Simulation of Pixel Detectors for ILC

Nikolai Sinev[*]
[†]

University of Oregon, Eugene, Oregon

E-mail: sinev@slac.stanford.edu

A group of SLAC (SLAC National Accelerator Laboratory) scientists are working on developing a software package for the simulation of the reconstruction and physics analysis of the events generated in the particle collisions of the projected International Linear Collider. In such a simulation the detailed response of the parts of the Linear Collider Detector need to be simulated. Since 2003 the author worked on the detailed simulation of the response of the vertex detector sensors to charged particles. It started as a detailed simulation of the CCD signals (see [1]), and later a universal package for the simulation of any type of pixel detector was developped. This publication emphasizes the results of simulations for the suggested pixel sensor design for one of the possible International Linear Collider options, the CLIC [2] accelerator. Many other pixel detector variations were also investigated using this package. Detailed realistic simulation of the vertex detector signals is important for understanding potential problems in the collision events reconstruction. It is also essential in optimizing sensor design to meet required performance parameters and in the evaluation of different sensor options (Fast CCDs, CMOS pixels, Fine Pixels CCDs and so on - there are more than 10 potential sensor technologies).

The 20th Anniversary International Workshop on Vertex Detectors
June 19 - 24, 2011
Rust, Lake Neusiedl, Austria

[*]Speaker.

[†]Many thanks to Norman Graf, Jeremy McCormic and Tony Johnson from SLAC for their help in resolving software issues

1. Introduction

SLAC physicists have developed the physics analysis software tool Java Analysis Studio (JAS). This tool allows users to integrate their own analysis code with the large suite of event reconstruction code developed by the team. This event reconstruction and analysis package is written in Java programming language and can be downloaded together with JAS by anyone interested. The package has the name *org.lcsim*. In its framework original event information is read from the file in *lcio* format. The package can be used for processing of data from real experiments. However, so far it has been used for processing of simulated events only.

The events are simulated (generated) by GEANT software, using physics event generators like *Pythia* and the detector description. The GEANT output files are then converted to *lcio* (Linear Collider Input-Output) format, which can be read by *org.lcsim* software. In the *lcio* files information about points where particles are crossing detector elements do not include measurement errors introduced by detector elements due to limited resolution. In the *lcio* file information is written in the objects, called *SimTrackerHits*. Co-ordinate information in *SimTrackerHits* is the information about the "real" point where the particle hits the detector element. To be able to do realistic simulation of the reconstruction of real events, detector measurement errors should be introduced. The objects, which are the result of the application of detector properties to *SimTrackerHits* are called *TrackerHits*. The *TrackerHits* objects are similar to what can be obtained from the real detector, and can be considered as a universal interface to reconstruction software. When the real detector is built, signals coming from that detector will be converted into the *TrackerHits* format and we will be able to use all of the reconstruction and analysis suite used for simulated events without any modifications.

The task of converting *SimTrackerHits* into *TrackerHits*, taking into account particular detector sensor properties is the task of the *Digitization software*. The *pixsim* package described below is part of this software.

2. Description of the package.

2.1 Simulation of electron-hole pairs production.

To convert *SimTrackerHit* information into detector output signals, the amount and positions of electron-hole pairs produced by the ionizing particle in the active media of our sensor (silicon in the case of silicon vertex detector sensors) should be calculated first.

It is well known that in very thin layers of material the fluctuation in the number of ionizing collisions of charged particles passing through does not follow the Landau distribution usually used for thicker layers. To simulate the electron-hole production with realistic fluctuations the algorithm, developed by H.Bichsel [3] is used. With the help of Bichsel calculations the distributions of energy transfer from charged particle to silicon atom electrons in a single ionizing collision is obtained. The simulation of the position of each ionizing collision of particles passing through the active layer of the sensor is performed, and above distribution is used to determine how much energy was transferred to a kicked-off electron in every collision. It is assumed that all such energy is spent by kicked-off electrons for secondary collisions, producing more electron-hole pairs. If the energy of a kicked-off electron is large enough to travel more than a couple of microns, the electron is considerd

