SimSipm

Generated by Doxygen 1.9.1

Chapter 1

Todo List

Member sipm::SiPMAdc::addJitter (std::vector< double > &, const double) const

Maybe better to return by reference here

Member sipm::SiPMProperties::readSettings (std::string &)

Still to implement

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

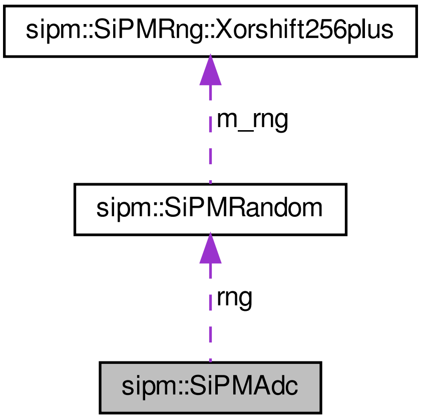
Class Documentation

3.1 sipm::SiPMAdc Class Reference

Class that simulates the operation of an ADC converter.

```
#include <SiPMAdc.h>
```

Collaboration diagram for sipm::SiPMAdc:



Public Member Functions

- SiPMAdc ()=default

  SiPMAdc default constructor.
- SiPMAdc (const uint32_t, const double, const double)

  SiPMAdc contructor with given parameters.
- SiPMDigitalSignal digitize (const SiPMAnalogSignal &) const

  Digitizes an analog signalt to a digital one.

- void setJitter (const double jit)

  Sets jitter parameter.
- void setRange (const double rng)

  Sets range parameter.
- void setGain (const double gn)

  Sets gain parameter.
- void setBits (const double bts)

  Sets number of bits.

Private Member Functions

- std::vector< int32_t > quantize (const std::vector< double > &, uint32_t, double, double) const

  Quantizes a signal using a given number of bits.
- std::vector< double > addJitter (std::vector< double > &, const double) const

  Adds jitter to a signal.

3.1.1 Detailed Description

Class that simulates the operation of an ADC converter.

This class is used to convert a SiPMAnalogSignal into a SiPMDigitalSignal. The signal is quantized using a given number of bits after it is amplified using a given gain.

Author

Edoardo Proserpio

Date

2020

Definition at line 20 of file SiPMAdc.h.

3.1.2 Member Function Documentation

3.1.2.1 addJitter()

```
std::vector< double > sipm::SiPMAdc::addJitter (
            std::vector< double > & signal,
            const double jit ) const  [private]
```

Adds jitter to a signal.

Todo Maybe better to return by reference here

Parameters

signal	Input signal to apply jitter to
jit	Jitter value to apply

Returns

Signal with jitter applied

Definition at line 45 of file SiPMAdc.cpp.

3.1.2.2 digitize()

```
SiPMDigitalSignal sipm::SiPMAdc::digitize (
            const SiPMAnalogSignal & signal ) const
```

Digitizes an analog signalt to a digital one.

Parameters

signal	Input signal to be digitized

Returns

Digitized SiPMAnalogSignal

Definition at line 99 of file SiPMAdc.cpp.

3.1.2.3 quantize()

```
std::vector< int32_t > sipm::SiPMAdc::quantize (
            const std::vector< double > & v,
            uint32_t nbits,
            double range,
            double gain ) const  [private]
```

Quantizes a signal using a given number of bits.

Parameters

v	Vector to quantize
nbits	Number of bits to use for quantization
range	Range to use for the quantization [-range,+range]
gain	Gain in dB to apply before quantization

Returns

Quantized input vector

Definition at line 26 of file SiPMAdc.cpp.

The documentation for this class was generated from the following files:

- /home/edo/UbuntuData/Projects/SimSiPM/SimSiPM/SiPMAdc.h
- /home/edo/UbuntuData/Projects/SimSiPM/src/lib/SiPMAdc.cpp

3.2 sipm::SiPMAnalogSignal Class Reference

Class containing the waveform of the generated signal.

```
#include <SimSiPM/SimSiPM/SiPMAnalogSignal.h>
```

Public Member Functions

- SiPMAnalogSignal ()=default

  SiPMAnalogSignal default constructor.
- SiPMAnalogSignal (const std::vector< double > &wav, const double sampling) noexcept

  SiPMAnalogSignal constructor from a std::vector.
- SiPMAnalogSignal & operator= (const std::vector< double > &&aVect) noexcept

  Move assignement operator from a std::vector.
- SiPMAnalogSignal & operator= (const std::vector< double > &aVect) noexcept

  Copy assignement operator from a std::vector.
- double & operator[ ] (const uint32_t i) noexcept

  Used to access signal elements as if it is a std::vector.
- const double & operator[ ] (const uint32_t i) const noexcept

  Used to access signal elements as if it is a std::vector.
- const uint32_t size () const

  Returns the size of the vector containing the signal.
- const double sampling () const

  Returns the sampling time of the signal in ns.
- const std::vector< double > & waveform () const

  Returns a std::vector containing the sampled waveform.
- const double integral (const double, const double, const double) const

  Returns integral of the signal.
- const double peak (const double, const double, const double) const

  Returns peak of the signal.
- const double tot (const double, const double, const double) const

  Returns time over threshold of the signal.
- const double toa (const double, const double, const double) const

  Returns time of arrival of the signal.
- const double top (const double, const double, const double) const

  Returns time of peak.
- void setSampling (const double x)

  Sets the sampligng time of the signal.
- SiPMAnalogSignal lowpass (const double) const

  Applies a low-pass filter to the input vector.

3.2.1 Detailed Description

Class containing the waveform of the generated signal.

