

Getting Started

Version 1.30

Author:
Patrick Eckert



KIRCHHOFF-
INSTITUTE
FOR PHYSICS



UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Contents

1	Introduction	2
2	Installation	3
2.1	Linux, Mac	3
2.2	Windows	3
3	Simulation Principle	4
4	Input Parameters	5
4.1	SiPM	5
4.2	Light Source	7
4.3	DAQ	8
5	Tutorials (C++)	10
5.1	Simulate the SiPM Response	10
5.1.1	Using the LightSource	10
5.1.2	Building Your Own PhotonList	10
5.2	Using the DAQ	10
5.3	Interfacing with Geant4	10

1 Introduction

What can the simulation do for you?

The simulation models the response (signal waveform and charge) of a SiPM to an arbitrary light pulse. In order to model a specific SiPM, you have to provide a set of basic parameters for the simulation input. For a precise simulating of the response, the input parameters have to be carefully measured!

What can the simulation NOT do?

The simulation does not simulate the avalanche process itself and therefore cannot predict the basic parameters (e.g. PDE, gain, dark-rate, single pixel waveform,...) and their voltage and temperature dependence. This would require significantly more simulation effort and depends on the details of the doping structure which is typically not accessible. In the GosSiP simulation the avalanche process is parameterised by the basic SiPM parameters, which can easily be measured (or taken from the data sheet).

What are typical application?

- Quickly compare different sensors for a specific application.
- Simulate the performance of a detector system using SiPMs (e.g. SiPM + scintillator).
- Analyse which SiPM parameters / noise sources limit the performance for your application.

2 Installation

2.1 Linux, Mac

- Requirements:
cmake, ROOT (<https://root.cern.ch/>)
- Download GosSiP source code from:
<http://www.kip.uni-heidelberg.de/hep-detektoren/gossip/versions>
- Unzip files:

```
$ unzip GosSiP\_Vx.xx.zip  
$ cd GosSiP\_Vx.xx.zip
```
- Create build directory:

```
$ mkdir build  
$ cd build
```
- Run cmake:

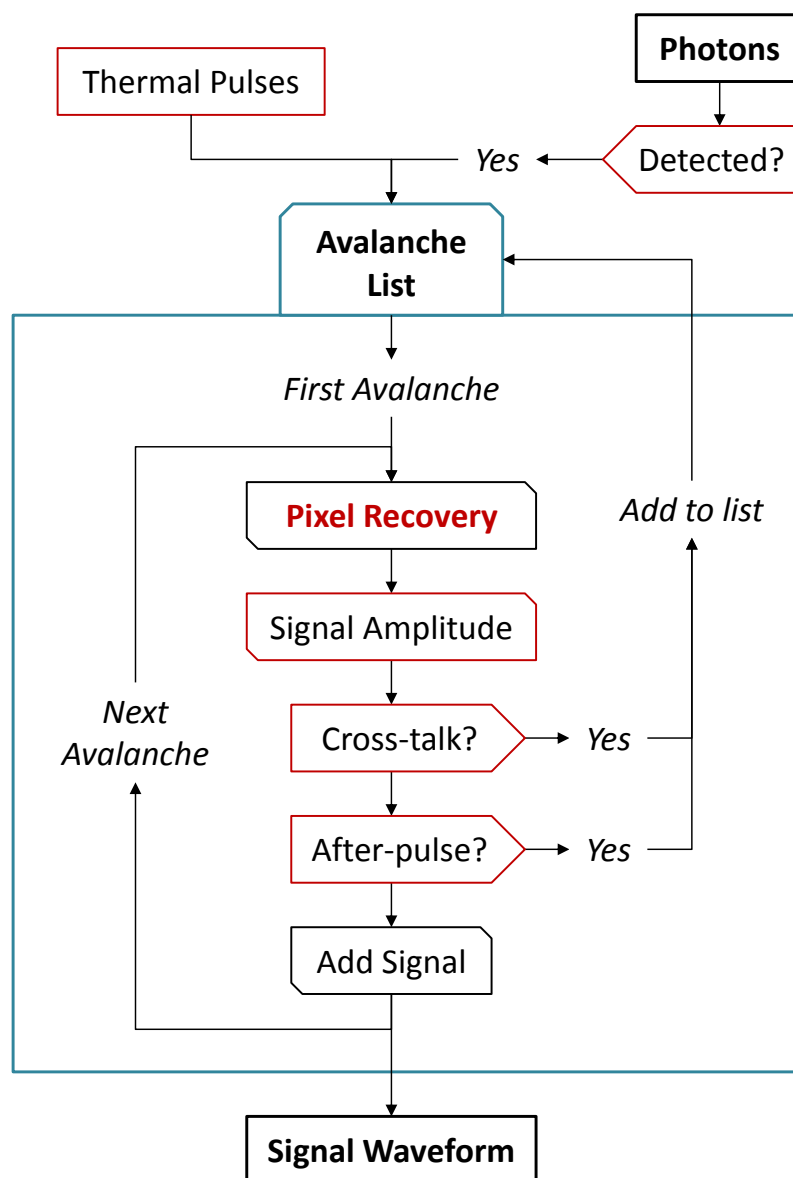
```
$ cmake  
$ make  
$ sudo make install
```
- You're ready to go:

```
$ gossip  
or  
$ gossip 'your_sipm_parameter_file'
```