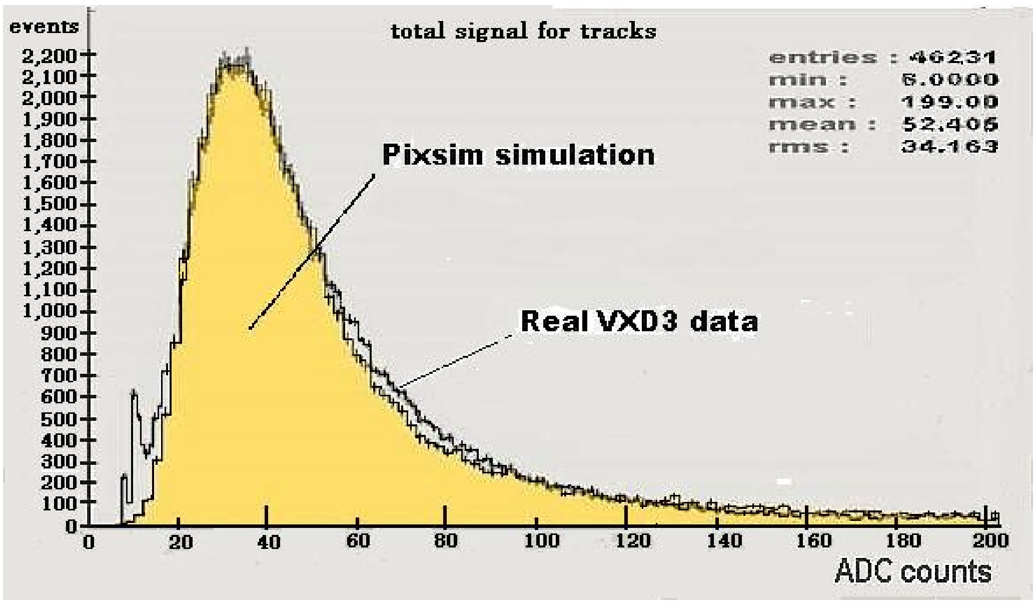

Figure 1: Comparison of real signal amplitude distribution obtained in VXD3 CCD (pixel size 20μm X 20μm, epi layer thickness 20μm) with expected distribution using Bichsel calculations. Slight difference in the width (real data excess in the area of the 60-80 ADC counts) may be explained by the contribution of background signals - for example,δ-electrons from support structures. Data excess in the small signals area may be due to the radiation damage effects in VXD3.

as a δ-electron, and the simulation of the length of its path is performed using the National Institute of Standards electron range tables. The direction of the δ-electron path is random, and it is assumed it spends its energy uniformly along its path.

As a result of these calculations the position of every electron-hole pair generated by a charged particle is obtained.

2.2 Propagation of the charge carriers through silicon.

The next task is to determine which of the generated electrons (or holes) are collected by the sensor charge collecting electrodes. For this we need to know the electric and magnetic fields inside the sensor, and parameters of the charge carriers movement in the silicon - mobility, diffusion coefficient, Lorenz angle. There is a class *Silicon.java*, from which all known information about such parameters can be obtained. The class *CarrierPropagator.java* uses this information and electric and magnetic field maps to simulate the path through the sensor of each individual charge carrier (electron or hole). The geometry of the sensor element (pixel in our case) is given by *PixelConfiguration.java* class, which contains a list of the objects *SensorRegion.java*, describing simple shapes of signal collecting, charge absorbing and charge reflecting regions inside sensor. Class *PixelConfiguration.java* also contains electric and magnetic field maps. The electric field map may be obtained by TCAD simulation of the sensor, or just from expected properties of the electric field in the flat, partially depleted silicon sensor.