SiPMAnalogSignal.h

This class stores the generated signal as a std::vector<double> representing the sampled analog waveform. It also has some methods that can be used to extract some simple features from the signal.

Author

Edoardo Proserpio

Date

2020

Definition at line 22 of file SiPMAnalogSignal.h.

3.2.2 Member Function Documentation

3.2.2.1 integral()

```
const double sipm::SiPMAnalogSignal::integral (
            const double intstart,
            const double intgate,
            const double threshold ) const
```

Returns integral of the signal.

Integral of the signal defined as the sum of all samples in the integration window normalized for the sampling time. If the signal is below the threshold the output is set to -1.

Parameters

intstart	Starting time of integration in ns
intgate	Length of the integration gate
threshold	Threshold to use for one-suppression

Definition at line 16 of file SiPMAnalogSignal.cpp.

3.2.2.2 lowpass()

```
SiPMAnalogSignal sipm::SiPMAnalogSignal::lowpass (
            const double bw ) const
```

Applies a low-pass filter to the input vector.

Parameters

signal	Signal to filter
bw	Bandwidth for the low-pass filter (-3dB cut-off)

Returns

Signal with filter applied

Definition at line 114 of file SiPMAnalogSignal.cpp.

3.2.2.3 peak()

```
const double sipm::SiPMAnalogSignal::peak (
             const double intstart,
             const double intgate,
             const double threshold ) const
```

Returns peak of the signal.

Peak of the signal defined as sample with maximum amplitude in the integration gate. If the signal is below the threshold the output is set to -1.

Parameters

intstart	Starting time of integration in ns
intgate	Length of the integration gate
threshold	Threshold to use for one-suppression

Definition at line 35 of file SiPMAnalogSignal.cpp.

3.2.2.4 toa()

```
const double sipm::SiPMAnalogSignal::toa (
             const double intstart,
             const double intgate,
             const double threshold ) const
```

Returns time of arrival of the signal.

Arriving time of the signal defined as the time in ns of the first sample above the threshold. If the signal is below the threshold the output is set to -1.

Parameters

intstart	Starting time of integration in ns
intgate	Length of the integration gate
threshold	Threshold to use for one-suppression

Definition at line 75 of file SiPMAnalogSignal.cpp.

3.2.2.5 top()

```
const double sipm::SiPMAnalogSignal::top (
             const double intstart,
             const double intgate,
             const double threshold ) const
```

Returns time of peak.

Time in ns of the sample in the peak If the signal is below the threshold the output is set to -1.

Parameters

intstart	Starting time of integration in ns
intgate	Length of the integration gate
threshold	Threshold to use for one-suppression

Definition at line 99 of file SiPMAnalogSignal.cpp.

3.2.2.6 tot()

```
const double sipm::SiPMAnalogSignal::tot (
             const double intstart,
             const double intgate,
             const double threshold ) const
```

Returns time over threshold of the signal.

Time over threshold of the signal in the integration gate defined as the number of samples higher than the threshold normalized for the sampling time. If the signal is below the threshold the output is set to -1.

Parameters

intstart	Starting time of integration in ns
intgate	Length of the integration gate
threshold	Threshold to use for one-suppression

Definition at line 53 of file SiPMAnalogSignal.cpp.

The documentation for this class was generated from the following files:

- /home/edo/UbuntuData/Projects/SimSiPM/SimSiPM/SiPMAnalogSignal.h
- /home/edo/UbuntuData/Projects/SimSiPM/src/lib/SiPMAnalogSignal.cpp

3.3 sipm::SiPMDebugInfo Struct Reference

Stores MC-Truth informations.

```
#include <SiPMDebugInfo.h>
```

Public Member Functions

- SiPMDebugInfo (uint32_t, uint32_t, uint32_t, uint32_t, uint32_t) noexcept

  Constructor of SiPMDebugInfo.

Public Attributes

- const uint32_t nPhotons

  Number of photons given as input.
- const uint32_t nPhotoelectrons

  Number of photoelectrons (hitted cells).
- const uint32_t nDcr

  Number of DCR events generated.
- const uint32_t nXt

  Number of XT events generated.
- const uint32_t nAp

  Number of AP events generated.

3.3.1 Detailed Description

Stores MC-Truth informations.

This class is used to store some MC-Truth informations about the generated event for debug purposes.

Author

Edoardo Proserpio

Date

2020

Definition at line 13 of file SiPMDebugInfo.h.

The documentation for this struct was generated from the following file:

- /home/edo/UbuntuData/Projects/SimSiPM/SimSiPM/SiPMDebugInfo.h

3.4 sipm::SiPMDigitalSignal Class Reference

Class containing the digitized waveform of the generated signal.

```
#include <SimSiPM/SimSiPM/SiPMDigitalSignal.h>
```

Public Member Functions

- SiPMDigitalSignal (const double sampling) noexcept

  SiPMDigitalSignal default constructor.
- SiPMDigitalSignal (const std::vector< int32_t > &wav, const double sampling) noexcept

  SiPMDigitalSignal constructor from a std::vector.
- SiPMDigitalSignal & operator= (const std::vector< int32_t > &&aVect) noexcept

  Move assignement operator from a std::vector.
- int32_t & operator[ ] (const uint32_t i) noexcept

  Copy assignement operator from a std::vector.
- const uint32_t size () const

  Returns the size of the vector containing the signal.
- const double sampling () const

  Returns the sampling time of the signal in ns.
- const std::vector< int32_t > & waveform () const

  Used to access signal elements as if it is a std::vector.
- const int32_t integral (const double, const double, const int32_t) const

  Returns integral of the signal.
- const int32_t peak (const double, const double, const int32_t) const

  Returns peak of the signal.
- const double tot (const double, const double, const int32_t) const

  Returns time over threshold of the signal.
- const double toa (const double, const double, const int32_t) const

  Returns time of arrival of the signal.
- const double top (const double, const double, const int32_t) const

  Returns time of peak.

