

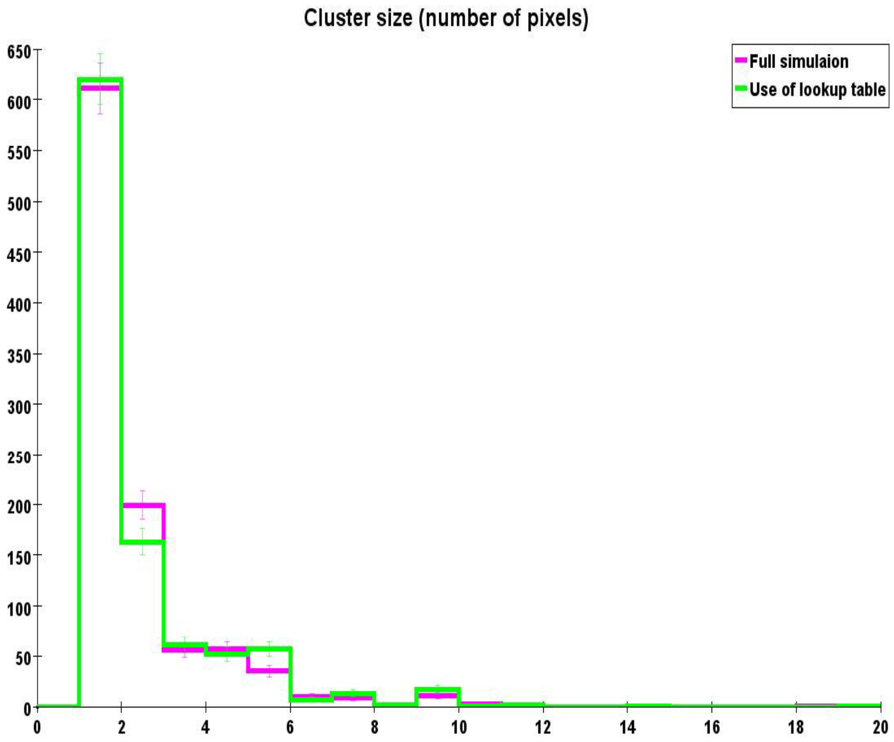
Simulation speed

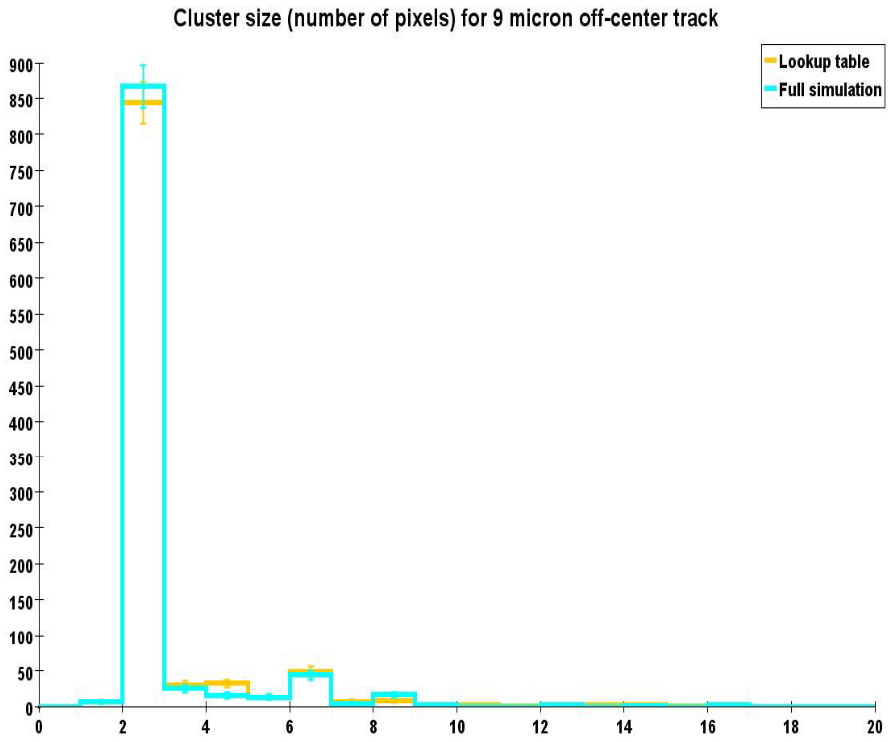
- On my Dell laptop (1.83 GHz Intel T2400, 2 GB memory) simulation of 20 micron track segment takes about 2 seconds. Typical ILC event without backgrounds have ~100 pixel hits, so it would take about 3 minutes per event. It is slow enough, but adding background hits (and we may have thousands of them) makes it too slow. Notice, however, that speed problem arises only when dealing with un-depleted detector. If there is even very weak electric field, charge movement simulation is very fast.

