Alpide Dataflow SystemC Model

Generated by Doxygen 1.8.11

Contents

Chapter 1

Introduction

This is the Alpide Dataflow Model documentation, a simulation model of the Alpide chip written in C++ and using the SystemC framework.

The simulation model aims to be a fairly accurate model of the Alpide chip's readout chain, but with the performance benefits that C++ and SystemC offer compared to traditional HDL simulations.

Building the code and getting started

Instructions from the README on how to build, use and run the code can be found here:

- Building and Running the code

Codying style

Below is a brief description of the coding style used for this code.

Class member names

- SystemC port and signal names: all lowercase, with s_ prefix, and _in or _out postfix to indicate input/output. They should also be at the top of the class definition, separated from normal C++ member variables.
- Class names: Camel case names, starting with capital letter
- Class methods: Camel case names, starting with lower case letter (except for constructor etc. obviously)
- Class member variables: camel case names, starting with an m to indicate member variable, e.g. mNum↩ Events.
- Class function parameters: all lowercase, words separated with underscore, to separate from class member variables. Example: num_events.

Documentation

Doxygen style comments

Chapter 2

Building and Running the code

A simple Dataflow SystemC Model of ITS and the Alpide chip

Getting started - building the project

Building the project requires GCC with C++11 support (included from version 4.8.x I think).

Required Libraries

- Requires qmake and boost libraries. On ubuntu, they can be installed with:
- qmake:

```
1 sudo apt-get install qt5-default
```

- boost (full installation):

```
1 sudo apt-get install libboost-all-dev
```

- SystemC:

Can be downloaded from http://accellera.org/downloads/standards/systemc

The code has been tested and developed with version 2.3.1 of SystemC. Refer to SystemC documentation for build/install instructions.

Environment

The project's makefile expects to find the path to the base directory of the SystemC installation in $SYSTEM↩ C_HOME. It may also be necessary to add the path to the SystemC libraries to $LD_LIBRARY_PATH. Adding something like this to $HOME/.profile should do (for version 2.3.1):

```
1 export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path/to/systemc-2.3.1/lib-linux64/
2 export SYSTEMC_HOME=/path/to/systemc-2.3.1
```

Compiling the code

```
1 cd alpide_dataflow_model
2 make
```

To run:

```
1 ./alpide_dataflow_model
```

The program requires a settings.txt file with simulation settings. If it does not exist, running the program generates a settings.txt file with default settings. This file can be edited and the simulation rerun to use those settings. The settings.txt file will not be overwritten.

Simulation results will be saved in sim_output/Run {timestamp}/

Simulation output is stored in alpide_dataflow_model/sim_output/{timestamp}/

To process simulation data:

```
1 cd alpide_dataflow_model/sim_output/{timestamp}/
2 root -b -q -l '../../process/process_event_data.C+("physics_events_data.csv")'
```

Chapter 3

Todo List

Member Alpide::dataTransmission (void)

Implement more advanced data transmission method.

Member Alpide::frameReadout (void)

Strobe extended not implemented yet

Should the busy_transition flag always be set like this when chip is busy, or should it only happen when the chip goes into or out of busy state?

Member Alpide::mainProcess (void)

Implement more advanced data transmission method.

Member Alpide::s_event_buffers_used_debug

Should these signals be private maybe?

Member Alpide::strobeInput (void)

What should I do in data overrun mode (when readout_abort is set)? Should I still accept events? I need the frame end word to be added, for the normal transmission of CHIP HEADER/TRAILER words. This is currently done by the frameReadout() method, which requires there to be events in the MEB.

Should rejected triggers count be increased in data overrun mode?

The FATAL overflow bit/signal has to be cleared by a RORST/GRST command in the Alpide chip, it will not be cleared by automatically.

Need to clear RRU FIFOs, and MEBs when entering this state

File alpide_data_parser.h

Move this class to a separate directory/module. Don't mix it with the Alpide simulation model.

Member AlpideDataWord::operator<< (std::ostream &stream, const AlpideDataWord &alpide_dw)

Overload this for all AlpideDataWord classes, so SystemC can print them to trace files properly?

Member AlpideEventBuilder::inputDataWord (AlpideDataWord dw)

Busy on here

Busy off here

Unknown Alpide data word received. Do something smart here?

Member EventGenerator::addHitsToTriggerEvent (TriggerEvent &e)

Is this check worth it performance wise, or is it better to just iterate through the whole list?

Member EventGenerator::generateNextPhysicsEvent (void)

Account larger/bigger clusters here (when implemented)

Remove?

Member EventGenerator::generateNextTriggerEvent (int64_t event_start, int64_t event_end, int chip_id)

Should I check distance between start time of two triggers? Or the distance in time between the end of the first trigger and the start of the next trigger?

Member EventGenerator::mDataPath

This is currently used.. remove or update code that uses it..

Member EventGenerator::physicsEventProcess (void)

Maybe do this only on strobe falling edge? Saves some CPU cycles that way?

Member EventGenerator::scaleDiscreteDistribution (std::vector< double > &dist_vector, double new_mean_value)

This changes the mean value slightly.. and the sum isn't that far off 1.0 before this anyway...

Member EventGenerator::setRandomSeed (int seed)

More than one seed? What if seed is set after random number generators have been started?

Member FrameEndFifoWord::operator<< (std::ostream &stream, const FrameEndFifoWord &dw)

Overload this for all FrameEndFifoWord classes, so SystemC can print them to trace files properly?

Member FrameStartFifoWord::operator<< (std::ostream &stream, const FrameStartFifoWord &dw)

Overload this for all FrameStartFifoWord classes, so SystemC can print them to trace files properly?

Member PixelMatrix::mColumnBuffs

Implement event ID somewhere. Maybe make an MEB class, and use it as the datatype for this queue?

Member print_event_rate (const std::list< int > &t_delta_queue)

Update/fix/remove this function.. currently not used..

Member RegionReadoutUnit::addTraces (sc_trace_file *wf, std::string name_prefix) const

Probably need to a stream << operator to allow values from fifo to be printed to trace file

Probably need to a stream << operator to allow values from fifo to be printed to trace file

Member sc_main (int argc, char **argv)

Pass vcd trace object to constructor of Stimuli class and Alpide classes?

Add a warning here if user tries to simulate over 1000 events with this option enabled, because it will consume 100s of megabytes

Member Stimuli::mNumEvents

Make it a 64-bit int?

Member Stimuli::stimuliEventProcess (void)

Check if there are actually events? Throw an error if we get notification but there are not events?

Member TopReadoutUnit::topRegionReadoutProcess (void)

Update state machine pictures with Alpide documentation + simplified FSM diagram

Should I maybe use the empty signal for the current region here??

File trigger_event.h

Use SystemC time data type instead of int64_t?