3.4.1 Detailed Description

Class containing the digitized waveform of the generated signal.

SiPMDigitalSignal.h

This class stores the generated signal as a std::vector<int32_t> representing the sampled analog waveform passed through an ADC. It also has some methods that can be used to extract some simple features from the signal.

Author

Edoardo Proserpio

Date

2020

Definition at line 22 of file SiPMDigitalSignal.h.

3.4.2 Member Function Documentation

3.4.2.1 integral()

```
const int32_t sipm::SiPMDigitalSignal::integral (
             const double intstart,
             const double intgate,
             const int32_t threshold ) const
```

Returns integral of the signal.

Integral of the signal defined as the sum of all samples in the integration window normalized for the sampling time. If the signal is below the threshold the output is set to -1.

Parameters

intstart	Starting time of integration in ns
intgate	Length of the integration gate
threshold	Threshold to use for one-suppression

Definition at line 15 of file SiPMDigitalSignal.cpp.

3.4.2.2 peak()

```
const int32_t sipm::SiPMDigitalSignal::peak (
             const double intstart,
             const double intgate,
             const int32_t threshold ) const
```

Returns peak of the signal.

Peak of the signal defined as sample with maximum amplitude in the integration gate. If the signal is below the threshold the output is set to -1.

Parameters

intstart	Starting time of integration in ns
intgate	Length of the integration gate
threshold	Threshold to use for one-suppression

Definition at line 38 of file SiPMDigitalSignal.cpp.

3.4.2.3 toa()

```
const double sipm::SiPMDigitalSignal::toa (
             const double intstart,
             const double intgate,
             const int32_t threshold ) const
```

Returns time of arrival of the signal.

Arriving time of the signal defined as the time in ns of the first sample above the threshold. If the signal is below the threshold the output is set to -1.

Parameters

intstart	Starting time of integration in ns
intgate	Length of the integration gate
threshold	Threshold to use for one-suppression

Definition at line 87 of file SiPMDigitalSignal.cpp.

3.4.2.4 top()

```
const double sipm::SiPMDigitalSignal::top (
             const double intstart,
             const double intgate,
             const int32_t threshold ) const
```

Returns time of peak.

Time in ns of the sample in the peak If the signal is below the threshold the output is set to -1.

Parameters

intstart	Starting time of integration in ns
intgate	Length of the integration gate
threshold	Threshold to use for one-suppression

Definition at line 111 of file SiPMDigitalSignal.cpp.

3.4.2.5 tot()

```
const double sipm::SiPMDigitalSignal::tot (
             const double intstart,
             const double intgate,
             const int32_t threshold ) const
```

Returns time over threshold of the signal.

Time over threshold of the signal in the integration gate defined as the number of samples higher than the threshold normalized for the sampling time. If the signal is below the threshold the output is set to -1.

Parameters

intstart	Starting time of integration in ns
intgate	Length of the integration gate
threshold	Threshold to use for one-suppression

Definition at line 62 of file SiPMDigitalSignal.cpp.

The documentation for this class was generated from the following files:

- /home/edo/UbuntuData/Projects/SimSiPM/SimSiPM/SiPMDigitalSignal.h
- /home/edo/UbuntuData/Projects/SimSiPM/src/lib/SiPMDigitalSignal.cpp

3.5 sipm::SiPMHit Class Reference

Class storing informations relative to a single SiPM hitted cell.

```
#include <SiPMHit.h>
```

Public Types

- enum class HitType { kPhotoelectron , kDarkCount , kOpticalCrosstalk , kAfterPulse }

Public Member Functions

- SiPMHit (const double, const double, const uint32_t, const uint32_t, const HitType) noexcept

  Constructor of SiPMHit.
- const bool operator< (const SiPMHit &aHit) const noexcept

  Operator used to sort hits.
- const double time () const

  Returns hit time.
- const uint32_t row () const

  Returns row of hitted cell.
- const uint32_t col () const

  Returns column of hitted cell.
- const uint32_t id () const

  Returns a unique id to identify a hitted cell.
- const double amplitude () const

  Returns amplitude of the signal produced by the hit.
- double & amplitude ()

  Used to modify the amplitude if needed.
- const HitType & hitType () const

  Returns hit type to identify the hits.

Static Private Member Functions

- static const uint32_t makePair (const uint32_t x, const uint32_t y)

  Creates an unique id from two integers (based on Cantor pairing function)

3.5.1 Detailed Description

Class storing informations relative to a single SiPM hitted cell.

This class is used mainly to store informations relative to a single hit on a SiPM cell. Informations stored in thiss class will be used to generate the signal for each SiPM cell.

Author

Edoardo Proserpio

Date

2020

Definition at line 20 of file SiPMHit.h.

3.5.2 Member Enumeration Documentation

3.5.2.1 HitType

```
enum sipm::SiPMHit::HitType  [strong]
```

Used to distinguish between hits generated by different processes

Enumerator

kPhotoelectron	Hit generated by a photoelectron.
kDarkCount	Hit generated by a dark count.
kOpticalCrosstalk	Hit generated by an optical crosstalk.
kAfterPulse	Hit generated by an afterpulse.

Definition at line 25 of file SiPMHit.h.

3.5.3 Constructor & Destructor Documentation

3.5.3.1 SiPMHit()

```
sipm::SiPMHit::SiPMHit (
             const double aTime,
             const double aAmplitude,
             const uint32_t aRow,
             const uint32_t aCol,
             const HitType aHitType )  [noexcept]
```

Constructor of SiPMHit.