Using lookup table

To increase simulation speed we can use pre-generated tables of probabilities to be collected by particular pixel and travel time for charge generated in any point inside sensitive layer. Such table can be made for any pixel parameters an be stored on disk. Generation of such table takes a lot of time (depending on grid of sampling points dimensions and statistics). For example, generating table with 1 micron grid with 10000 probe charges in each point took 10 hours on my laptop. But use of lookup table increases simulation speed by factor ~100, so now I could simulate 10000 track segments in 5 minutes. This is still not too comfortable (~10 large background events/h) but this will be not a major contributor in processing time for such events. Pattern recognition for 10000 hits will take a while also!

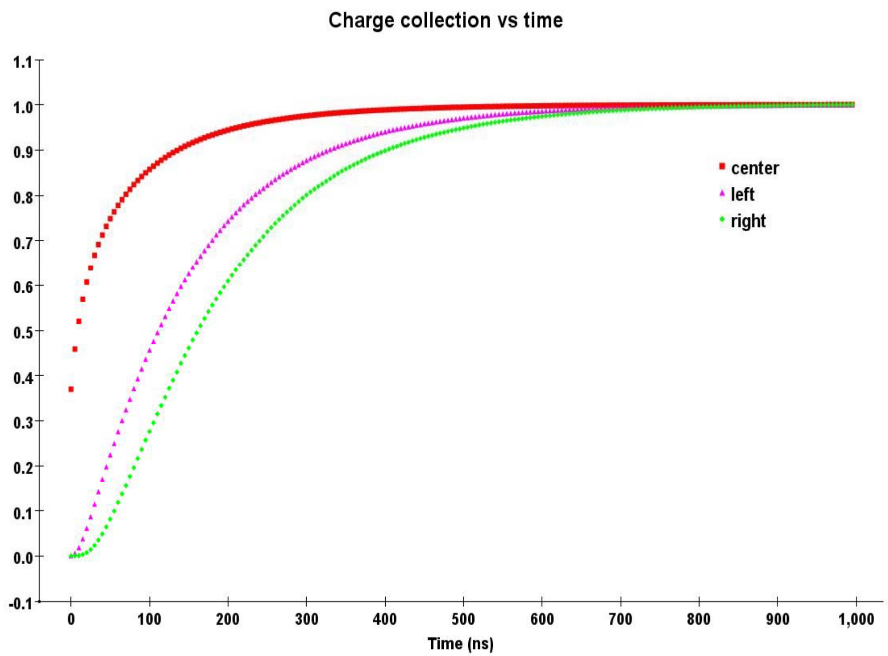
Some examples - 1

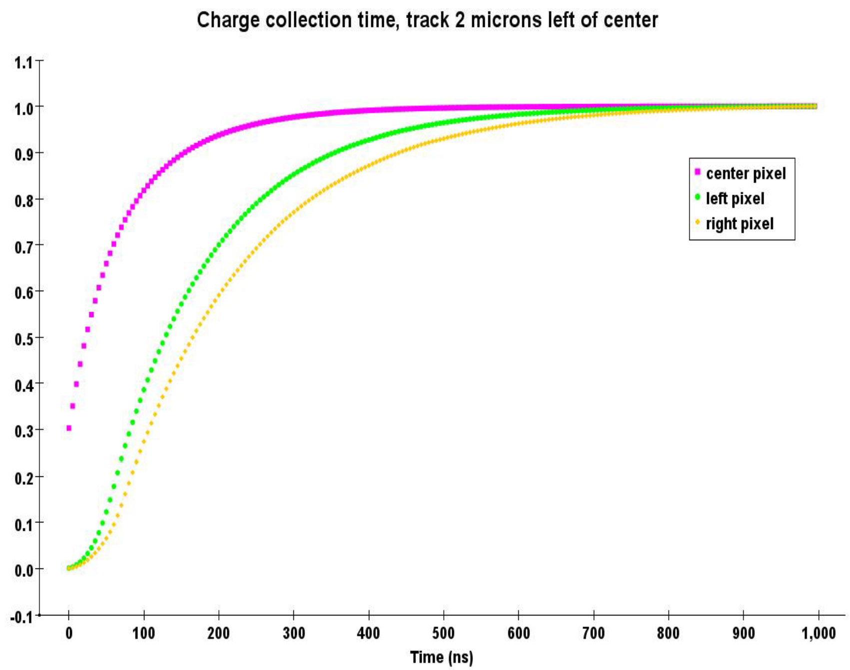




Cluster size for tracks close to pixel center (left) and close to pixel edge (right)