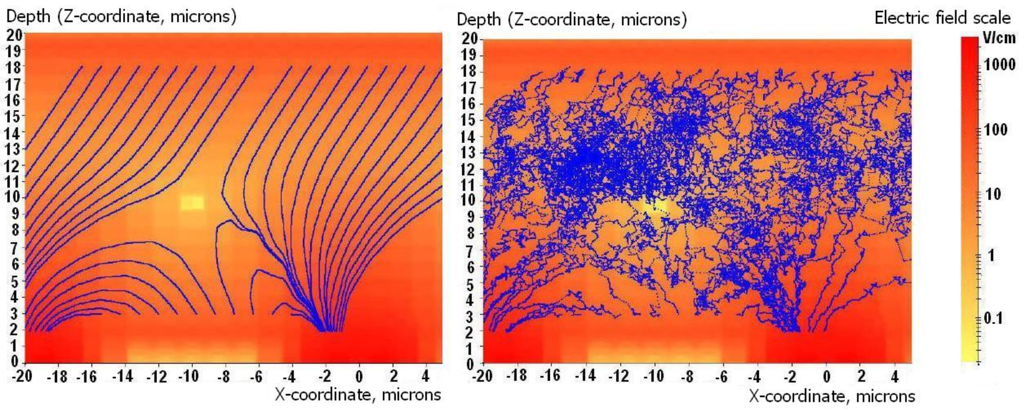

Figure 2: Illustration of the simulation of carriers movement inside silicon sensor. In the left picture I have turned off the diffusion simulation, so the movement is due only to electric and magnetic fields. On the right is a simulation of real movement. The background color on these pictures shows the electric field strength in each point inside sensor. The color scale for the field strength is at the right of the figure. The magnetic field of 5 Tesla is directed along Y axis.

2.3 Signal formation and digitization.

After the movement of every individual charge carrier is completed, the information of the final destination point is recorded together with the time of the arrival. If the destination point is inside the signal collecting electrode, all information about arriving carriers is combined in the object *ChargeTrain* which is the array of the time bins filled with the amount of carriers arriving in each bin. When multiple particles hit surrounding pixels within the sensitive time interval, the pixel record will have a list of *ChargeTrains* rather than a single object.

After all *ChargeTrains* within a given event are generated, the class *IDetectorElectronics.java* processes information contained in the *ChargeTrains* list for the given pixel and simulates electronics output by adding noise and digitizing the signal according to electronics parameters. The class *IDetectorElectronics.java* determines the interface which may be implemented by different electronics models. There are currently 3 implementations of this class: *CCDElectronics.java*, *ChronoPixelElectronics.java* and *FastAnalogElectronics.java*. The last one was designed with the CLIC detector in mind. It generates a time stamp of the signal crossing threshold and the digitized amplitude of the signal.

2.4 Pixels clustering and cluster processing.

Each *SimTrackerHit* usually generates signals in a few pixels. Keep in the mind, that the described software should be able to deal with real life events in which one never knows if observed signals in arrays of pixels were generated by single or multiple particles. The reconstruction software should be able to look through all active (signal above threshold) pixels and detect the clusters of pixels expected to be created by individual particles. In some cases a single particle can generate multiple clusters (energetic δ-electron is produced), and in some cases clusters from different particles may merge into a single cluster (particles near each other). So, the next step the software package performs is the cluster finding. Signals, formed by electronics simulation, are recorded in the single array for each sensor, and the software tries to find clusters of pixels in that array. The cluster finding algorithm in the package assumes that any set of touching active pixels constitutes a cluster. The user can choose how to deal with situations when there is a small, single pixel cluster separated by only one pixel gap from large cluster. The options include merging this single pixel together with the larger cluster considering it one cluster, or to keep it a separate cluster. When choosing the option, the user considers, if the cluster appears inside a tight cluster of clusters which indicates possible dense jet of particles, or is it an insulated cluster.

The clustering algorithm also has the ability to split clusters having few outstanding in the signal amplitude pixels separated by pixels with lower amplitude.

After all the clusters of active pixels are found, the center of each cluster needs to be calculated. Here the center of gravity method or more sophisticated algorithms can be used. If sensor electronics are capable of only determining if the signal in the pixel exceeded threshold without measuring its amplitude, the only option is the center of gravity (Such electronics are referenced as "binary readout". Electronics capable of measuring amplitude are called "analog readout".) However, in the case of analog readout the center of gravity method may not give the best estimation of the co-ordinates of particle hit. It usually gives too much weight to the central pixel, biasing the hit position estimation towards the center of the pixel. So the center calculating algorithm may want to limit the weight of the maximum signal pixel or use a calibration curve of real position versus center of gravity calculation to find best estimation of real position.

2.5 Input for the track reconstruction software.

Having identified all clusters and the co-ordinates of their centers, TrackerHit objects can be created. However, additional information is needed for the measurement error estimation to generate the covariance matrix of position errors. For this the simulation of multiple events may be run to get the distribution of residuals between real hit position, given by *SimTrackerHit* and the position calculated as cluster center position. This distribution will give the measurement error estimation, which can be used for all hits. This step should be done before completing the *TrackerHit* creation code.