Parameters

aTime	Time of the hit in ns
aAmplitude	Amplitude of the hit
aRow	Row of the hitted cell
aCol	Column of the hitted cell
aHitType	Hit type (PE,DCR,XT,...)

Definition at line 11 of file SiPMHit.cpp.

3.5.4 Member Function Documentation

3.5.4.1 operator<()

```
const bool sipm::SiPMHit::operator< (
            const SiPMHit & aHit ) const  [inline], [noexcept]
```

Operator used to sort hits.

Hits are sorted based on theyr time parameter:

$$Hit_1 < Hit_2 \Leftrightarrow Hit_1.time < Hit_2.time$$

Definition at line 40 of file SiPMHit.h.

The documentation for this class was generated from the following files:

- /home/edo/UbuntuData/Projects/SimSiPM/SimSiPM/SiPMHit.h
- /home/edo/UbuntuData/Projects/SimSiPM/src/lib/SiPMHit.cpp

3.6 sipm::SiPMProperties Class Reference

Class storing all the parameters that describe a SiPM.

```
#include <SimSiPM/SimSiPM/SiPMProperties.h>
```

Public Types

- enum class PdeType { kNoPde , kSimplePde , kSpectrumPde }

    Used to set different methods to evaluate PDE for each photon.
- enum class HitDistribution { kUniform }

Public Member Functions

- void readSettings (std::string &)

  Used to read settings from a macro file.
- void dumpSettings () const

  Prints current settings of the sensor.
- const uint32_t nCells () const

  Returns total number of cells in the sensor.
- const uint32_t nSideCells () const

  Returns number of cells in the side of the sensor.
- const uint32_t nSignalPoints () const

  Returns total number of points in the signal.
- const HitDistribution hitDistribution () const

  Returns HitDistribution type of the sensor.
- const double signalLength () const

  Returns total signal length in ns.
- const double sampling () const

  Returns sampling time considered by the sensor in ns.
- const double risingTime () const

  Returns rising time constant.
- const double fallingTimeFast () const

  Returns falling time constant.
- const double fallingTimeSlow () const

  Returns falling time constant of slow component.
- const double slowComponentFraction () const

  Returns weight of slow component of the signal.
- const double recoveryTime () const

  Returns recovery time of SiPM cells.
- const double dcr () const

  Returns DCR value.
- const double xt () const

  Returns XT value.
- const double ap () const

  Returns AP value.
- const double tauApFast () const

  Returns fast time constant for AP.
- const double tauApSlow () const

  Returns slow time constant for AP.
- const double apSlowFraction () const

  Returns fraction of AP generated as slow.
- const double ccgv () const

  Returns value of cell-to-cell gain variation.
- const double snrdB () const

  Returns SNR in dB.
- const double snrLinear () const

  Returns RMS of the noise.
- const double pde () const

  Returns value of PDE if PdeType::kSimplePde is set.
- const std::map< double, double > pdeSpectrum () const

  Returns wavelength-PDE values if PdeType::kSpectrumPde is set.
- const PdeType pdeType ()

Returns type of PDE calculation used.

- const bool hasDcr () const

  Returns true if DCR is considered.
- const bool hasXt () const

  Returns true if XT is considered.
- const bool hasAp () const

  Returns true if AP is considered.
- const bool hasSlowComponent () const

  Returns true if slow component of the signal is considered.
- void setProperty (const std::string &, const double)

  Sets a property using its name.
- void setSize (const double x)

  Set size of SiPM sensitive area (side in mm)
- void setPitch (const double x)

  Set pitch of SiPM cells (side in um)
- void setSampling (const double x)

  Set sampling time of the signal in ns.
- void setSignalLength (const double x)

  Set length of the signa in ns.
- void setRiseTime (const double x)

  Set rising time constant of signal.
- void setFallTimeFast (const double x)

  Set falling time constant of signal.
- void setFallTimeSlow (const double x)

  Set falling time constant for the slow component of signal.
- void setSlowComponentFraction (const double x)

  Set weigth of slow component in the signal.
- void setRecoveryTime (const double x)

  Set recovery time of the SiPM cell.
- void setSnr (const double x)

  Set SNR value in dB.
- void setTauApFastComponent (const double x)

  Set time constant for the delay of fast afterpulses.
- void setTauApSlowComponent (const double x)

  Set time constant for the delay of slow afterpulses.
- void setTauApSlowFraction (const double x)

  Set probability to have slow afterpulses over fast ones.
- void setCcgv (const double x)

  Set cell-to-cell gain variation.
- void setPde (const double x)

  Set value for PDE (and sets PdeType::kSimplePde)
- void setDcr (const double aDcr)

  Set dark counts rate.
- void setXt (const double aXt)

  Set optical crosstalk probability.
- void setAp (const double aAp)

  Set afterpulse probability.
- void setDcrOff ()

  Turn off dark counts.
- void setXtOff ()

  Turn off optical crosstalk.

- void setApOff ()

  Turn off afterpulses.
- void setSlowComponentOff ()

  Turns off slow component of the signal.
- void setDcrOn ()

  Turn on dark counts.
- void setXtOn ()

  Turn on optical crosstalk.
- void setApOn ()

  Turn on afterpulses.
- void setSlowComponentOn ()

  Turns on slow component of the signal.
- void setPdeOff ()

  Turn off PDE: set PdeType::kNoPde.
- void setPdeSpectrum (const std::map< double, double > &)

  Set a spectral response of the SiPM and sets PdeType::kSpectrumPde.
- void setPdeSpectrum (const std::vector< double > &, const std::vector< double > &)

  Set a spectral response of the SiPM and sets PdeType::kSpectrumPde.

