Update on DIGMAPS and StPixelFastSimMaker

Mustafa Mustafa
Purdue University

We received an “unfinished” version of DIGMAPS documentation

IPHC - CNRS - Université de Strasbourg

A MAPS digitiser
Building a digitiser algorithm for CMOS/MAPS sensors with analog or digital output

Author: Auguste BESSON (abesson in2p3.fr)

Date: March 4^{th} 2012

Abstract

We present an overview of an algorithm which simulates the response of a CMOS/MAPS pixel sensor (CPS) which can be implemented in a complete simulation program. It includes the simulation of the charge deposition, the charge transport in the the digital response (ADC or discriminator) of the sensor. The algorithm is based on a data driven approach and uses extensively the result of test beam data performed by the IPHC group (and collaborators) on various sensors, both with analog or digital output. The algorithm gives as an output a list of pixels hit with their corresponding signal. It is able to take into account the incident angle of the impinging charged particle which crosses the detector. Results and performances of the algorithm are compared with test beam data. It is shown that the multiplicity of the clusters, the resolution and the efficiency of the sensor are correctly reproduced with a precision of the order of 10 %.

DIGMAPS strategy

Energy deposition

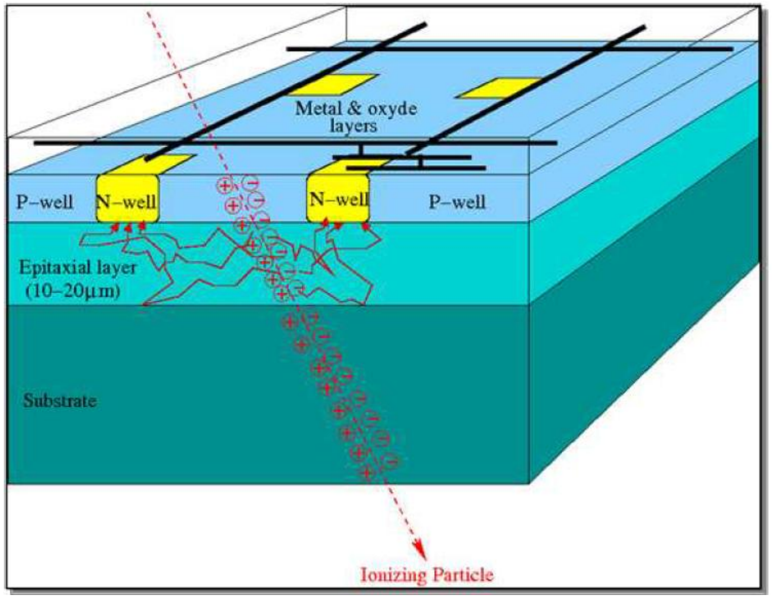
Deposited energy is converted into electron-hole pair (3.6 eV/e-)

