SimSipm

Generated by Doxygen 1.9.1

Chapter 1

Todo List

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>`

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 setBits (const double bts)
  Sets number of bits.
- void setGain (const double gn)
  Sets gain parameter.
- void setJitter (const double jit)
  Sets jitter parameter.
- void setRange (const double rng)
  Sets range parameter.

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 digitize()

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

Digitizes an analog signalt to a digital one.

Parameters

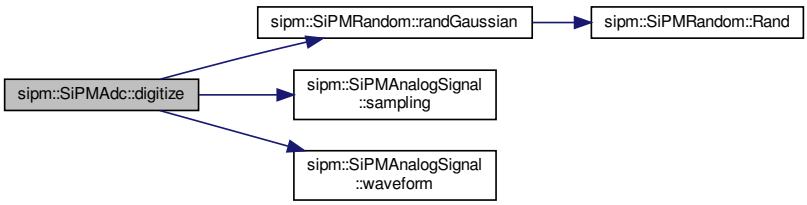
signal	Input signal to be digitized

Returns

Digitized SiPMAnalogSignal

Definition at line 99 of file SiPMAdc.cpp.

Here is the call graph for this function:



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

- SiPMAdc.h
- 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.
- const void clear ()
  Clears all elements of the vector containing the signal.
- const double integral (const double, const double, const double) const
  Returns integral of the signal.
- SiPMAnalogSignal lowpass (const double) const
  Applies a low-pass filter to the input 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.
- const double & operator[ ] (const uint32_t i) const noexcept
  Used to access signal elements as if it is a std::vector.
- double & operator[ ] (const uint32_t i) noexcept
  Used to access signal elements as if it is a std::vector.
- const double peak (const double, const double, const double) const
  Returns peak of the signal.
- const double sampling () const
  Returns the sampling time of the signal in ns.
- void setSampling (const double x)
  Sets the sampligng time of the signal.
- const uint32_t size () const
  Returns the size of the vector containing 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.
- const double tot (const double, const double, const double) const
  Returns time over threshold of the signal.
- const std::vector< double > & waveform () const
  Returns a std::vector containing the sampled waveform.

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

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

Returns

Signal with filter applied

Definition at line 100 of file SiPMAnalogSignal.cpp.

Here is the call graph for this function:

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 32 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 66 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 88 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 48 of file SiPMAnalogSignal.cpp.

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

- SiPMAnalogSignal.h
- 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 nAp
  Number of AP events generated.
- const uint32_t nDcr
  Number of DCR events generated.
- const uint32_t nPhotoelectrons
  Number of photoelectrons (hitted cells).
- const uint32_t nPhotons
  Number of photons given as input.
- const uint32_t nXt
  Number of XT 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 15 of file SiPMDebugInfo.h.

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

- 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.
- const void clear ()
  Clears all elements of the vector containing the signal.
- const int32_t integral (const double, const double, const int32_t) const
  Returns integral of the signal.
- 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 int32_t peak (const double, const double, const int32_t) const
  Returns peak of the signal.
- const double sampling () const
  Returns the sampling time of the signal in ns.
- const uint32_t size () const
  Returns the size of the vector containing 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.
- const double tot (const double, const double, const int32_t) const
  Returns time over threshold of the signal.
- const std::vector< int32_t > & waveform () const
  Used to access signal elements as if it is a std::vector.

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 35 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 76 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 98 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 56 of file SiPMDigitalSignal.cpp.

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

- SiPMDigitalSignal.h
- 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.
- double & amplitude ()
  Used to modify the amplitude if needed.
- const double amplitude () const
  Returns amplitude of the signal produced by the hit.
- const uint32_t col () const
  Returns column of hitted cell.
- const HitType & hitType () const
  Returns hit type to identify the hits.
- const uint32_t id () const
  Returns a unique id to identify a hitted cell.
- const bool operator< (const SiPMHit &aHit) const noexcept
  Operator used to sort hits.
- const uint32_t row () const
  Returns row of hitted cell.
- const double time () const
  Returns hit time.

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 19 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 24 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 38 of file SiPMHit.h.