3.6.1 Detailed Description

Class storing all the parameters that describe a SiPM.

SiPMDigitalSignal.h

This class stores all the parameters and values used to describe a SiPM sensor or signal. It also allows to switch on or off some noise effects and can set different levels of detail in the evaluation of PDE.

Author

Edoardo Proserpio

Date

2020

Definition at line 25 of file SiPMProperties.h.

3.6.2 Member Enumeration Documentation

3.6.2.1 HitDistribution

```
enum sipm::SiPMProperties::HitDistribution  [strong]
```

Used to describe how photoelectrons are distributed on the SiPM surface

Enumerator

kUniform	Photons uniformly distributed on the sensor surface.

Definition at line 38 of file SiPMProperties.h.

3.6.2.2 PdeType

```
enum sipm::SiPMProperties::PdeType  [strong]
```

Used to set different methods to evaluate PDE for each photon.

Enumerator

kNoPde	No PDE applied, all photons will turn in photoelectrons.
kSimplePde	Same PDE value used for all photons.
kSpectrumPde	PDE calculated considering the wavelength of each photon.

Definition at line 30 of file SiPMProperties.h.

3.6.3 Member Function Documentation

3.6.3.1 fallingTimeFast()

```
const double sipm::SiPMProperties::fallingTimeFast ( ) const  [inline]
```

Returns falling time constant.

See also

SiPMSensor::signalShape

Definition at line 70 of file SiPMProperties.h.

3.6.3.2 fallingTimeSlow()

```
const double sipm::SiPMProperties::fallingTimeSlow ( ) const  [inline]
```

Returns falling time constant of slow component.

See also

SiPMSensor::signalShape

Definition at line 74 of file SiPMProperties.h.

3.6.3.3 hasSlowComponent()

```
const bool sipm::SiPMProperties::hasSlowComponent ( ) const  [inline]
```

Returns true if slow component of the signal is considered.

See also

SiPMSensor::signalShape

Definition at line 130 of file SiPMProperties.h.

3.6.3.4 readSettings()

```
void sipm::SiPMProperties::readSettings (
             std::string &  )
```

Used to read settings from a macro file.

Todo Still to implement

3.6.3.5 risingTime()

```
const double sipm::SiPMProperties::risingTime ( ) const  [inline]
```

Returns rising time constant.

See also

SiPMSensor::signalShape

Definition at line 67 of file SiPMProperties.h.

3.6.3.6 setAp()

```
void sipm::SiPMProperties::setAp (
             const double aAp )  [inline]
```

Set afterpulse probability.

Parameters

aAp	afterpulse probability [0-1]

Definition at line 207 of file SiPMProperties.h.

3.6.3.7 setCcgv()

```
void sipm::SiPMProperties::setCcgv (
             const double x )  [inline]
```

Set cell-to-cell gain variation.

See also

m_Ccgv

Definition at line 185 of file SiPMProperties.h.

3.6.3.8 setDcr()

```
void sipm::SiPMProperties::setDcr (
             const double aDcr )  [inline]
```

Set dark counts rate.

Parameters

aDcr	Dark counts rate in Hz

Definition at line 195 of file SiPMProperties.h.

3.6.3.9 setFallTimeFast()

```
void sipm::SiPMProperties::setFallTimeFast (
             const double x )  [inline]
```

Set falling time constant of signal.

See also

SiPMSensor::signalShape

Definition at line 151 of file SiPMProperties.h.

3.6.3.10 setFallTimeSlow()

```
void sipm::SiPMProperties::setFallTimeSlow (
             const double x )  [inline]
```

Set falling time constant for the slow component of signal.

See also

SiPMSensor::signalShape

Definition at line 155 of file SiPMProperties.h.

3.6.3.11 setRiseTime()

```
void sipm::SiPMProperties::setRiseTime (
             const double x )  [inline]
```

Set rising time constant of signal.

See also

SiPMSensor::signalShape

Definition at line 148 of file SiPMProperties.h.

3.6.3.12 setXt()

```
void sipm::SiPMProperties::setXt (
             const double aXt )  [inline]
```

Set optical crosstalk probability.

Parameters

a↩ Xt	optical crosstalk probability [0-1]

Definition at line 201 of file SiPMProperties.h.

3.6.3.13 slowComponentFraction()

```
const double sipm::SiPMProperties::slowComponentFraction ( ) const  [inline]
```

Returns weight of slow component of the signal.

See also

SiPMSensor::signalShape.

Definition at line 78 of file SiPMProperties.h.

The documentation for this class was generated from the following files:

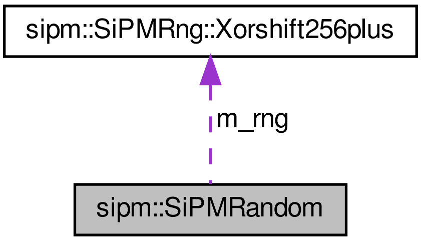
- /home/edo/UbuntuData/Projects/SimSiPM/SimSiPM/SiPMProperties.h
- /home/edo/UbuntuData/Projects/SimSiPM/src/lib/SiPMProperties.cpp

3.7 sipm::SiPMRandom Class Reference

Class for random number generation.

```
#include <SiPMRandom.h>
```

Collaboration diagram for sipm::SiPMRandom:



Public Member Functions

- void seed (uint64_t aSeed)

    Sets a seed for the rng.
- void seed ()

    Sets a seed for the rng obtained from rand().
- void jump ()

    Jump function of the rng, usefull in case of parallel execution to avoid correlation between number generated by different workers.
- double Rand ()
- uint32_t randInteger (const uint32_t)

    Returns a random integer in range [0,max].
- double randGaussian (const double, const double)

    Returns a value from a gaussian distribution given its mean value and sigma.
- double randExponential (const double)

    Returns a value from a exponential distribution given its mean value.
- uint32_t randPoisson (const double mu)

Returns a value from a poisson distribution given its mean value.