Member TriggerEvent::mEventFilteredFlag

With the new way of doing things, I don't need to have an Event object to keep track of hits, they are stored in the EventGenerator object. So I can get rid off this?

Member TriggerEvent::writeToFile (const std::string path="")

Note in use.. Revisit this function, since I have changed this class a lot...

Implement layers etc.

Chapter 4

Module Index

4.1 Modules

Here is a list of all modules:

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 6

File Index

6.1 File List

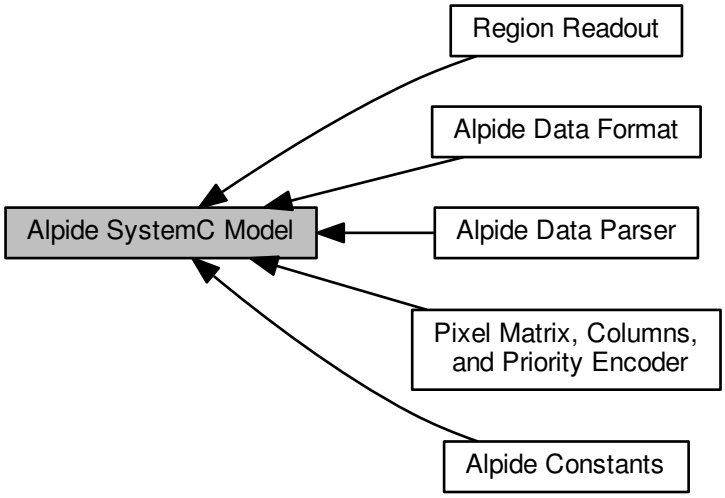
Here is a list of all documented files with brief descriptions:

Chapter 7

Module Documentation

7.1 Alpide SystemC Model

Collaboration diagram for Alpide SystemC Model:



Modules

- Alpide Constants
- Alpide Data Format
- Alpide Data Parser
- Pixel Matrix, Columns, and Priority Encoder
- Region Readout

Classes

- class Alpide
- class TopReadoutUnit

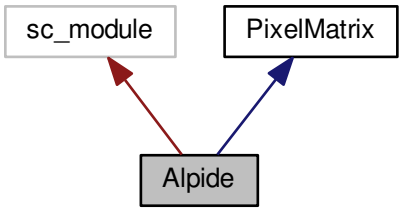
7.1.1 Detailed Description

7.1.2 Class Documentation

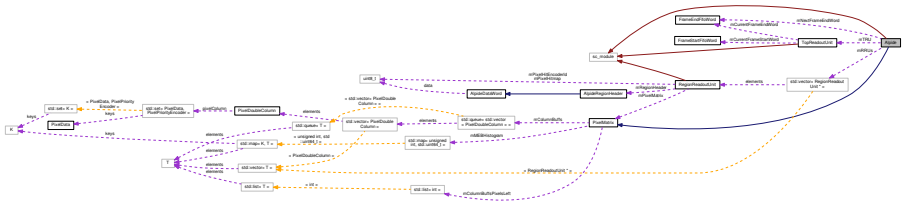
7.1.2.1 class Alpide

Alpide main class. Currently it only implements the MEBs, no RRU FIFOs, and no TRU FIFO. It will be used to run some initial estimations for probability of MEB overflow (busy).

Inheritance diagram for Alpide:



Collaboration diagram for Alpide:



Public Member Functions

- Alpide (sc_core::sc_module_name name, int chip_id, int region_fifo_size, int dmu_fifo_size, bool enable_↩ clustering, bool continuous_mode, bool matrix_readout_speed)

  Constructor for Alpide.
- int **getChipId** (void)
- void addTraces (sc_trace_file ∗wf, std::string name_prefix) const

  Add SystemC signals to log in VCD trace file.
- uint64_t **getTriggerEventsAcceptedCount** (void) const
- uint64_t **getTriggerEventsRejectedCount** (void) const

Public Attributes

- sc_in_clk s_system_clk_in

  40MHz LHC clock
- sc_in< bool > **s_strobe_n_in**
- sc_out< bool > s_chip_ready_out

  Indicates that the chip is ready to accept hits and setPixel() can be called.
- sc_out< sc_uint< 24 > > **s_serial_data_output**
- sc_signal< sc_uint< 8 > > s_event_buffers_used_debug

  Number of events stored in the chip at any given time.
- sc_signal< sc_uint< 8 > > **s_frame_start_fifo_size_debug**
- sc_signal< sc_uint< 8 > > **s_frame_end_fifo_size_debug**
- sc_signal< sc_uint< 32 > > s_total_number_of_hits

  Sum of all hits in all multi event buffers.
- sc_signal< sc_uint< 32 > > s_oldest_event_number_of_hits

  Number of hits in oldest multi event buffer.
- sc_signal< bool > **s_region_fifo_empty** [N_REGIONS]
- sc_signal< bool > **s_region_valid** [N_REGIONS]
- sc_signal< bool > **s_region_data_read** [N_REGIONS]
- sc_signal< bool > **s_region_event_start**
- sc_signal< bool > **s_region_event_pop**
- sc_signal< AlpideDataWord > **s_region_data** [N_REGIONS]
- sc_signal< bool > s_frame_readout_start

  Frame Readout Managment Unit (FROMU) signals.
- sc_signal< bool > **s_frame_readout_done** [N_REGIONS]
- sc_signal< bool > **s_frame_readout_done_all**
- sc_signal< bool > **s_frame_fifo_busy**
- sc_signal< bool > **s_multi_event_buffers_busy**
- sc_signal< bool > **s_fatal_state**
- sc_signal< bool > **s_readout_abort**
- sc_signal< bool > **s_flushed_incomplete**
- sc_signal< bool > **s_busy_violation**
- sc_signal< bool > **s_busy_status**
- sc_fifo< AlpideDataWord > **s_dmu_fifo**
- sc_signal< sc_uint< 8 > > **s_dmu_fifo_size**
- sc_signal< bool > **s_chip_ready_internal**

Private Types

- enum **FROMU_readout_state_t** { **WAIT_FOR_EVENTS** = 0, **REGION_READOUT_START** = 1, **WAIT_F↩ OR_REGION_READOUT** = 2, **REGION_READOUT_DONE** = 3 }

Private Member Functions

- void mainProcess (void)

  Data transmission SystemC method. Currently runs on 40MHz clock.
- void strobeInput (void)

  This function handles the strobe input to the Alpide class object. Controls creation of new Multi Event Buffers (MEBs). Together with the frameReadout function, this process essentially does the same as the FROMU (Frame Read Out Management Unit) in the Alpide chip. Note: it is assumed that STROBE is synchronous to the clock. It will not be "dangerous" if it is not, but it will deviate from the real chip implementation.
- void frameReadout (void)

Frame readout SystemC method @ 40MHz (system clock). Together with the strobeProcess, this function essentially does the same job as the FROMU (Frame Read Out Management Unit) in the Alpide chip.