We could also try to assign errors based on the cluster shape and measured signal amplitudes for every individual hit. This also requires running simulations for identifiying dependencies prior to the completion of the code.

3. The speed of the simulation.

3.1 The use of lookup tables to speed up simulation process.

The previously described simulation process is very slow. The simulation of one track hits digitization (5 hits in the 5 layers of vertex detector) can take about 20 seconds. Considering the typical TTbar event contains about 100 tracks, finding a way to speed up simulation is essential. Therefore a faster simulation process using lookup tables was developed. These tables contain the probability for the charge carrier to reach the charge collecting electrode in the pixel where it was generated, or in one of the neighbor pixels, and the propagation time to such electrode. The tables are generated for each point inside a pixel with any pre-defined granularity. We consider 1 μm

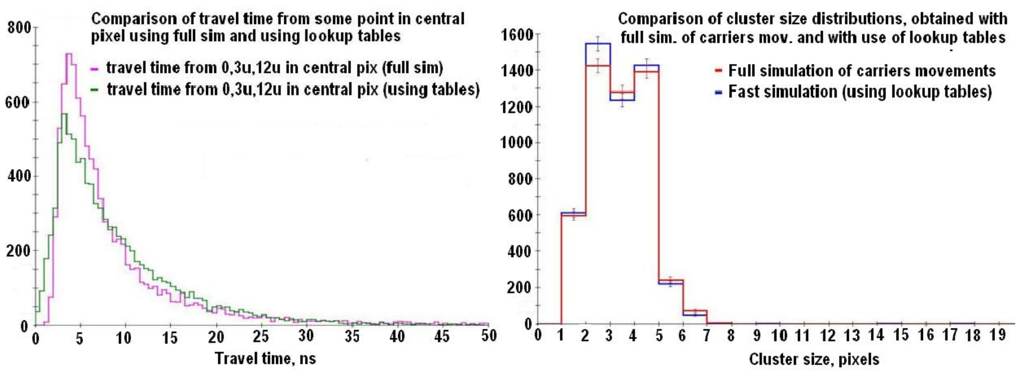

Figure 3: Comparison of distributions obtained with the simulations using the lookup tables (green and blue curves) and the full simulation of carrier movement. Left plots show distribution of travel time from some point inside pixel to signal electrode, right - distribution of number of pixels in the pixel cluster, generated by charge particle hit.

spaced grid of points suitable for such tables. Tables are generated during a special software run, using the individual carrier propagation simulation described earlier. This run could take many hours, but after tables are generated, there is no need to simulate the path of every individual charge carrier. First, the carrier generating part of the code is run, and then lookup tables are used to quickly find the fate of each carrier - where it goes and in what time.

The use of lookup tables speeds up simulation by a factor of approximately a hundred.

3.2 Comparison of the results using lookup tables with the full simulations results.

As seen in figure 3, the results are close, though some differences could be observed. The differences are especially noticeable in the distributions involving the time of the charge carrier travel. This is because the tables contain only the average value of such time. In reality the distribution of the travel times has a non-Gaussian shape and is difficult to accurately reproduce. We believe the lookup table based simulation is accurate enough for most applications of event reconstruction and analysis.

4. Simulation of the sensor, suggested for the CLIC Vertex Detector.

4.1 Is the sensor fast enough for CLIC?

The monolithic CMOS sensors, the most probable sensor options for future linear colliders, do not have a fully depleted active silicon volume. A large part of charge collection involves carrier diffusion. This process is rather slow, and even for very short distances of travel (few microns) it can take tens and even hundreds of nanoseconds. On the other hand, the time interval between bunch crossings in CLIC is only 0.5 ns. Apparently there is no way we can assign every signal observed in a CMOS sensor to a particular bunch crossing. But if we could determine the time of the particle crossing sensor with accuracy equal to 10-20 bunch crossing intervals, it would be enough to help reduce unrelated hits background in physics event reconstruction. So, can we get time stamping of CMOS hits with 10 ns accuracy?