- std::vector< double > Rand (const uint32_t)

  Vector of random uniforms in [0-1].
- std::vector< double > randGaussian (const double, const double, const uint32_t)

  Vector of random gaussian given mean an sigma.
- std::vector< uint32_t > randInteger (const uint32_t max, const uint32_t n)

  Vector of random integers in range [0-max].

3.7.1 Detailed Description

Class for random number generation.

Class used for random number generation. The simulation needs very fast pseudo-random number generation, Xorhift256+ algorithm is used as it is one of the fastest considering modern x86-64 architectures.

Author

Edoardo Proserpio

Date

2020

Definition at line 69 of file SiPMRandom.h.

3.7.2 Member Function Documentation

3.7.2.1 Rand() [1/2]

```
double sipm::SiPMRandom::Rand ( )  [inline]
```

Returns a uniform random in range [0,1]

Definition at line 110 of file SiPMRandom.h.

3.7.2.2 Rand() [2/2]

```
std::vector< double > sipm::SiPMRandom::Rand (
             const uint32_t n )
```

Vector of random uniforms in [0-1].

Parameters

n	Number of values to generate

Definition at line 108 of file SiPMRandom.cpp.

3.7.2.3 randExponential()

```
double sipm::SiPMRandom::randExponential (
            const double mu )
```

Returns a value from a exponential distribution given its mean value.

Parameters

mu	Mean value of the exponential distribution

Definition at line 73 of file SiPMRandom.cpp.

3.7.2.4 randGaussian() [1/2]

```
double sipm::SiPMRandom::randGaussian (
            const double mu,
            const double sigma )
```

Returns a value from a gaussian distribution given its mean value and sigma.

This function is based on Ziggurat algorithm for random gaussian generation.

Parameters

mu	Mean value of the gaussian distribution
sigma	Standard deviation of the gaussian distribution

Definition at line 84 of file SiPMRandom.cpp.

3.7.2.5 randGaussian() [2/2]

```
std::vector< double > sipm::SiPMRandom::randGaussian (
            const double mu,
            const double sigma,
            const uint32_t n )
```

Vector of random gaussian given mean an sigma.

Parameters

mu	Mean value of the gaussuan
sigma	Standard deviation value of the gaussuan
n	Number of values to generate

Definition at line 122 of file SiPMRandom.cpp.

3.7.2.6 randInteger() [1/2]

```
std::vector< uint32_t > sipm::SiPMRandom::randInteger (
            const uint32_t max,
            const uint32_t n )
```

Vector of random integers in range [0-max].

Parameters

max	Max value to generate
n	Number of values to generate

Definition at line 149 of file SiPMRandom.cpp.

3.7.2.7 randInteger() [2/2]

```
uint32_t sipm::SiPMRandom::randInteger (
            const uint32_t max )  [inline]
```

Returns a random integer in range [0,max].

Parameters

max	Maximum value of integer to generate

Definition at line 117 of file SiPMRandom.h.

3.7.2.8 randPoisson()

```
uint32_t sipm::SiPMRandom::randPoisson (
            const double mu )
```

Returns a value from a poisson distribution given its mean value.

Parameters

mu	Mean value of the poisson distribution

Definition at line 57 of file SiPMRandom.cpp.

3.7.2.9 seed()

```
void sipm::SiPMRandom::seed (
             uint64_t aSeed )  [inline]
```

Sets a seed for the rng.

Parameters

aSeed	Seed used to initialize the rng algorithm

Definition at line 77 of file SiPMRandom.h.

The documentation for this class was generated from the following files:

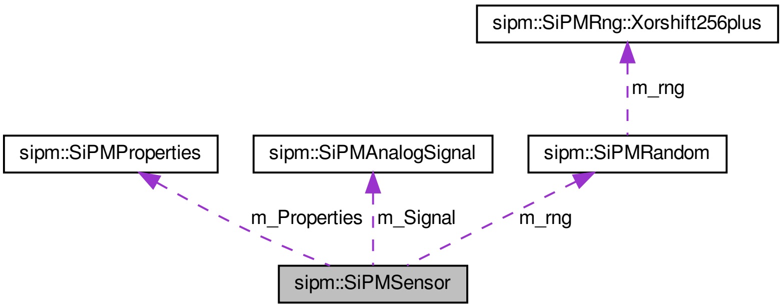
- /home/edo/UbuntuData/Projects/SimSiPM/src/components/SiPMRandom.h
- /home/edo/UbuntuData/Projects/SimSiPM/src/components/SiPMRandom.cpp

3.8 sipm::SiPMSensor Class Reference

Main class used to simulate a SiPM.

```
#include <SiPMSensor.h>
```

Collaboration diagram for sipm::SiPMSensor:



Public Types

- enum class PrecisionLevel

Public Member Functions

- SiPMSensor (const SiPMProperties &) noexcept

    SiPMSensor constructor from a SiPMProperties instance.
- SiPMSensor ()=default

    Default SiPMSensor contructor.
- const SiPMProperties & properties () const

    Returns a const reference to the SiPMProperties object.
- SiPMProperties & properties ()

    Returns a reference to the SiPMProperties object.
- const SiPMAnalogSignal & signal () const