- void dataTransmission (void)

  Read out DMU FIFOs and output data on "serial" line. Should be called one time per clock cycle.
- bool getFrameReadoutDone (void)

  Get logical AND/product of all regions' frame_readout_done signals.
- void updateBusyStatus (void)

  Update internal busy status signals.

Private Attributes

- tlm::tlm_fifo< FrameStartFifoWord > **s_frame_start_fifo**
- tlm::tlm_fifo< FrameEndFifoWord > **s_frame_end_fifo**
- FrameEndFifoWord **mNextFrameEndWord**
- sc_signal< sc_uint< 8 > > **s_fromu_readout_state**
- int **mChipId**
- bool **mEnableReadoutTraces**
- bool **mStrobeActive**
- uint16_t **mBunchCounter**
- uint64_t mTriggerEventsAccepted = 0

  Number of (trigger) events that are accepted into an MEB by the chip.
- uint64_t mTriggerEventsRejected = 0

  Triggered mode: If 3 MEBs are already full, the chip will not accept more events until one of those 3 MEBs have been read out. This variable is counted up for each event that is not accepted.
- uint64_t mTriggerEventsFlushed = 0

  Continuous mode only. The Alpide chip will try to guarantee that there is a free MEB slice in continuous mode. It does this by deleting the oldest MEB slice (even if it has not been read out) when the 3rd one is filled. This variable counts up in that case.
- std::vector< RegionReadoutUnit * > **mRRUs**
- TopReadoutUnit * **mTRU**

Additional Inherited Members

7.1.2.1.1 Constructor & Destructor Documentation

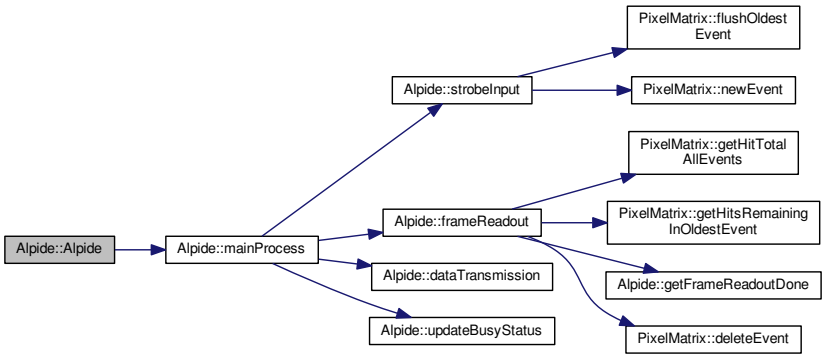
7.1.2.1.1.1 Alpide::Alpide (sc_core::sc_module_name *name*, int *chip_id*, int *region_fifo_size*, int *dmu_fifo_size*, bool *enable_clustering*, bool *continuous_mode*, bool *matrix_readout_speed*)

Constructor for Alpide.

Parameters

in	*name*	SystemC module name
in	*chip_id*	Desired chip id
in	*region_fifo_size*	Depth of Region Readout Unit (RRU) FIFOs
in	*dmu_fifo_size*	Depth of DMU (Data Management Unit) FIFO.
in	*enable_clustering*	Enable clustering and use of DATA LONG words
in	*continuous_mode*	Enable continuous mode (triggered mode if false)
in	*matrix_readout_speed*	True for fast readout (2 clock cycles), false is slow (4 cycles).

Here is the call graph for this function:



7.1.2.1.2 Member Function Documentation

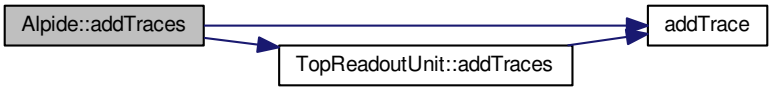
7.1.2.1.2.1 void Alpide::addTraces (sc_trace_file * *wf*, std::string *name_prefix*) const

Add SystemC signals to log in VCD trace file.

Parameters

in,out	*wf*	Pointer to VCD trace file object
in	*name_prefix*	Name prefix to be added to all the trace names

Here is the call graph for this function:



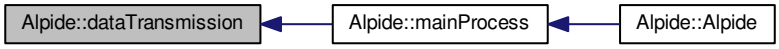
7.1.2.1.2.2 void Alpide::dataTransmission (void) [private]

Read out DMU FIFOs and output data on "serial" line. Should be called one time per clock cycle.

Todo Implement more advanced data transmission method.

Referenced by mainProcess().

Here is the caller graph for this function:



7.1.2.1.2.3 void Alpide::frameReadout (void) [private]

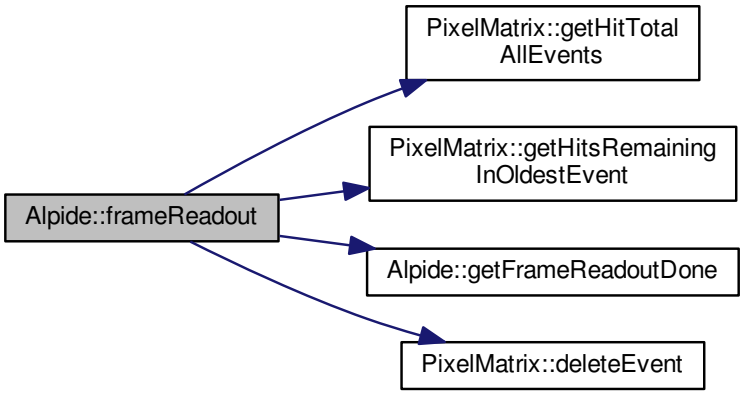
Frame readout SystemC method @ 40MHz (system clock). Together with the strobeProcess, this function essentially does the same job as the FROMU (Frame Read Out Management Unit) in the Alpide chip.

Todo Strobe extended not implemented yet

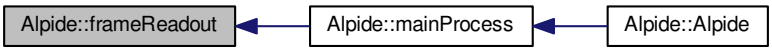
Todo Should the busy_transition flag always be set like this when chip is busy, or should it only happen when the chip goes into or out of busy state?

Referenced by mainProcess().

Here is the call graph for this function:



Here is the caller graph for this function:



7.1.2.1.2.4 bool Alpide::getFrameReadoutDone (void) [private]

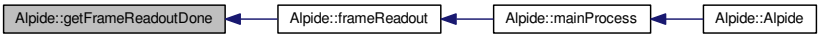
Get logical AND/product of all regions' frame_readout_done signals.

Returns

True when frame_readout_done is set in all regions

Referenced by frameReadout().