Charge transport

Charge transport up to N-well charge collecting diodes through thermal diffusion

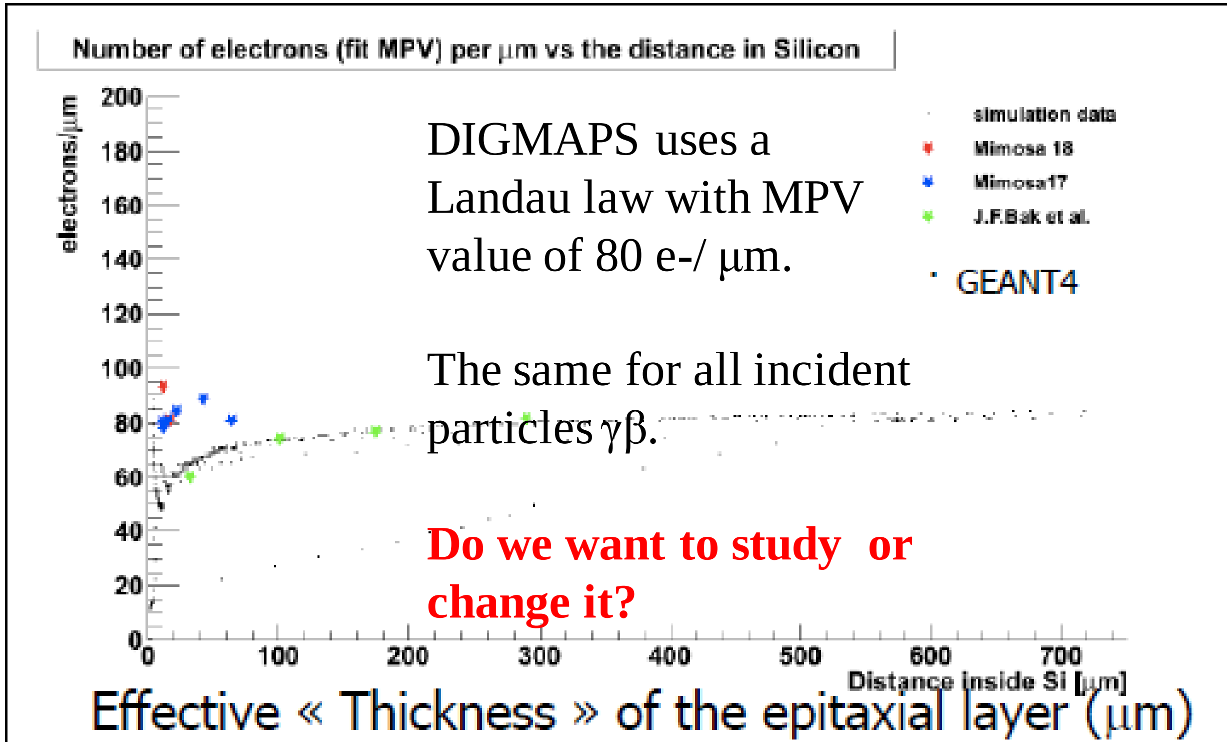
Digitisation

Digital conversion of the collected charge (Descriminator /ADC)



Energy deposition

Auguste showed before that Geant does not compute deposited energy very well for very thin material. **ULTIMATE epitaxial layer thickness is 15µm.**



DIGMAPS uses a Landau law with MPV value of 80 e-/ µm.

The same for all incident particles γβ.

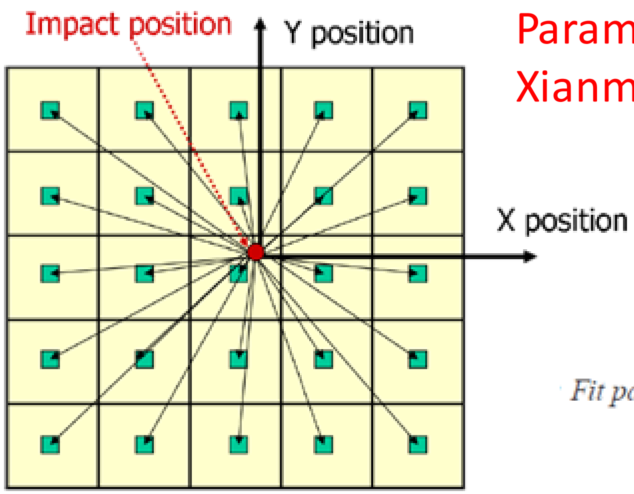
Do we want to study or change it?

One should note that the effective epitaxial thickness is not necessarly close to a measured one since it includes secondary effects like electrons recombination and charge collection inefficiencies, electron creation within the collecting diode, possible slope in the doping profile, reflexion of the charge at the epi/substrate interface, etc. This parameter depends highly on the considered prototype (process technology, resistivity, pitch, didoe sizes, etc.), and has to be adjuted with experimental data. So, since the charge recombination do exists, the charge collection efficiency is not 100 %. In practice it has been measured routinely in the range of 85 % to 95 %, on a lot (∼ 30) of different chips during the last decade.

What is the effective epitaxial thickness for ultimate?
Is it temperature dependent?

Charge Transport (DIGMAPS Core)

- each track has an entry point and an exit point in the epitaxial layer.
- The tracks is divided in N segments of equal sizes. Total created charge Q_{tot} is shared equally between the N segments. $Q_i = Q_{tot}/N$ can be as low as 1 e^-.
- Depending on the x and y position of the segment and on the position of the 25 diodes around, the 25 probabilities that the charge Q_i is collected by the diode j ($j = 1, 25$) are computed thank to the probability density function (see next section). Then a random number is generated and the charge Q_i is deposited in one of the 25 diodes.
- The procedure is repeated for the N segments and the total collected charge on each diode is computed.



Parameters fitted to ULTIMATE-1 beam studies. According to Xianming these should be the same for ULTIMATE-2

First square pixels					
N_0	d_0	σ	Γ	d_1	N_1
0.458	-3.98	13.2	3.99	1.80	6.45
Other pixels					
N_0	d_0	σ	Γ	d_1	N_1
0.117	-1.07	17.5	47.1	-4.64	3.71

Fit parameters of the probability density function obtained from Ultimate sensor test beam data.

The probability density function is obtained from data.

$$p(d) = N_0 \times e^{\frac{-(d-d_0)^2}{2\sigma^2}} + N_1 \times \frac{\Gamma}{(d-d_1)^2 + \Gamma^2}$$

where:

- d = distance between the segment position and the center of the considered diode

Digitisation

At this stage, the analog collected charge on each diode is known. The pixel noise (in electrons units) is then added on each pixel, depending on the measured noise of the considered prototype. This noise is actually not gaussian and an approximation is made here.