    Returns a reference to SiPMAnalogSignal.
- const SiPMRandom & rng () const

    Returns a const reference to the SiPMRandom.
- SiPMRandom & rng ()

    Returns a reference to the SiPMRandom.
- SiPMDebugInfo debug ()

    Returns a SiPMDebugInfo.
- void setProperty (const std::string &, const double)

    Sets a property from its name.
- void setProperties (const SiPMProperties &)

    Sets a different SiPMProperties for the SiPMSensor.
- void addPhoton (const double)

    Adds a single photon to the list of photons to be simulated.
- void addPhoton (const double, const double)

    Adds a single photon to the list of photons to be simulated.
- void addPhotons (const std::vector< double > &)

    Adds all photons to the list of photons to be simulated at once.
- void addPhotons (const std::vector< double > &, const std::vector< double > &)

    Adds all photons to the list of photons to be simulated at once.
- void runEvent ()

    Runs a complete SiPM event.
- void resetState ()

    Resets internal state of the SiPMSensor.
- void setPrecisionLevel (const PrecisionLevel)

    Used to specify different PrecisionLevel.

Private Member Functions

- const std::vector< double > signalShape () const

    Returns the shape of the signal generated.
- const double evaluatePde (const double) const

    Returns the PDE value corresponding to the given wavelength.
- const bool isDetected (const double aPde) const

    Return wether the photon is detected given a PDE value.
- const bool isInSensor (const int32_t, const int32_t) const

Return wether the generated SiPMHit coordinates are allowed on the sensor's surface.

- const std::pair< int32_t, int32_t > hitCell () const

  Generates coordinates for a new hit.
- const std::vector< uint32_t > getCellIds () const

  Returns the id's of all hitted cells.
- void sortHits ()

  Inplace sorting of hitted cells.
- void addDcrEvents ()

  Generated DCR events.
- void addPhotoelectrons ()

  Generates photoelectrons starting from the photons.
- void addXtEvents ()

  Adds XT events.
- void addApEvents ()

  Add AP events.
- void calculateSignalAmplitudes ()

  Calculates signal amplitudes for all hits.
- void generateSignal ()

  Generates the SiPM signal.

3.8.1 Detailed Description

Main class used to simulate a SiPM.

This class provides all the methods to simulate a SiPM sensor.

Author

Edoardo Proserpio

Date

2020

Definition at line 26 of file SiPMSensor.h.

3.8.2 Member Enumeration Documentation

3.8.2.1 PrecisionLevel

```
enum sipm::SiPMSensor::PrecisionLevel  [strong]
```

Still unused

Definition at line 32 of file SiPMSensor.h.

3.8.3 Constructor & Destructor Documentation

3.8.3.1 SiPMSensor() [1/2]

```
sipm::SiPMSensor::SiPMSensor (
             const SiPMProperties & aProperty )  [noexcept]
```

SiPMSensor constructor from a SiPMProperties instance.

Instantiates a SiPMSensor with parameter specified in the SiPMProperties.

Definition at line 11 of file SiPMSensor.cpp.

3.8.3.2 SiPMSensor() [2/2]

```
sipm::SiPMSensor::SiPMSensor ( )  [default]
```

Default SiPMSensor contructor.

Instantiates a SiPMSensor with default settings.

3.8.4 Member Function Documentation

3.8.4.1 addApEvents()

```
void sipm::SiPMSensor::addApEvents ( )  [private]
```

Add AP events.

Adds afterpulse events. Each hit can produce a poissonian number of afterpulses. Each afterpulse is delayed from the generating signal following a slow/fast exponential distribution.

Definition at line 248 of file SiPMSensor.cpp.

3.8.4.2 addDcrEvents()

```
void sipm::SiPMSensor::addDcrEvents ( )  [private]
```

Generated DCR events.

Dark counts events are generated as poisson processes and directly added to the list of hitted cells.

Definition at line 143 of file SiPMSensor.cpp.

3.8.4.3 addPhotoelectrons()

```
void sipm::SiPMSensor::addPhotoelectrons ( )  [private]
```

Generates photoelectrons starting from the photons.

Starting from the all the photons added to the sensor a list of SiPMHit is created considering the PDE type and values set by the user and those hits are distributed on the SiPM surface considered the SiPMProperties::HitDistribution specified

Definition at line 164 of file SiPMSensor.cpp.

3.8.4.4 addXtEvents()

```
void sipm::SiPMSensor::addXtEvents ( )  [private]
```

Adds XT events.

Adds optical crosstalk events to the already existing photoelectrons. Each hitted cell may trigger a poissonian number of adjacent cells with mean value given by the XT probability. XT events are added to the listo of hits with the same time of the generating hit and theyr position is choosen randomly between the 9 neighbouring cells.

Definition at line 215 of file SiPMSensor.cpp.

3.8.4.5 calculateSignalAmplitudes()

```
void sipm::SiPMSensor::calculateSignalAmplitudes ( )  [private]
```

Calculates signal amplitudes for all hits.

Each hit has a starting amplitude of 1 but if the same cell has been previously hitted the resulting amplitude will be calculated considering the cell as an RC circuit, hence: $a = 1 - e^{-frac\Delta_t\tau}$

Definition at line 291 of file SiPMSensor.cpp.

3.8.4.6 debug()

```
SiPMDebugInfo sipm::SiPMSensor::debug ( )  [inline]
```

Returns a SiPMDebugInfo.

See also

SiPMDebugInfo

Definition at line 68 of file SiPMSensor.h.

3.8.4.7 evaluatePde()

```
const double sipm::SiPMSensor::evaluatePde (
             const double aPhotonWavelength ) const  [private]
```

Returns the PDE value corresponding to the given wavelength.