Here is the caller graph for this function:



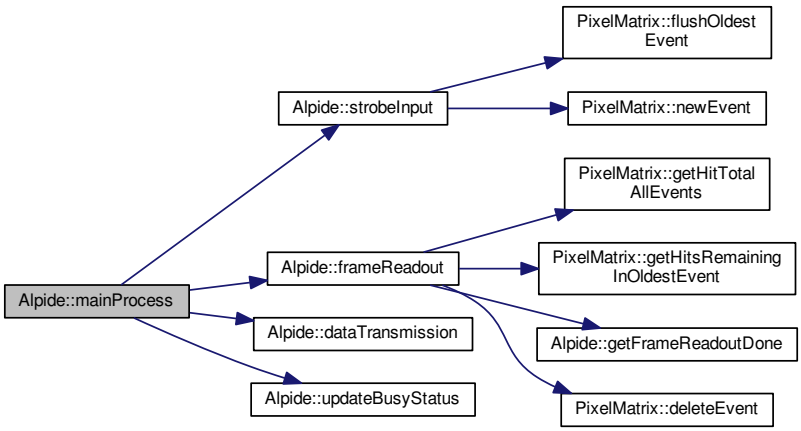
7.1.2.1.2.5 void Alpide::mainProcess (void) [private]

Data transmission SystemC method. Currently runs on 40MHz clock.

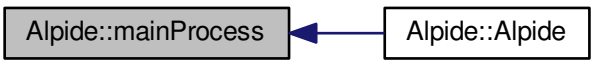
Todo Implement more advanced data transmission method.

Referenced by Alpide().

Here is the call graph for this function:



Here is the caller graph for this function:



7.1.2.1.2.6 void Alpide::strobeInput (void) [private]

This function handles the strobe input to the Alpide class object. Controls creation of new Multi Event Buffers (MEBs). Together with the frameReadout function, this process essentially does the same as the FROMU (Frame Read Out Management Unit) in the Alpide chip. Note: it is assumed that STROBE is synchronous to the clock. It will not be "dangerous" if it is not, but it will deviate from the real chip implementation.

Todo What should I do in data overrun mode (when readout_abort is set)? Should I still accept events? I need the frame end word to be added, for the normal transmission of CHIP HEADER/TRAILER words. This is currently done by the frameReadout() method, which requires there to be events in the MEB.

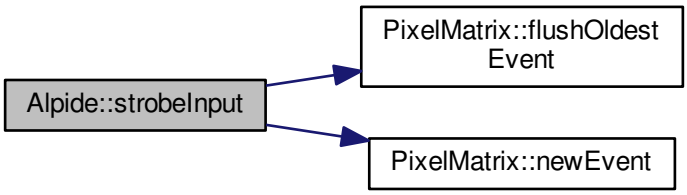
Todo Should rejected triggers count be increased in data overrun mode?

Todo The FATAL overflow bit/signal has to be cleared by a RORST/GRST command in the Alpide chip, it will not be cleared by automatically.

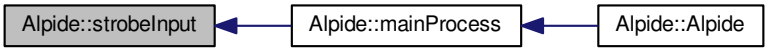
Todo Need to clear RRU FIFOs, and MEBs when entering this state

Referenced by mainProcess().

Here is the call graph for this function:



Here is the caller graph for this function:



7.1.2.1.3 Member Data Documentation

7.1.2.1.3.1 sc_signal<sc_uint<8> > Alpide::s_event_buffers_used_debug

Number of events stored in the chip at any given time.

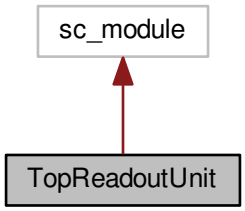
Todo Should these signals be private maybe?

Referenced by addTraces(), Alpide(), and frameReadout().

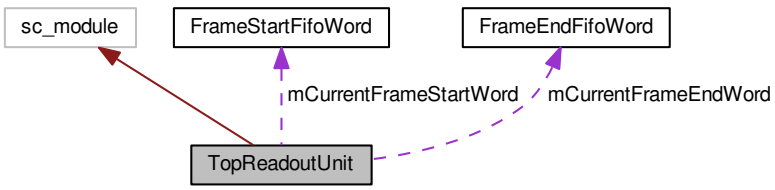
7.1.2.2 class TopReadoutUnit

The TopReadoutUnit (TRU) class is a simple representation of the TRU in the Alpide chip. It should be connected to the Region Readout Unit (RRU) in the Alpide object, and will be responsible for reading out from the RRUs with the topRegionReadoutProcess, which should run at the system clock (40MHz).

Inheritance diagram for TopReadoutUnit:



Collaboration diagram for TopReadoutUnit:



Public Member Functions

- TopReadoutUnit (sc_core::sc_module_name name, unsigned int chip_id)

  Constructor for TopReadoutUnit.
- void topRegionReadoutProcess (void)

  SystemC method that controls readout from regions, should run on the 40MHz clock. The regions are read out in ascending order, and each event is encapsulated with a CHIP_HEADER and CHIP_TRAILER word. See the state machine diagram for a better explanation.
- void addTraces (sc_trace_file ∗wf, std::string name_prefix) const

  Add SystemC signals to log in VCD trace file.

Public Attributes

- sc_in_clk s_clk_in

  Alpide chip clock (typically 40MHz)
- sc_in< bool > **s_readout_abort_in**
- sc_in< bool > **s_fatal_state_in**
- sc_in< bool > **s_region_fifo_empty_in** [N_REGIONS]
- sc_in< bool > **s_region_valid_in** [N_REGIONS]
- sc_in< AlpideDataWord > **s_region_data_in** [N_REGIONS]
- sc_out< bool > **s_region_event_pop_out**
- sc_out< bool > **s_region_event_start_out**
- sc_out< bool > **s_region_data_read_out** [N_REGIONS]
- sc_port< tlm::tlm_nonblocking_get_peek_if< FrameStartFifoWord > > **s_frame_start_fifo_output**
- sc_port< tlm::tlm_nonblocking_get_peek_if< FrameEndFifoWord > > **s_frame_end_fifo_output**
- sc_port< sc_fifo_out_if< AlpideDataWord > > s_dmu_fifo_input

  Output from TRU.

Private Types

- enum **TRU_state_t** {
  EMPTY = 0, **IDLE** = 1, **WAIT_REGION_DATA** = 2, **CHIP_HEADER** = 3,
  BUSY_VIOLATION = 4, **REGION_DATA** = 5, **WAIT** = 6, **CHIP_TRAILER** = 7 }

Private Member Functions

- bool getNextRegion (unsigned int &region_out)

  Find the first valid region, and return its region id.
- bool getAllRegionsEmpty (void)

  AND all region empty signals together.