2.2 Windows

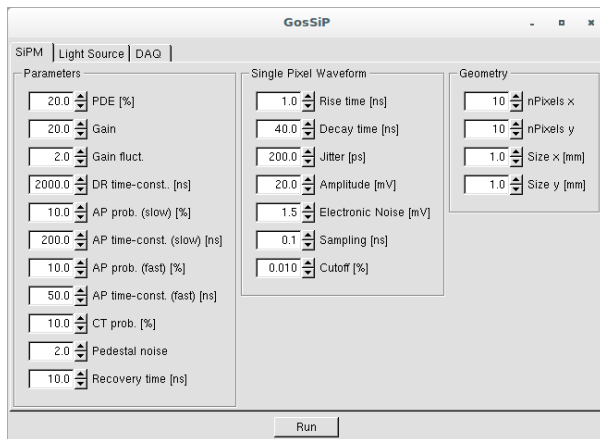
You need to install Cygwin (<https://www.cygwin.com/>) to emulate a Unix shell. Follow the Linux installation instructions using the Cygwin terminal.

3 Simulation Principle



4 Input Parameters

4.1 SiPM



All parameters are members of the sipmMC class.

Parameters

PDE (**PDE**)

Photon detection efficiency (not including cross-talk and after-pulses).

Gain (**gain**)

Gain in arbitrary units.

Gain fluct (**END**)

Gain fluctuations (same units as gain)

DR time-const (**tau_dr**)

Time constant of primary dark-counts, i.e. generated by thermal excitation or tunneling, excluding after-pulses.

AP prob (slow) (**Pap_s**)

Probability to trigger an after-pulse (slow component).

AP time-const (slow) (**tau_ap_s**)

Characteristic time constant for after-pulses (slow component).

AP prob (fast) (**Pap_f**)

Probability to trigger an after-pulse (fast component).

AP time-const (fast) (**tau_ap_s**)

Characteristic time constant for after-pulses (fast component).

CT prob (Pxt)

Probability to trigger optical cross-talk.

Pedestal noise (EN)

RMS width of the pedestal (same units as gain) (does only affect signal charge - not waveform).

Recovery time (tau_recovery)

Pixel recovery time.

Single Pixel Waveform**Rise time (SetPulseShape(double,double))**

Signal rise time - time constant of double exponential function.

Decay time (SetPulseShape(double,double))

Signal decay time - time constant of double exponential function.

Jitter (jitter)

Single pixel jitter.

Amplitude (signalAmp)

Single pixel amplitude.

Electronic Noise (noiseRMS)

Electronic noise (does only affect signal waveform - not charge).

Sampling (SetSampling(double))

Sampling interval of waveform.

Cutoff (SetCutoff(double))

The single pixel waveform is cut-off when the falling edge falls below a certain value. This *cutoff* value is given in % of the peak amplitude.

Geometry**nPixels x (NpixX)**

Number of pixels in x direction.

nPixels y (NpixY)

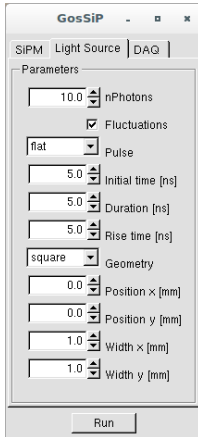
Number of pixels in y direction.