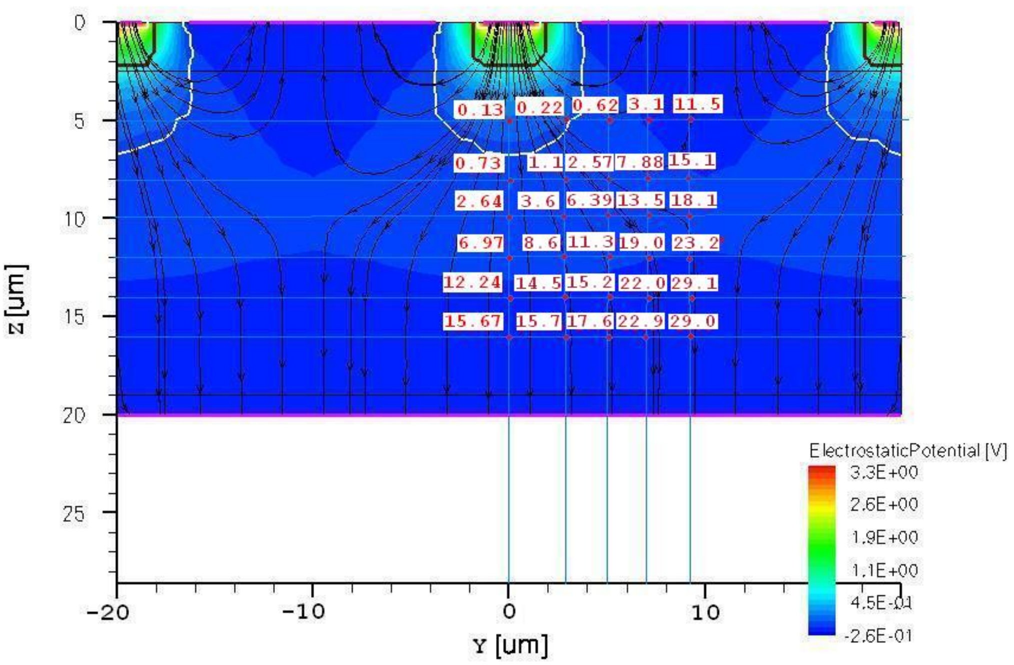

Figure 4: Average travel time of an electron, generated at a given point inside the CMOS sensor volume to the charge collection electrode. Here the CMOS sensor has 18 μm thick high resistivity layer with 3.3V potential in the charge collecting NWELL

As seen from figure 4, most of the tracks generate a large fraction of charge carriers in areas where average propagation time to collection is just few nanoseconds. Of course, a track crossing the detector exactly in the middle between two pixels will have a rather large delay of the signal (more than 10 ns). However, the number of these tracks is not large. So, setting a 10 ns time window for assigning hits to physics events will produce a low percent of inefficiency for hits. This is illustrated in figure 5.

4.2 Optimization of the sensor

Now let us try to answer 2 questions:

1.What number of ADC bits are needed for better single point resolution of the sensor?

2.What is the optimum thickness of epi layer in the sensor?

The answers to these questions can be seen in figure 6 and figure 7

We can see that increasing the number of ADC bits beyond 5 bits does not improve single point resolution of the sensor. In general, 4 bits looks like pretty optimal solution.

Track parameters resolution is certainly better with a thicker layer. Though from figure 5 we see, that hit assignment efficiency for short time windows maybe be even slightly better for thinner epi layer, the larger signal amplitude in thicker layers gives much better single point resolution. Figure 7 shows, that track impact parameter resolution is better for thicker epi layer for any time window. A time window larger than 20 CLIC beam crossing intervals (10 ns) does not improve track reconstruction.