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

- SiPMHit.h
- 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 HitDistribution { kUniform , kCircle }
- enum class PdeType { kNoPde , kSimplePde , kSpectrumPde }

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

Public Member Functions

- const double ap () const

  Returns AP value.
- 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 dcr () const

  Returns DCR value.
- void dumpSettings () const

  Prints current settings of the sensor.
- const double fallingTimeFast () const

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

  Returns falling time constant of slow component.
- const bool hasAp () const

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

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

  Returns true if slow component of the signal is considered.
- const bool hasXt () const

  Returns true if XT is considered.
- const HitDistribution hitDistribution () const

  Returns HitDistribution type 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 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.
- void readSettings (std::string &)

  Used to read settings from a macro file.
- const double recoveryTime () const

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

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

  Returns sampling time considered by the sensor in ns.
- void setAp (const double aAp)

  Set afterpulse probability.
- void setApOff ()

  Turn off afterpulses.
- void setApOn ()

Turn on afterpulses.

- void setCcgv (const double x)

  Set cell-to-cell gain variation.
- void setDcr (const double aDcr)

  Set dark counts rate.
- void setDcrOff ()

  Turn off dark counts.
- void setDcrOn ()

  Turn on dark counts.
- 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 setPde (const double x)

  Set value for PDE (and sets PdeType::kSimplePde)
- 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.
- void setPdeType (PdeType aPdeType)

  Turn off PDE: set PdeType::kNoPde.
- void setPitch (const double x)

  Set pitch of SiPM cells (side in um)
- void setProperty (const std::string &, const double)

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

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

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

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

  Set weigth of slow component in the signal.
- void setSlowComponentOff ()

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

  Turns on slow component of the signal.
- 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 setXt (const double aXt)

  Set optical crosstalk probability.

- void setXtOff ()
  Turn off optical crosstalk.
- void setXtOn ()
  Turn on optical crosstalk.
- const double signalLength () const
  Returns total signal length in ns.
- const double slowComponentFraction () const
  Returns weight of slow component of the signal.
- const double snrdB () const
  Returns SNR in dB.
- const double snrLinear () const
  Returns RMS of the noise.
- const double tauApFast () const
  Returns fast time constant for AP.
- const double tauApSlow () const
  Returns slow time constant for AP.
- const double xt () const
  Returns XT value.

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.
kCircle	95% of photons are uniformly distributed on a circle

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 71 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 75 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 131 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 68 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 195 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 180 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 187 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 152 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 156 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 149 of file SiPMProperties.h.

3.6.3.12 setXt()

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

Set optical crosstalk probability.

Parameters

aXt	optical crosstalk probability [0-1]

Definition at line 191 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 79 of file SiPMProperties.h.

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

- SiPMProperties.h
- SiPMProperties.cpp

3.7 sipm::SiPMRandom Class Reference

Class for random number generation.

`#include <SiPMRandom.h>`

Public Member Functions

- void jump ()

  This is the jump function for the generator. It is equivalent to 2^ 128 calls to next(); it can be used to generate 2^ 128 non-overlapping subsequences for parallel computations.
- double Rand ()
- std::vector< double > Rand (const uint32_t)

  Vector of random uniforms in [0-1].
- double randExponential (const double)

  Returns a value from a exponential distribution given its mean value.
- double randGaussian (const double, const double)

  Returns a value from a gaussian distribution given its mean value and sigma.
- 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].
- uint32_t randInteger (const uint32_t)

  Returns a random integer in range [0,max].
- uint32_t randPoisson (const double mu)

  Returns a value from a poisson distribution given its mean value.
- void seed ()

  Sets a seed for the rng obtained from rand().
- void seed (const uint64_t aSeed)

  Sets a seed for the rng.

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 78 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 119 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 100 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 67 of file SiPMRandom.cpp.

Here is the call graph for this function:

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.

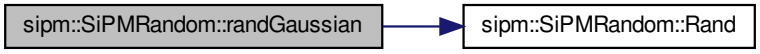
This function is based on Ziggurat algorithm for random gaussian generation.

Parameters

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

Definition at line 76 of file SiPMRandom.cpp.

Here is the call graph for this function:



3.7.2.5 randGaussian() [2/2]

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

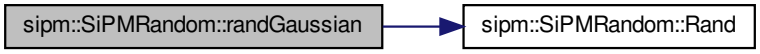
Vector of random gaussian given mean an sigma.

Parameters

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

Definition at line 118 of file SiPMRandom.cpp.