Size x (xSipm)

Size in x direction.

Size y (ySipm)

Size in y direction.



4.2 Light Source

All parameters are members of the LightSource class.

nPhotons ([SetNgamma\(int\)](#))

Mean number of photons in the light pulse.

Fluctuations ([SetFluctuation\(bool\)](#))

Enables / disables fluctuations (Gaus/Poisson) in the number of photons.

Pulse ([SetPulse\(string\)](#))

Time distribution of the photons (*flat*: flat distribution, *gaus*: gaussian distribution, *exp*: exponentially decaying distribution, *exp2*: double exponential distribution).

Initial time ([SetTime\(double\)](#))

Starting time of the light pulse.

Duration ([SetTimeWidth\(double\)](#))

Duration of the light pulse (*flat*: duration, *gaus*: sigma, *exp*: decay time constant, *exp2*: decay time constant).

Rise time ([SetRiseTime\(double\)](#))

Rise time of the light pulse (only for *exp2*!).

Geometry ([SetShape\(string\)](#))

Spatial distribution of the photons (*square*, *elliptic*).

Position x ([SetXY\(double,double\)](#))

Position in x direction.

Position y ([SetXY\(double,double\)](#))

Position in y direction.

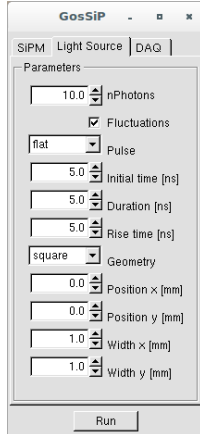
Width x ([SetXYWidth\(double,double\)](#))

Width in x direction.

Width y ([SetXYWidth\(double,double\)](#))

Width in y direction.

4.3 DAQ



All parameters are members of the `daqMC` class.

Waveform ([Scope\(\)](#))

Displays the signal waveform, as measured with an oscilloscope. In addition, the *HitMatrix* is shown: x-axis: pixel number in x, y-axis: pixel number in y, colors: detected photon (blue), dark-count (green), cross-talk (yellow), after-pulse (red).

- **nEntries** Number of waveforms
- **Integration gate** Measurement time
- **Pregate** Time interval simulated before start of measurement

Charge Spectrum ([QDCSpectrum\(int\)](#))

Generates the charge spectrum (Single Photon Spectrum).

- **nEntries** Number of entries
- **Integration gate** Integration gate / measurement time
- **Pregate** Time interval simulated before start of measurement
- **Pedestal** Pedestal value (same units as gain)

Time Spectrum ([TDCSpectrum\(int\)](#))

Generates the dark-rate time spectrum, i.e. the time interval between two consecutive dark-rate pulses.

- **nEntries** Number of entries
- **Pregate** Time interval simulated before start of measurement

Threshold Scan ([ThreshScan\(double,double,double,double\)](#))

Generates the dark-rate as a function of the discrimination threshold.

- **Integration gate** Integration gate / measurement time
- **Pregate** Time interval simulated before start of measurement
- **Threshold start** Start value of discrimination threshold scan
- **Threshold stop** Stop value of discrimination threshold scan

- **Threshold step** Step size of discrimination threshold scan
- **Discri mintime** Minimum time a signal has to be above the threshold to be detected
- **Discri width** Minimum width of the discriminator output pulse

Statistics (**Statistic(int)**) Histogram of the number of fired pixels triggered by photon detection, dark-rate, cross-talk and after-pulsing.

- **nEntries** Number of entries
- **Integration gate** Integration gate / measurement time
- **Pregate** Time interval simulated before start of measurement

Response Curve (**DynamicRange(int,double,double)**)

Mean signal charge and RMS as a function of the number of photons. The contribution from different noise sources are indicated by different colours.

- **nEntries** Number of entries per step
- **Integration gate** Integration gate / measurement time
- **Pregate** Time interval simulated before start of measurement
- **nPhotons max** Maximum number of photons in scan
- **nPhotons step** Step size for the scan

5 Tutorials (C++)

5.1 Simulate the SiPM Response

5.1.1 Using the LightSource

5.1.2 Building Your Own PhotonList

5.2 Using the DAQ

5.3 Interfacing with Geant4