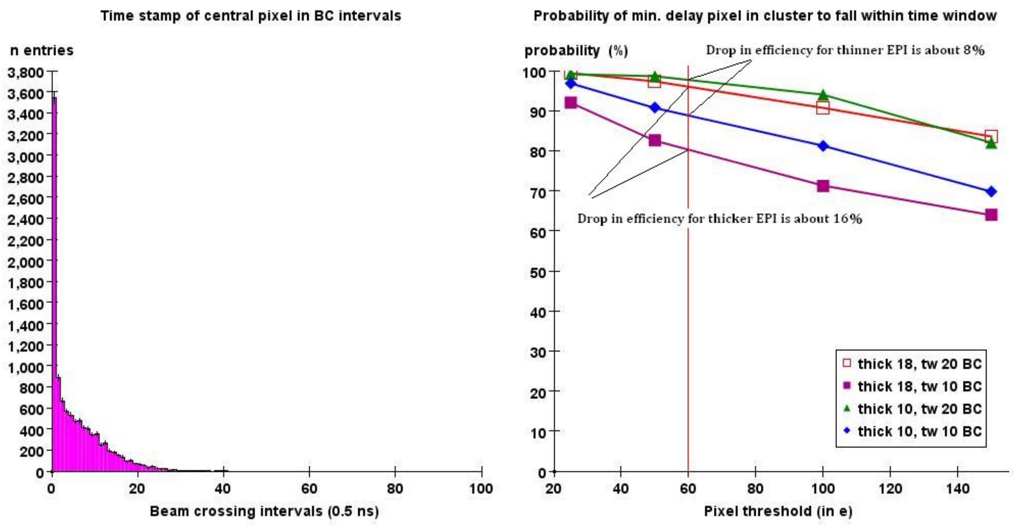

Figure 5: Left plot shows the distribution of time stamps of the hits in 18 μ thick epi layer sensor. X axis on this plot is not in ns, but in CLIC beam crossing intervals, which are 0.5 ns. Right plot shows dependence of hit probability to have a time stamp within time window of 10 or 20 beam crossing intervals on signal registration threshold for 10 μm and 18 μm thick epi layer sensors. A thin layer has the same efficiency as a thick one, because the time stamp is determined only by the fraction of the sensor closest to the charge collecting electrodes.

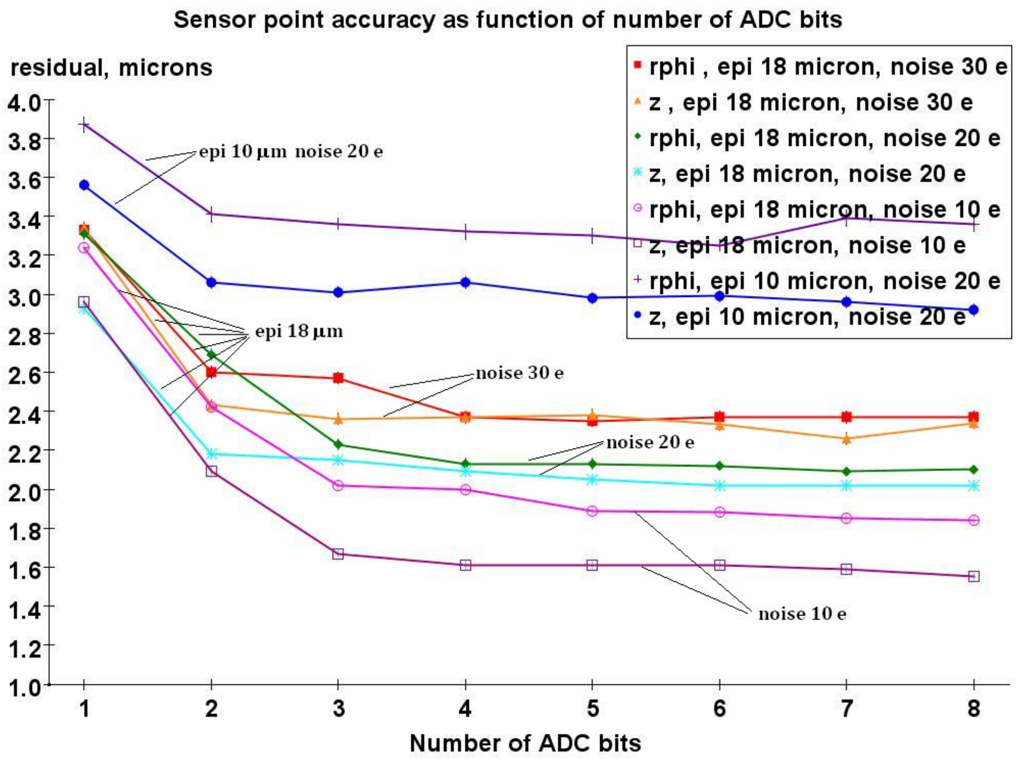

Figure 6: Single point accuracy of the hit reconstruction as function of number of ADC bits in the signal amplitude digitization for different epi layer thickness and different electronics noise level.