Private Attributes

- sc_signal< sc_uint< 8 > > **s_tru_state**
- sc_signal< sc_uint< 8 > > **s_previous_region**
- sc_signal< bool > s_all_regions_empty_debug

  Signal copy of all_regions_empty variable, 1 cycle delayed.
- sc_signal< bool > s_no_regions_valid_debug

  Signal copy of no_regions_valid variable, 1 cycle delayed.
- unsigned int **mChipId**
- FrameStartFifoWord **mCurrentFrameStartWord**
- FrameEndFifoWord **mCurrentFrameEndWord**

7.1.2.2.1 Constructor & Destructor Documentation

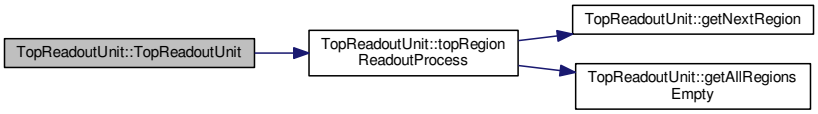
7.1.2.2.1.1 TopReadoutUnit::TopReadoutUnit (sc_core::sc_module_name *name*, unsigned int *chip_id*)

Constructor for TopReadoutUnit.

Parameters

in	*name*	SystemC module name
in	*chip↩ _id*	Chip ID number

Here is the call graph for this function:



7.1.2.2.2 Member Function Documentation

7.1.2.2.2.1 void TopReadoutUnit::addTraces (sc_trace_file * *wf*, std::string *name_prefix*) const

Add SystemC signals to log in VCD trace file.

Parameters

in,out	*wf*	Pointer to VCD trace file object
in	*name_prefix*	Name prefix to be added to all the trace names

Referenced by Alpide::addTraces().

Here is the call graph for this function:

Here is the caller graph for this function:

7.1.2.2.2 bool TopReadoutUnit::getAllRegionsEmpty (void) [private]

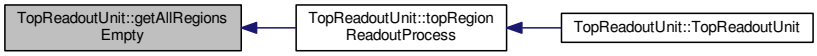
AND all region empty signals together.

Returns

true if all regions are empty

Referenced by topRegionReadoutProcess().

Here is the caller graph for this function:



7.1.2.2.3 bool TopReadoutUnit::getNextRegion (unsigned int & *region_out*) [private]

Find the first valid region, and return its region id.

Parameters

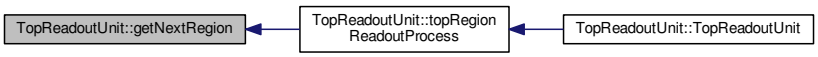
out	*region_out*	Reference to an integer that will hold the region id.

Returns

True if a valid region was found.

Referenced by topRegionReadoutProcess().

Here is the caller graph for this function:



7.1.2.2.4 void TopReadoutUnit::topRegionReadoutProcess (void)

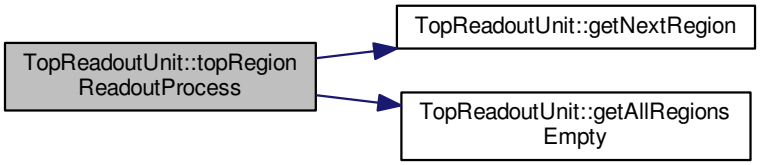
SystemC method that controls readout from regions, should run on the 40MHz clock. The regions are read out in ascending order, and each event is encapsulated with a CHIP_HEADER and CHIP_TRAILER word. See the state machine diagram for a better explanation.

Todo Update state machine pictures with Alpide documentation + simplified FSM diagram

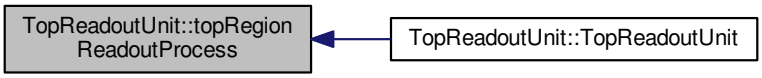
Todo Should I maybe use the empty signal for the current region here??

Referenced by TopReadoutUnit().

Here is the call graph for this function:

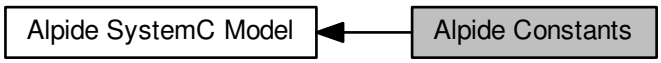


Here is the caller graph for this function:



7.2 Alpide Constants

Collaboration diagram for Alpide Constants:



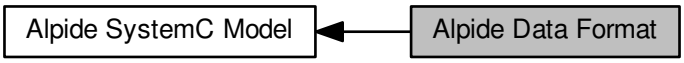
Macros

- #define **N_REGIONS** 32
- #define **N_PIXEL_ROWS** 512
- #define **N_PIXEL_COLS** 1024
- #define **N_PIXEL_COLS_PER_REGION** (N_PIXEL_COLS/N_REGIONS)
- #define **N_PIXEL_DOUBLE_COLS_PER_REGION** (N_PIXEL_COLS_PER_REGION/2)
- #define **N_PIXELS_PER_REGION** (N_PIXEL_COLS/N_REGIONS)
- #define **TRU_FRAME_FIFO_ALMOST_FULL1** 48
- #define **TRU_FRAME_FIFO_ALMOST_FULL2** 56
- #define **TRU_FRAME_FIFO_SIZE** 64
- #define **DATA_LONG_PIXMAP_SIZE** ((unsigned int) 7)
- #define **LHC_ORBIT_BUNCH_COUNT** 3564
- #define **CHIP_WIDTH_CM** 3
- #define **CHIP_HEIGHT_CM** 1.5

7.2.1 Detailed Description

7.3 Alpide Data Format

Collaboration diagram for Alpide Data Format:



Classes

- struct FrameStartFifoWord

  Data word stored in FRAME START FIFO. More...
- struct FrameEndFifoWord

  Data word stored in FRAME END FIFO. More...
- class AlpideDataWord

  The FIFOs in the Alpide chip are 24 bits, or 3 bytes, wide. This is a base class for the data words that holds 3 bytes, and is used as the data type in the SystemC FIFO templates. This class shouldn't be used on its own, the various types of data words are implemented in derived classes. More...
- class AlpideIdle
- class AlpideChipHeader
- class AlpideChipTrailer
- class AlpideChipEmptyFrame
- class AlpideRegionHeader
- class AlpideRegionTrailer
- class AlpideDataShort
- class AlpideDataLong
- class AlpideBusyOn
- class AlpideBusyOff
- class AlpideComma

Variables

- const uint8_t DW_IDLE = 0b11111111
- const uint8_t **DW_CHIP_HEADER** = 0b10100000
- const uint8_t **DW_CHIP_TRAILER** = 0b10110000
- const uint8_t **DW_CHIP_EMPTY_FRAME** = 0b11100000
- const uint8_t **DW_REGION_HEADER** = 0b11000000
- const uint8_t **DW_REGION_TRAILER** = 0b11110011
- const uint8_t **DW_DATA_SHORT** = 0b01000000
- const uint8_t **DW_DATA_LONG** = 0b00000000
- const uint8_t **DW_BUSY_ON** = 0b11110001
- const uint8_t **DW_BUSY_OFF** = 0b11110000
- const uint8_t DW_COMMA = 0b11111110
- const uint8_t **READOUT_FLAGS_BUSY_VIOLATION** = 0b00001000
- const uint8_t **READOUT_FLAGS_FLUSHED_INCOMPLETE** = 0b00000100
- const uint8_t **READOUT_FLAGS_STROBE_EXTENDED** = 0b00000010

- const uint8_t **READOUT_FLAGS_BUSY_TRANSITION** = 0b00000001
- const uint8_t MASK_IDLE_BUSY_COMMA = 0b11111111

  Mask for busy, idle and comma words.
- const uint8_t MASK_CHIP = 0b11110000

  Mask for chip header/trailer/empty frame words.
- const uint8_t MASK_REGION_HEADER = 0b11100000

  Mask for region header word.
- const uint8_t MASK_DATA = 0b11000000

  Mask for data short/long words.