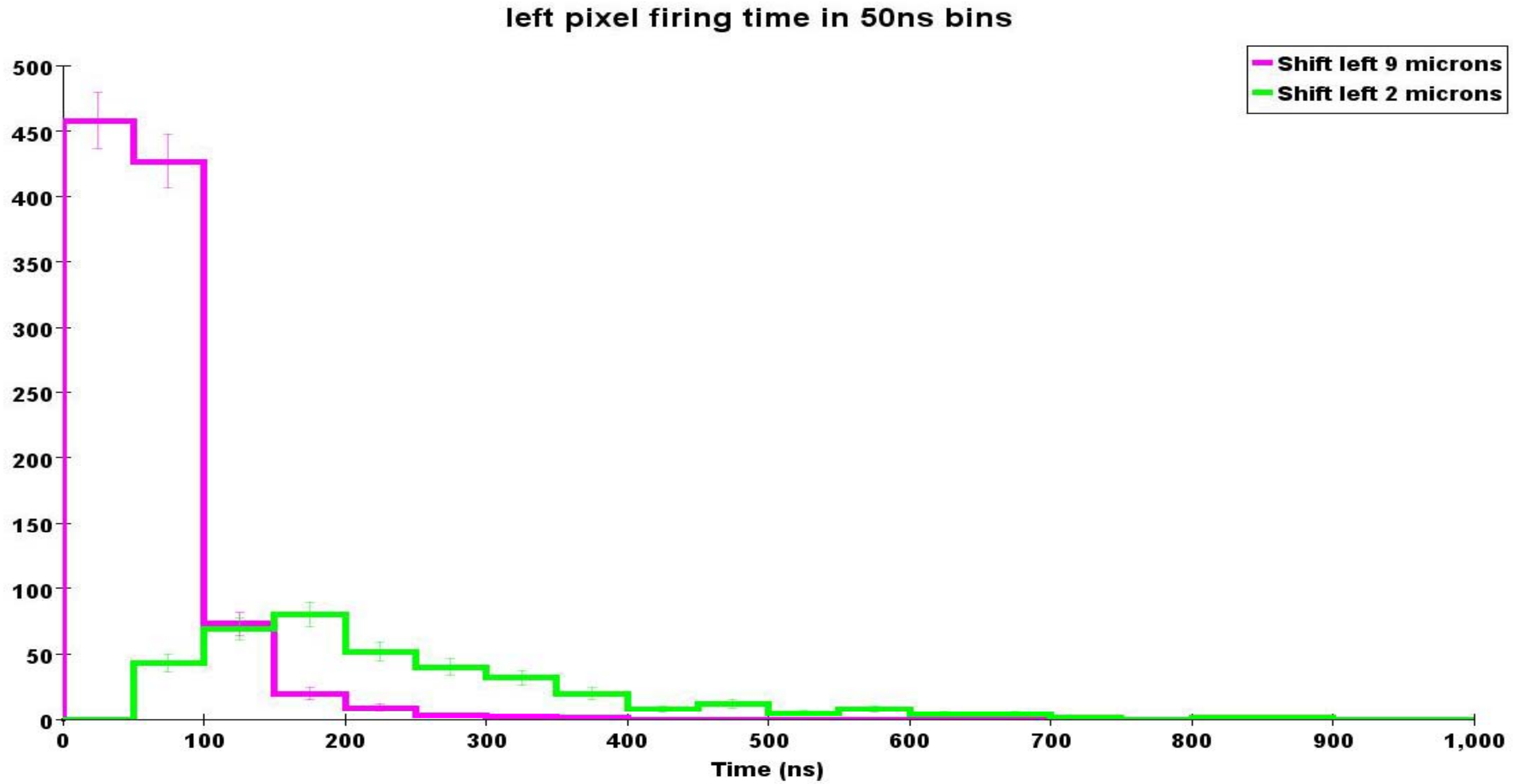
Some examples - 2





Charge collection vs time in central, left and right of center pixels simulated using lookup table (left) or full charge movement simulation (right)

Some examples - 3



Difference in discriminator firing time in two neighboring pixels – when track is close to the center of one pixel (green) or close to the border between pixels (magenta).

What is not completed yet.

- Electronics simulation. It may be simple if we want ADC outputs or discriminator firing time, but may be complicated if we want simulate Marty's Short Column CCDs. And correct electronics simulation for many type of detectors is more important than charge collection simulation.
- Cluster finding, processing and forming TrackerHit objects. Tim Nelson is doing generic package for such job, my simulation output can be easily plugged into it.

What is not completed yet – but not by myself

- Vertex detector is unique among other parts of ILC detector in the sense that many technology choices do not provide good timing resolution. To simulate effect of the events pileup we need the ability to read multiple background events, assigning different time to their hits (different bunch crossings). It was easily done in old lcd.event framework, but is not available in new org.lcsim. If we are serious about vertex detector ability to work in high backgrounds environment, this issue need to be resolved.

Conclusions

- PixSim package will give us a tool for detailed simulation of signals in any pixilated detector.
- When used in table lookup mode this tool is fast enough.
- Full simulation of charge movement allows very detailed study of different types of pixilated detectors, and generation of lookup tables for faster simulation.
- Complete package will be deployed soon, and it will be extendable for different detector types by adding specific pixel and electronics configuration modules.