Here is the call graph for this function:



3.7.2.6 randInteger() [1/2]

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

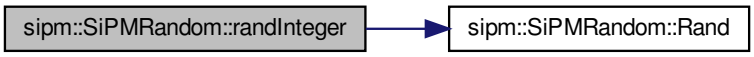
Vector of random integers in range [0-max].

Parameters

max	Max value to generate
n	Number of values to generate

Definition at line 149 of file SiPMRandom.cpp.

Here is the call graph for this function:



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 124 of file SiPMRandom.h.

3.7.2.8 randPoisson()

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

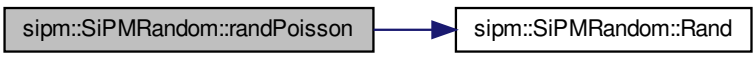
Returns a value from a poisson distribution given its mean value.

Parameters

mu	Mean value of the poisson distribution

Definition at line 49 of file SiPMRandom.cpp.

Here is the call graph for this function:



3.7.2.9 seed()

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

Sets a seed for the rng.

Parameters

aSeed	Seed used to initialize the rng algorithm

Definition at line 86 of file SiPMRandom.h.

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

- SiPMRandom.h
- SiPMRandom.cpp

3.8 sipm::SiPMSensor Class Reference

Main class used to simulate a SiPM.

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

Public Types

- enum class PrecisionLevel

Public Member Functions

- SiPMSensor ()=default

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

  SiPMSensor constructor from a SiPMProperties instance.
- 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.
- SiPMDebugInfo debug ()

  Returns a SiPMDebugInfo.
- SiPMProperties & properties ()

Returns a reference to the SiPMProperties object.

- const SiPMProperties & properties () const

  Returns a const reference to the SiPMProperties object.
- void resetState ()

  Resets internal state of the SiPMSensor.
- SiPMRandom & rng ()

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

  Returns a const reference to the SiPMRandom.
- void runEvent ()

  Runs a complete SiPM event.
- void setPrecisionLevel (const PrecisionLevel)

  Used to specify different PrecisionLevel.
- void setProperties (const SiPMProperties &)

  Sets a different SiPMProperties for the SiPMSensor.
- void setProperty (const std::string &, const double)

  Sets a property from its name.
- const SiPMAnalogSignal & signal () const

  Returns a reference to SiPMAnalogSignal.

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 27 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 12 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 debug()

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

Returns a SiPMDebugInfo.

See also

SiPMDebugInfo

Definition at line 68 of file SiPMSensor.h.

3.8.4.2 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.

3.8.4.3 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.

3.8.4.4 resetState()

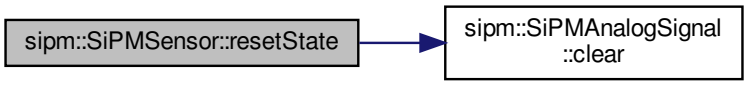
```
void sipm::SiPMSensor::resetState ( )
```

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 61 of file SiPMSensor.cpp.

Here is the call graph for this function:



3.8.4.5 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.

3.8.4.6 setPrecisionLevel()

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

Used to specify different PrecisionLevel.

Still to implement

Definition at line 30 of file SiPMSensor.cpp.

3.8.4.7 setProperties()

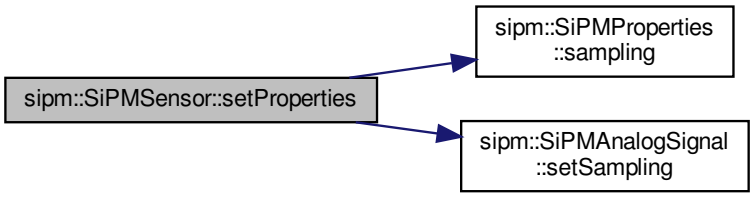
```
void sipm::SiPMSensor::setProperties (
            const SiPMProperties & x )
```

Sets a different SiPMProperties for the SiPMSensor.

Changes the underlying SiPMProperties object with a new one.

Definition at line 24 of file SiPMSensor.cpp.

Here is the call graph for this function:



3.8.4.8 setProperty()

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

Sets a property from its name.

