Simtime
Ver 1.0 April 16th 2007

1- Introduction

Simtime is a Monte-Carlo tool written as a set of Matlab code files that evaluates the time resolution of semi-conductor detectors (eg. Silicon strips or pixels), for a given set of relevant detector and associated electronics parameters. Other fast detectors could be covered at the expense of some modifications in the code. Each section of code is documented, therefore, making any change easy to the user familiar with Matlab.

Three techniques can be compared: leading edge, constant fraction, and pulse sampling. Simtime runs a predefined number of noise realizations through the whole readout process, and returns the time resolution (rms) as well as the amplitude resolution for the case of pulse sampling. Noise contributions are parallel thermal, parallel shot from detector, serial thermal, serial flicker from amplification. Noise contributions can be displayed and compared.

Detector pulse is a triangular current waveform by default with electron leading and hole trailing slopes, as in a N strip over P Silicon detector, as timing is mainly concerned.

Readout electronics is modeled as a low-noise charge amplifier followed by a pulse shaper (CR-RC by default), a discriminator (leading edge or constant fraction), or an analog to digital converter in case pulse sampling is selected; pulse sampling is followed by an iterative least square fit (Cleland and Stern ref [1]) to derive pulse amplitude and delay from the measured noisy signal, assuming the shaping waveform is known.

2- Parameters

Parameters are defined in even parameter files (*.params files). All units are SI.

detector_params

function[telec,thole,C,Ileak,Rbias,T]=detector_params()

telec, thole:	electrons and hole collection times
C:	detector capacitance
Ileak:	detector leakage current
Rbias:	detector biasing resistance
T:	Absolute temperature

electronics_params

function[A,gbw,Cf,Id,gm,nadc,Samp,kf,af,ef,L,Cox,taumin,taumax,taustep,tstep,n]=
electronics_params()

A: charge amplifier open loop gain
gbw: gain bandwidth product
Cf: feedback charge integrating capacitor
Id: current in the charge amplifier input stage
gm: transconductance of the charge amplifier input transistor
nadc: ADC number of bits
Samp=8*(1.38e-23)*300/(3*gm);
srong inversion thermal noise spectral density
Samp=(2/3)*Samp
wak inversion thermal noise spectral density
kf: flicker noise coefficient
af: flicker noise
ef: flicker noise exponent
L: length of the input transistor
tox: input transistor gate oxide thickness
Cox=3.45*1e-11/tox;
input transistor capacitance per unit area
kf=kf/gm^2; flicker noise voltage coefficient
taumin: minimum peaking time
taumax: maximum peaking time
taustep: peaking time increment
tstep=(taumin+taumax)/200;
timestep
n=2*round[(100*taumax/tstep)/2);
simulation number of steps
taumin=floor(taumin/tstep);
minimum peaking time in number of steps
taumax=floor(taumax/tstep);
maximum peaking time in number of steps
taustep=floor(taustep/tstep);
peaking time increment in number of steps

lead_params

function[thresh]=lead_params(C,Cf,A);
th: threshold fraction of a Minimum Ionising Particle signal
Gq=-1/(Cf*(1+1/A)+C/A);
Input amplifier Charge gain
thresh=-Gq*th*25000*1.6e-19;
Absolute leading edge threshold

cfd_params

function[thresh,frac,delay]=cfd_params()
thresh: arming threshold fraction of a Minimum Ionising Particle signal

Absolute arming CFD threshold

frac: constant fraction threshold
delay: CFD delay

sampling_params

function[nmin,nmax]=sampling_params()
nmin minimum number of steps within one peaking time
nmax maximum number of steps within one peaking time

sampling_variables

function[ts]=sampling_variables(tstep,tau,nech)
ts=floor(2*tau/nech)+1;
number of steps of the sampling period

stat_params

function[stat,t0max]=stat_params(tau,tstep);
stat number of runs
t0max=10*tau+round(1e-9*(40+5*0.5)/tstep);
maximum input pulse delay (ILC timing)

3- Running Codes

The following code sets are used through the simulation:

lead
function[fy,snn,maxsig]=lead(i,tau,i0,t0);
Calculates shaper's noisy output for leading edge.

cfd
function[fy,snn,maxsig]=cfd(i,tau,i0,t0);
Calculates shaper's noisy output for an ideal cfd (amplitude fraction threshold).

sample
function[mns,snn,maxsig]=sample(i,nech,tau,i0,t0);
Calculates shaper's noisy output for the pulse sampling method edge (least square fit).

These three codes differ only by the selected input parameters that depends upon the time picking method used.

lead_stat
function[stdt,mean_snn]=lead_stat(tau,stat);

Generates amplitude and time distributions inputs signals (Landau, and random respectively) for the leading edge method. Calculates the leading edge delay statistics.

cfd_stat

function[stdt,mean_snn]=cfd_stat(tau,stat);
Generates amplitude and time distributions inputs signals (Landau, and random respectively) for the ideal cfd method. Calculates the ideal cfd derived delay statistics.

cfd_real_stat

function[stdt,mean_snn]=cfd_real_stat(tau,stat);
Generates amplitude and time distributions inputs signals (Landau, and random respectively) for the real cfd method. Calculates the real cfd derived delay statistics.

sampling_stat

function[stda,stdt,mean_snn]=sampling_stat(tau,nech,nn);
Generates amplitude and time distributions inputs signals (Landau, and random respectively) for the pulse sampling method. Calculates the pulse sampling derived delay statistics.

wave

function[y]=wave(tstep,telec,thole,or,n,i0);
Calculates detector triangular input pulse

iter
function[x,y]=iter(ys,yps,mns);
Used with the pulse sampling method. Cleland ands Stern least square fit algorithm.

ref_delayed
Used with the pulse sampling method. Delays the reference output shaper pulse according to the delay sample from the random distributions in *_stat files above.

simtime
[stda,stdt,snn]=simtime('method');
Runs the whole simulation using 'method' that can be:

lead	leading edge
cfd	cfd
real_cfd	real cfd
sampling	sampling

Returns amplitudes and delays spreads and mean signal to noise ratio, input noise figure, amplitude and delays histograms.

4- Example:

The following set of parameters corresponding to a typical Silicon strips detector serial noise dominated, can be run giving the following time resolutions:

Leading edge

Leading edge:	ns
Cfd:	ns
Real Cfd:	ns
Sampling:	ps

detector_params()

```
telec=5e-9;
thole=25e-9;
C=10e-12;
Ileak=1e-7;
Rbias=1e5;
T=300;
```

electronics_params()

```
A=1e4;
gbw=1e12;
Cf=133e-15;
Id=50e-6;
gm=1e-3;
nadc=12;
Samp=8*(1.38e-23)*300/(3*gm);
Samp=(2/3)*Samp
kf=1e-23;
af=1;
ef=1;
L=250e-9;
tox=5e-9;
Cox=3.45*1e-11/tox;
kf=kf/gm^2;
taumin=30e-9; taumax=30e-9; taustep=30e-9;
```

lead_params(C,Cf,A)
th=0.3;
Gq=-1/(Cf*(1+1/A)+C/A);
thresh=-Gq*th*25000*1.6e-19;

cfd_params()
thresh=0.4;
frac=0.3;
delay=5e-9;

sampling_params()
nmin=12;
nmax=12;

stat_params(tau,tstep);
stat=100;
t0max=10*tau+round(1e-9*(40+5*0.5)/tstep);