According to Marc Winter (IPHC)
Typical Noise performance is 12-14 e- ENC

The following steps consists in simulating the digitisation process, depending on the considered prototype. Some of the prototypes deliver an analog output (actually, the charge is usually encoded on 12 bits), whereas others delivers a pure digital output or delivers an output encoded with a 2 to 5 bits ADC. Furthermore, a zero suppression stage can exist. An ideal reponse of the ADC/discriminator is assumed, so only the dynamic range and the Least Significant Bit (or the discriminator thrshold) is used as an input.

The ADC/digitisation response of the sensor should be smeared due to the temporal noise and fixed patern noise in the digitisation process. One additionnal steps could be added consisting in adding noisy pixels to reproduce correctly the correct fake hit rate obtained with data. These last two steps are not included in the present study, but could be added easily.

I will need to have a closer look at the code and other documents to see how the digitisation is done for ULTIMATE-2.

ULTIMATE specifications input to the DIGMAPS:

xNPxl * yNPxl = 928 x 960
pitch = 20.7 μm
noise per pixel = 13.7 e-
epitaxial thickness = 12.2 μm
Diffusion range & reflexion coefficients.
Temp. 30deg. ?

According to Auguste's comparison of DIGMAPS cluster multiplicity output to beam study data:

algorithm are compared with test beam data. It is shown that the multiplicity of the clusters, the resolution and the efficiency of the sensor are correctly reproduced with a precision of the order of 10 %.

Are we planning to do any other studies? Or just use Auguste's tunning of DIGMAPS?

Integration with StPixelFastSimMaker

```
//simple simulator for perfect hits that just converts StMcPixelHit to StRnDHit
//as of 11/21/08, hits are now smeared with resolution taken from hit error table
UInt_t nh = pixHitCol->layer(k)->hits().size();
LOG_DEBUG << " Number of hits in layer "<< k+1 <<" =" << nh << endm;
for (UInt_t i = 0; i < nh; i++){
  counter++;
  int vid=g2tPix[k].volume_id;
  int layer=vid/1000000;
  int ladder=(vid%1000000)/10000;
  StMcHit *mcH = pixHitCol->layer(k)->hits()[i];
  StMcPixelHit* mcPix=dynamic_cast<StMcPixelHit*>(mcH);
  TString Path("");
  if(k==0) Path= Form("/HALL_1/CAVE_1/PXMO_1/PXLA_1/PLMI_%i/PLAC_1",mcPix->ladder());
  else Path=Form("/HALL_1/CAVE_1/PXMO_1/PXL1_2/PLM1_%i/PLA1_1",mcPix->ladder());
  //LOG_DEBUG<<"mc pixel hit location x: "<<g2tPix[k].x[0]<<"; y: "<<g2tPix[k].x[1]<<"; z: "<<g
  LOG_DEBUG<<"pixel hit layer/ladder is "<<layer<<"/"<<ladder<<" and volume id "<<vid<<endm;
  gGeoManager->RestoreMasterVolume();
  gGeoManager->CdTop();
  gGeoManager->cd(Path);
  //double globalPixHitPos[3]={g2tPix[k].x[0],g2tPix[k].x[1],g2tPix[k].x[2]};
  double globalPixHitPos[3]={mcPix->position().x(),mcPix->position().y(),mcPix->position().z()}
  double localPixHitPos[3]={0,0,0};
  gGeoManager->GetCurrentMatrix()->MasterToLocal(globalPixHitPos,localPixHitPos);
  smearedX=distortHit(localPixHitPos[0],resXPix,100);
  smearedZ=distortHit(localPixHitPos[2],resZPix,100);
  localPixHitPos[0]=smearedX;
  localPixHitPos[2]=smearedZ;
  double smearedGlobalPixHitPos[3]={0,0,0};
  gGeoManager->GetCurrentMatrix()->LocalToMaster(localPixHitPos,smearedGlobalPixHitPos);
  StThreeVectorF gpixpos(smearedGlobalPixHitPos);
  StThreeVectorD mRndHitError(0.,0.,0.);
  //StRnDHit* tempHit = new StRnDHit(mcPix->position(), mRndHitError, 1, 1., 0, 1, 1, id++, kPx
  StRnDHit* tempHit = new StRnDHit(gpixpos, mRndHitError, 1, 1., 0, 1, 1, id++, kPxlId);
  tempHit->setVolumeId(mcPix->volumeId());
  tempHit->setLayer(k+1);
  tempHit->setLadder(mcPix->ladder());
  tempHit->setKey(mcPix->key());
  Int_t truth =0;
  if((k==0)&&(counter<=counter_layer1)) {
    truth = g2tPix[mcPix->key()-1].track_p;
  }
```

This is the core of the current version of StPixelFastSimMaker

The least invasive update will be to replace this part of the code with input from DIGMAPS.

What kind of input?
Fit function? Hash table?
Do we need any DB information? Deposited energy from Geant?

I will look deeper into DIGMAPS code and study our options to integrate it with STAR software.