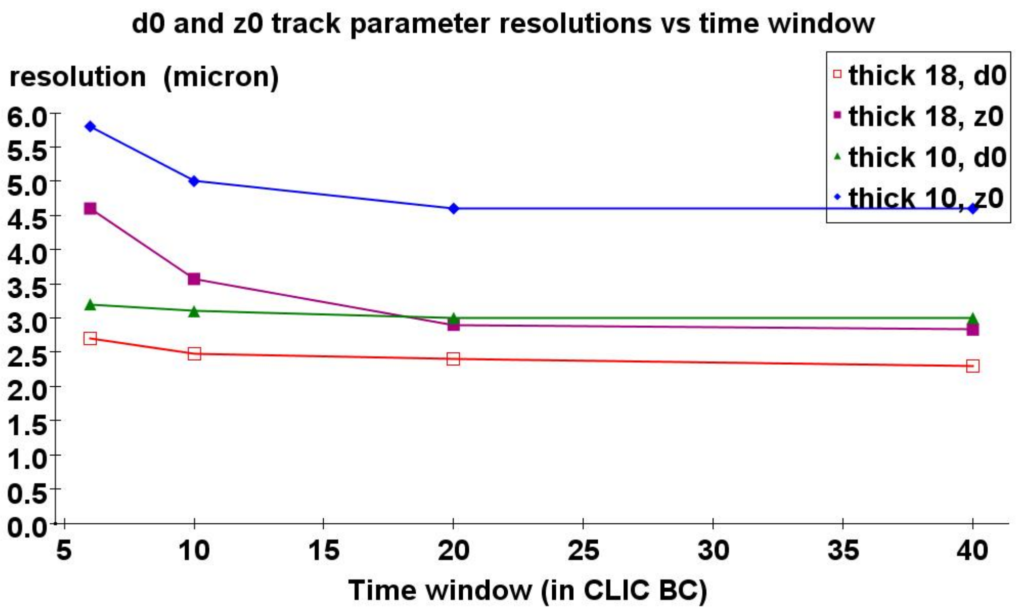

Figure 7: Accuracy of track impact parameters reconstruction as function of time window width (which impacts hit efficiency) for 18μm and 10μm thickness of epi layer. Parameter d0 accuracy is better than z0, because of the very good space resolution of the external silicon tracker in the rϕ direction and much worse in z-direction (because of small stereo layer angle). For the same reason d0 accuracy is less sensitive to layer inefficiency at smaller time windows.

Note we are discussing only effects on track parameter reconstruction, caused by the reduced number of hits on the track due to the hit registration inefficiency. Background hits falling within the time window of the sensor may affect track reconstruction and track parameter estimation also. To study such effects, simulation of the realistic background should be included. Such study is beyand the scope of the given discussion, though all tools for doing it are available in the *org.lcsim* framework.

5. Conclusions

1.The pixsim package is a valuable tool for evaluation of the different pixel detector technologies for the ILC or other future linear colliders (CLIC, muon collider).

2.This package (fast version) can be used as part of a simulation/reconstruction analysis of the future linear collider detector for a realistic detector performance evaluation.

3.The evaluation of a chronopixel type sensor for the CLIC detector demonstrates the sensor is capable of the assignment of hits to a time window of 10-20 CLIC beam crossing intervals with efficiency higher than 90%

4.The ADC resolution of more than 4 bits does not increase sensor single point accuracy. Even a 1 bit ("binary readout") can achieve acceptable single point resolution of about 3.5 μm with pixel size of 20μm X 20μm

References

[1] N.Sinev, *CCD output for full detector simulation in JAS*, LCWS2004,Paris,19-23 April 2004 http://indico.cern.ch/getFile.py/access?contribId=s60t1&sessionId=57&resId=0&materialId=0&confId=a04172

[2] L.Linssen,*CLIC accelerator and detector*, Nederland op CERN, 10.11.2010 http://www.bigscience4business.com/presentatie/CLIC%20accelerator%20and%20detector%20-%20L.Linssen.pdf

[3] H. Bichsel, *Straggling in Thin Silicon Detectors*, *Rev. Mod. Phys.* **60**, 663 (1988).