7.3.1 Detailed Description

7.3.2 Class Documentation

7.3.2.1 struct FrameStartFifoWord

Data word stored in FRAME START FIFO.

Public Member Functions

- bool **operator==** (const FrameStartFifoWord &rhs) const
- FrameStartFifoWord & **operator=** (const FrameStartFifoWord &rhs)

Public Attributes

- bool **busy_violation**
- uint16_t **BC_for_frame**

Friends

- void **sc_trace** (sc_trace_file *tf, const FrameStartFifoWord &dw, const std::string &name)
- std::ostream & operator<< (std::ostream &stream, const FrameStartFifoWord &dw)

7.3.2.1.1 Friends And Related Function Documentation

7.3.2.1.1.1 std::ostream& operator<< (std::ostream & *stream,* const FrameStartFifoWord & *dw*) [friend]

Todo Overload this for all FrameStartFifoWord classes, so SystemC can print them to trace files properly?

7.3.2.2 struct FrameEndFifoWord

Data word stored in FRAME END FIFO.

Public Member Functions

- bool **operator==** (const FrameEndFifoWord &rhs) const
- FrameEndFifoWord & **operator=** (const FrameEndFifoWord &rhs)

Public Attributes

- bool **flushed_incomplete**
- bool **strobe_extended**
- bool **busy_transition**

Friends

- void **sc_trace** (sc_trace_file *tf, const FrameEndFifoWord &dw, const std::string &name)
- std::ostream & operator<< (std::ostream &stream, const FrameEndFifoWord &dw)

7.3.2.2.1 Friends And Related Function Documentation

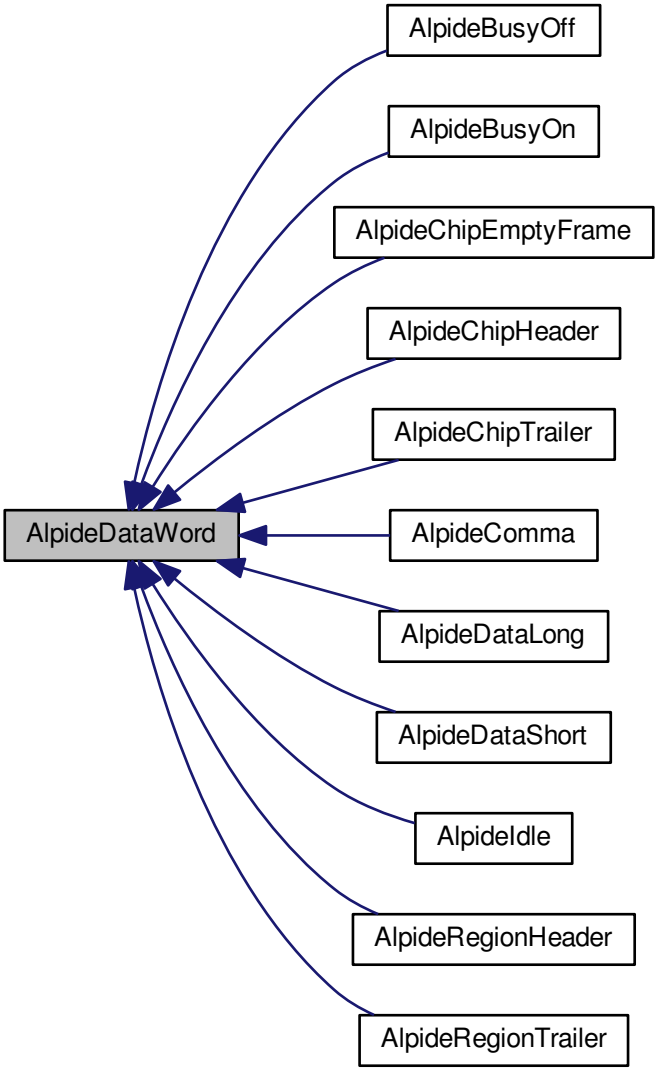
7.3.2.2.1.1 std::ostream& operator<< (std::ostream & *stream,* const FrameEndFifoWord & *dw*) [friend]

Todo Overload this for all FrameEndFifoWord classes, so SystemC can print them to trace files properly?

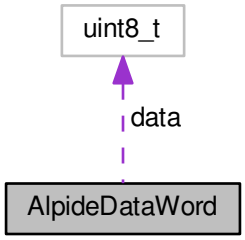
7.3.2.3 class AlpideDataWord

The FIFOs in the Alpide chip are 24 bits, or 3 bytes, wide. This is a base class for the data words that holds 3 bytes, and is used as the data type in the SystemC FIFO templates. This class shouldn't be used on its own, the various types of data words are implemented in derived classes.

Inheritance diagram for AlpideDataWord:



Collaboration diagram for AlpideDataWord:



Public Member Functions

- bool **signalBusyOn** (void)
- bool **signalBusyOff** (void)
- bool **operator==** (const AlpideDataWord &rhs) const
- AlpideDataWord & **operator=** (const AlpideDataWord &rhs)

Public Attributes

- uint8_t **data** [3]

Friends

- void **sc_trace** (sc_trace_file *tf, const AlpideDataWord &dw, const std::string &name)
- std::ostream & operator<< (std::ostream &stream, const AlpideDataWord &alpide_dw)

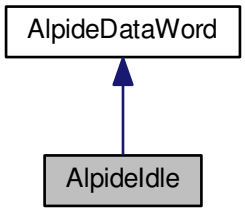
7.3.2.3.1 Friends And Related Function Documentation

7.3.2.3.1.1 std::ostream& operator<< (std::ostream & *stream,* const **AlpideDataWord** & *alpide_dw*) [friend]

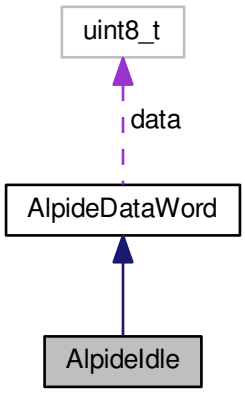
Todo Overload this for all AlpideDataWord classes, so SystemC can print them to trace files properly?