Uses the user defined spectral response to evaluate the PDE of photons gien theyr wavelength. PDE values are calculated by linearly interpolating the values stored in SiPMProperties::m_PdeSpectrum.

Definition at line 98 of file SiPMSensor.cpp.

3.8.4.8 generateSignal()

```
void sipm::SiPMSensor::generateSignal ( )  [private]
```

Generates the SiPM signal.

The SiPM signal is generated considering the arriving time of each hit and its amplitude. Signals are generated accorfingly to the signal produced by signalShape.

Definition at line 316 of file SiPMSensor.cpp.

3.8.4.9 hitCell()

```
const std::pair< int32_t, int32_t > sipm::SiPMSensor::hitCell ( ) const  [private]
```

Generates coordinates for a new hit.

This metod associates a photoelectron with a sipm cell. The coordinates of the hitted cell are generated randomly and accordingly to SiPMProperties::HitDistribution.

Definition at line 115 of file SiPMSensor.cpp.

3.8.4.10 properties() [1/2]

```
SiPMProperties& sipm::SiPMSensor::properties ( )  [inline]
```

Returns a reference to the SiPMProperties object.

used to setup ths SiPMSensor. Used to access and modify the SiPMSensor properties and settings

Definition at line 52 of file SiPMSensor.h.

3.8.4.11 properties() [2/2]

```
const SiPMProperties& sipm::SiPMSensor::properties ( ) const  [inline]
```

Returns a const reference to the SiPMProperties object.

used to setup ths SiPMSensor. Used to access the SiPMSensor properties and settings

Definition at line 47 of file SiPMSensor.h.

3.8.4.12 resetState()

```
void sipm::SiPMSensor::resetState ( )
```

Resets internal state of the SiPMSensor.

Resets the state of the SiPMSensor so it can be used again for a new event.

Definition at line 57 of file SiPMSensor.cpp.

3.8.4.13 rng()

```
SiPMRandom& sipm::SiPMSensor::rng ( )  [inline]
```

Returns a reference to the SiPMRandom.

Used to access and re-seed the underlying SiPMRandom object used for pseudo-random numbers generation.

Definition at line 64 of file SiPMSensor.h.

3.8.4.14 setPrecisionLevel()

```
void sipm::SiPMSensor::setPrecisionLevel (
             const PrecisionLevel x )
```

Used to specify different PrecisionLevel.

Still to implement

Definition at line 25 of file SiPMSensor.cpp.

3.8.4.15 setProperties()

```
void sipm::SiPMSensor::setProperties (
            const SiPMProperties & x )
```

Sets a different SiPMProperties for the SiPMSensor.

Changes the underlying SiPMProperties object with a new one.

Definition at line 23 of file SiPMSensor.cpp.

3.8.4.16 setProperty()

```
void sipm::SiPMSensor::setProperty (
            const std::string & prop,
            const double val )
```

Sets a property from its name.

Sets a SiPM property using its name. For a list of available SiPM properties names

See also

SiPMProperties

Definition at line 17 of file SiPMSensor.cpp.

3.8.4.17 signal()

```
const SiPMAnalogSignal& sipm::SiPMSensor::signal ( ) const  [inline]
```

Returns a reference to SiPMAnalogSignal.

Used to get the generated signal from the sensor. This method should be run after runEvent otherwise it will return only electronic noise.

Definition at line 57 of file SiPMSensor.h.

3.8.4.18 signalShape()

```
const std::vector< double > sipm::SiPMSensor::signalShape ( ) const  [private]
```

Returns the shape of the signal generated.

Return the ideal signal shape intended as the signal generated by a single photoelectron at time = 0. This signal will be used as a template to generate all other signals. Signal shape is based either on a two-exponential model or a three-exponential model in case slow component is considered. The two-exponential model is:

$$s(t) = e^{-\frac{t}{\tau_f}} - e^{-\frac{t}{\tau_r}}$$

The three exponential model adds another falling exponential term with a given weight.

Definition at line 69 of file SiPMSensor.cpp.

The documentation for this class was generated from the following files:

- /home/edo/UbuntuData/Projects/SimSiPM/SimSiPM/SiPMSensor.h
- /home/edo/UbuntuData/Projects/SimSiPM/src/lib/SiPMSensor.cpp

3.9 sipm::SiPMRng::Xorshift256plus Class Reference

Implementation of Xorshift256+ algorithm.

```
#include <SiPMRandom.h>
```

Public Member Functions

- Xorshift256plus () noexcept

  Default contructor for Xorshift256plus.
- Xorshift256plus (uint64_t aseed) noexcept

  Contructor for Xorshift256plus given a seed value.
- uint64_t operator() () noexcept

  Returns a pseud-random 64-bits intger.
- void jump ()

  Jump function for the alghoritm.
- void seed ()

  Sets a random seed generated with rand()
- void seed (uint64_t)

  Sets a new seed.

3.9.1 Detailed Description

Implementation of Xorshift256+ algorithm.

Author

Edoardo Proserpio

Date

2020

Definition at line 31 of file SiPMRandom.h.

3.9.2 Member Function Documentation

3.9.2.1 jump()

```
void sipm::SiPMRng::Xorshift256plus::jump ( )
```

Jump function for the alghoritm.

Usefull in case the same generator is used in multiple instancies. The jump function will make sure that pseudrandom values generated from the different instancies are uncorrelated.

Definition at line 27 of file SiPMRandom.cpp.

The documentation for this class was generated from the following files:

- /home/edo/UbuntuData/Projects/SimSiPM/src/components/SiPMRandom.h
- /home/edo/UbuntuData/Projects/SimSiPM/src/components/SiPMRandom.cpp

Index