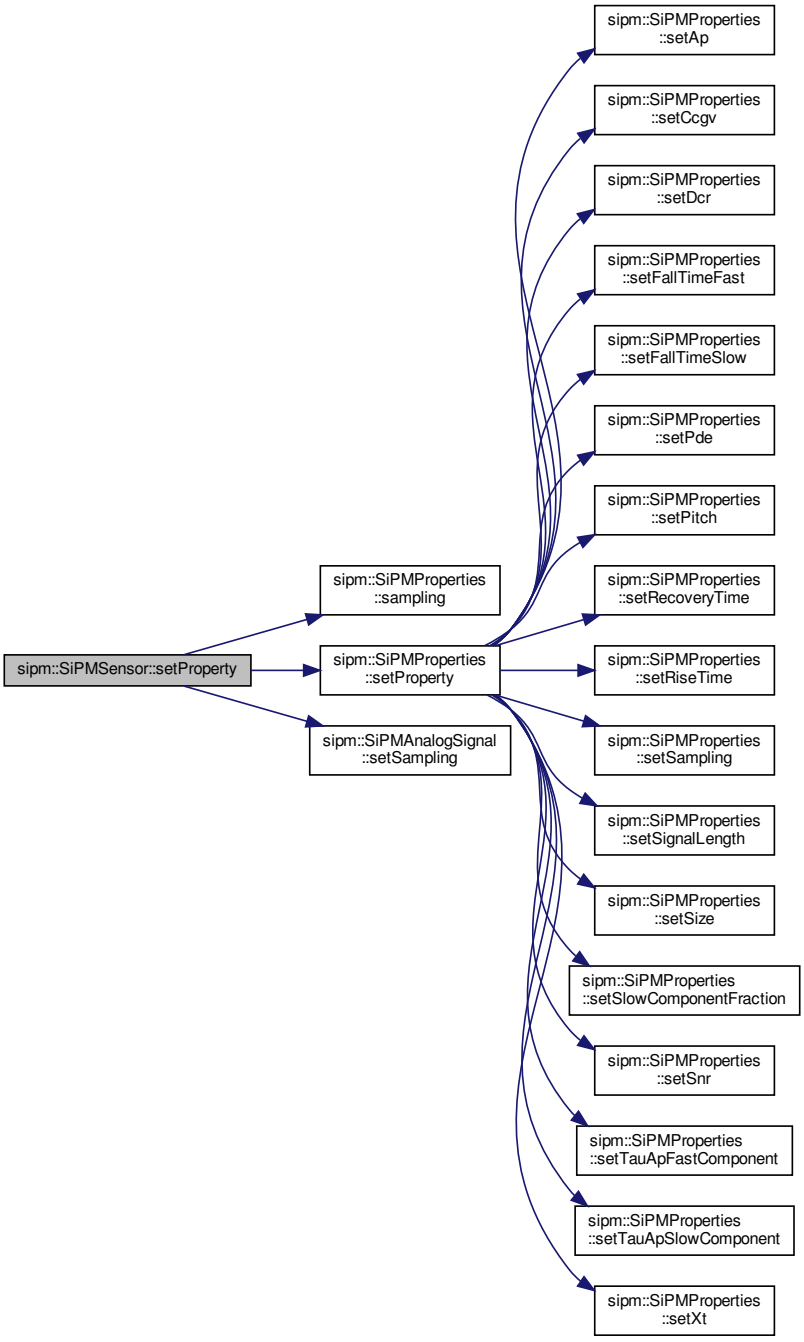
Sets a SiPM property using its name. For a list of available SiPM properties names

See also

SiPMProperties

Definition at line 18 of file SiPMSensor.cpp.

Here is the call graph for this function:



3.8.4.9 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.

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

- SiPMSensor.h
- SiPMSensor.cpp

3.9 sipm::SiPMRng::Xorshift256plus Class Reference

Implementation of Xorshift256+ algorithm.

`#include <SimSiPM/src/components/SiPMRandom.h>`

Public Member Functions

- Xorshift256plus () noexcept
  Default contructor for Xorshift256plus.
- Xorshift256plus (uint64_t aseed) noexcept
  Contructor for Xorshift256plus given a seed value.
- void jump ()
  Jump function for the alghoritm.
- uint64_t operator() () noexcept
  Returns a pseud-random 64-bits intger.
- 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.

SiPMRandom.h

Author

Edoardo Proserpio

Date

2020

Definition at line 36 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 pseudo-random values generated from the different instancies are uncorrelated.

Definition at line 23 of file SiPMRandom.cpp.

Here is the call graph for this function:

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

- SiPMRandom.h
- SiPMRandom.cpp

Index