7.3.2.4 class AlpideIdle

Inheritance diagram for AlpideIdle:



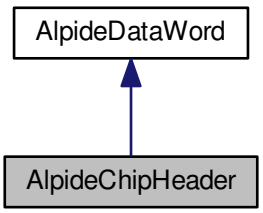
Collaboration diagram for AlpideIdle:



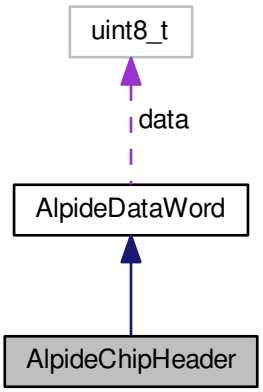
Additional Inherited Members

7.3.2.5 class AlpideChipHeader

Inheritance diagram for AlpideChipHeader:



Collaboration diagram for AlpideChipHeader:



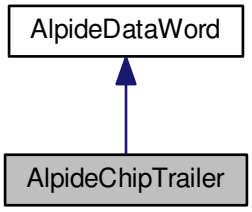
Public Member Functions

- **AlpideChipHeader** (uint8_t chip_id, uint16_t bunch_counter)
- **AlpideChipHeader** (uint8_t chip_id, FrameStartFifoWord &frame_start)

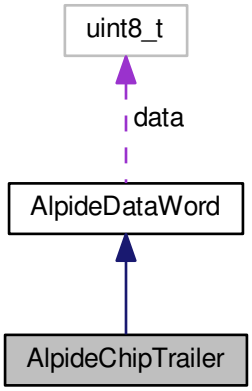
Additional Inherited Members

7.3.2.6 class AlpideChipTrailer

Inheritance diagram for AlpideChipTrailer:



Collaboration diagram for AlpideChipTrailer:



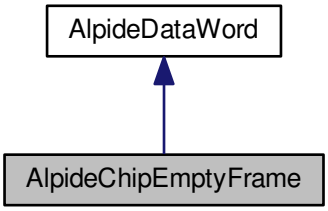
Public Member Functions

- **AlpideChipTrailer** (uint8_t readout_flags)
- **AlpideChipTrailer** (FrameStartFifoWord frame_start, FrameEndFifoWord frame_end, bool fatal_state, bool readout_abort)

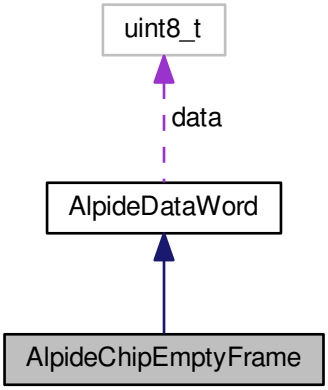
Additional Inherited Members

7.3.2.7 class AlpideChipEmptyFrame

Inheritance diagram for AlpideChipEmptyFrame:



Collaboration diagram for AlpideChipEmptyFrame:



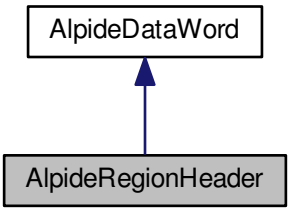
Public Member Functions

- **AlpideChipEmptyFrame** (uint8_t chip_id, uint16_t bunch_counter)
- **AlpideChipEmptyFrame** (uint8_t chip_id, FrameStartFifoWord &frame_start)

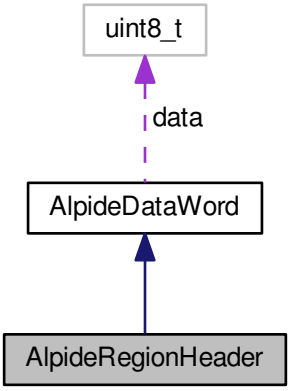
Additional Inherited Members

7.3.2.8 class AlpideRegionHeader

Inheritance diagram for AlpideRegionHeader:



Collaboration diagram for AlpideRegionHeader:



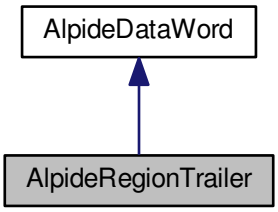
Public Member Functions

- **AlpideRegionHeader** (uint8_t region_id)

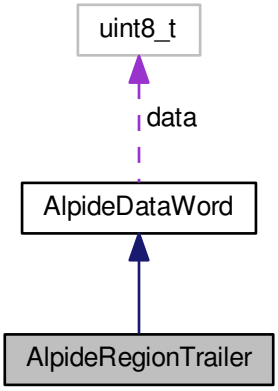
Additional Inherited Members

7.3.2.9 class AlpideRegionTrailer

Inheritance diagram for AlpideRegionTrailer:



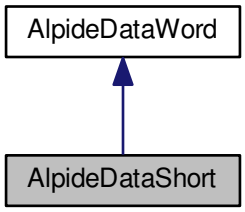
Collaboration diagram for AlpideRegionTrailer:



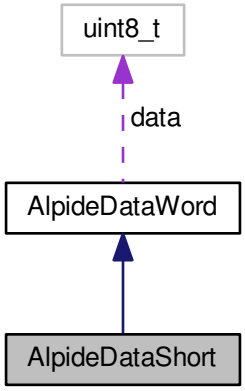
Additional Inherited Members

7.3.2.10 class AlpideDataShort

Inheritance diagram for AlpideDataShort:



Collaboration diagram for AlpideDataShort:



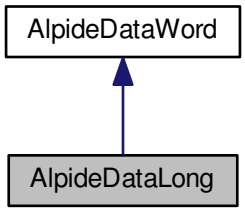
Public Member Functions

- **AlpideDataShort** (uint8_t encoder_id, uint16_t addr)

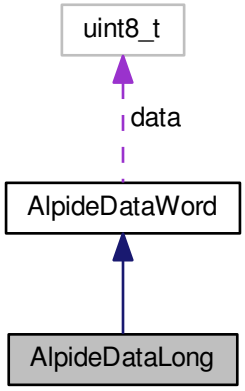
Additional Inherited Members

7.3.2.11 class AlpideDataLong

Inheritance diagram for AlpideDataLong:



Collaboration diagram for AlpideDataLong:



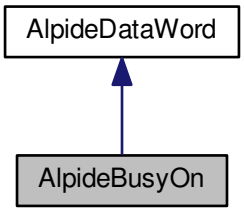
Public Member Functions

- **AlpideDataLong** (uint8_t encoder_id, uint16_t addr, uint8_t hitmap)

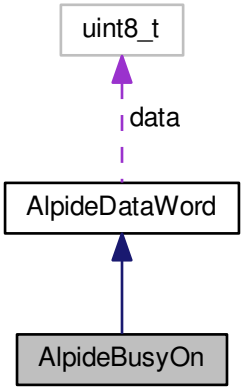
Additional Inherited Members

7.3.2.12 class AlpideBusyOn

Inheritance diagram for AlpideBusyOn:



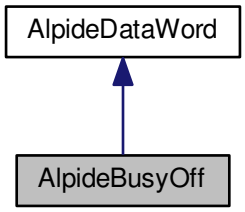
Collaboration diagram for AlpideBusyOn:



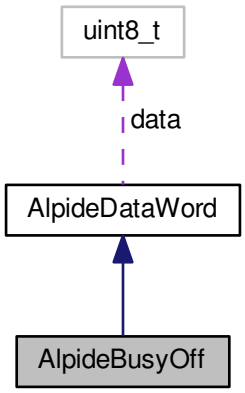
Additional Inherited Members

7.3.2.13 class AlpideBusyOff

Inheritance diagram for AlpideBusyOff:



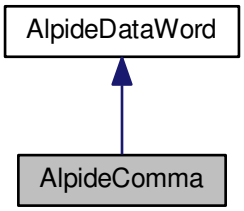
Collaboration diagram for AlpideBusyOff:



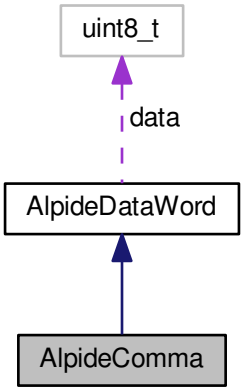
Additional Inherited Members

7.3.2.14 class AlpideComma

Inheritance diagram for AlpideComma:



Collaboration diagram for AlpideComma:



Additional Inherited Members

7.3.3 Variable Documentation

7.3.3.1 const uint8_t DW_COMMA = 0b11111110

This is not the correct COMMA word, but using this value instead makes this simulation model a bit simpler because the word cannot be confused with CHIP TRAILER 1111 1110 used to avoid confusion with CHIP TRAILER).

Referenced by AlpideEventBuilder::parseDataWord(), and AlpideEventBuilder::parseNonHeaderBytes().

7.3.3.2 const uint8_t DW_IDLE = 0b11111111

Alpide Data format and valid data words (from Alpide manual)

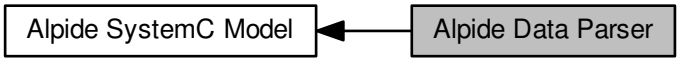
Data word	Header bits	Parameter bits
IDLE	1111 1111	None
CHIP HEADER	1010	<chip id[3:0]><BUNCH COUNTER FOR FRAME[10:3]>
CHIP TRAILER	1011	<readout flags[3:0]>
CHIP EMPTY FRAME	1110	<chip id[3:0]><BUNCH COUNTER FOR FRAME[10:3]>
REGION HEADER	110	<region id[4:0]>
DATA SHORT	01	<encoder id[3:0]><addr[9:0]>
DATA LONG	00	<encoder id[3:0]><addr[9:0]> 0 <hit map[6:0]> 1111 0001 1111
BUSY ON	1111 0001	None
BUSY OFF	1111 0000	None
COMMA	1011 1100	Note: 1111 1110 used instead in this code

Alpide data words, used to initialize the 24-bit FIFOs in the Alpide chip. The MSBs in the words identify datawords, the LSBs are parameters. Note 1: There is not a fixed width for the MSB identifier part. Note 2: Not to be confused with the definitions in AlpideDataTypes in alpide_data_parser.h, which is for identifying individual bytes in a datastream.

Referenced by AlpideEventBuilder::parseDataWord(), and AlpideEventBuilder::parseNonHeaderBytes().

7.4 Alpide Data Parser

Collaboration diagram for Alpide Data Parser:



Classes

- struct AlpideDataParsed
- class AlpideEventFrame
- class AlpideEventBuilder
- class AlpideDataParser

Enumerations

- enum AlpideDataTypes {
  ALPIDE_IDLE, **ALPIDE_CHIP_HEADER1**, **ALPIDE_CHIP_HEADER2**, **ALPIDE_CHIP_TRAILER**,
  ALPIDE_CHIP_EMPTY_FRAME1, **ALPIDE_CHIP_EMPTY_FRAME2**, **ALPIDE_REGION_HEADER**, **AL↩**
  PIDE_DATA_SHORT1,
  ALPIDE_DATA_SHORT2, **ALPIDE_DATA_LONG1**, **ALPIDE_DATA_LONG2**, **ALPIDE_DATA_LONG3**,
  ALPIDE_BUSY_ON, **ALPIDE_BUSY_OFF**, **ALPIDE_COMMA**, **ALPIDE_UNKNOWN** }

7.4.1 Detailed Description

7.4.2 Class Documentation

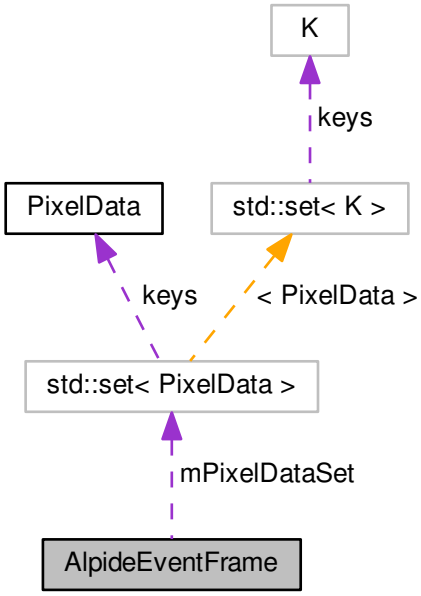
7.4.2.1 struct AlpideDataParsed

Public Attributes

- AlpideDataTypes **data** [3]

7.4.2.2 class AlpideEventFrame

Collaboration diagram for AlpideEventFrame:



Public Member Functions

- bool pixelHitInEvent (PixelData &pixel) const

  Look for a pixel hit in this event frame.
- void **setFrameCompleted** (bool val)
- bool **getFrameCompleted** (void)
- unsigned int **getEventSize** (void) const
- void **addPixelHit** (const PixelData &pixel)
- std::set< PixelData >::const_iterator **getPixelSetIterator** (void) const
- std::set< PixelData >::const_iterator **getPixelSetEnd** (void) const

Private Attributes

- std::set< PixelData > **mPixelDataSet**
- bool **mFrameCompleted**

7.4.2.2.1 Member Function Documentation

7.4.2.2.1.1 bool AlpideEventFrame::pixelHitInEvent (PixelData & *pixel*) const

Look for a pixel hit in this event frame.

Parameters

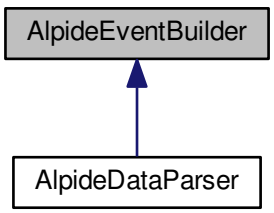
in	*pixel*	Reference to PixelData object

Returns

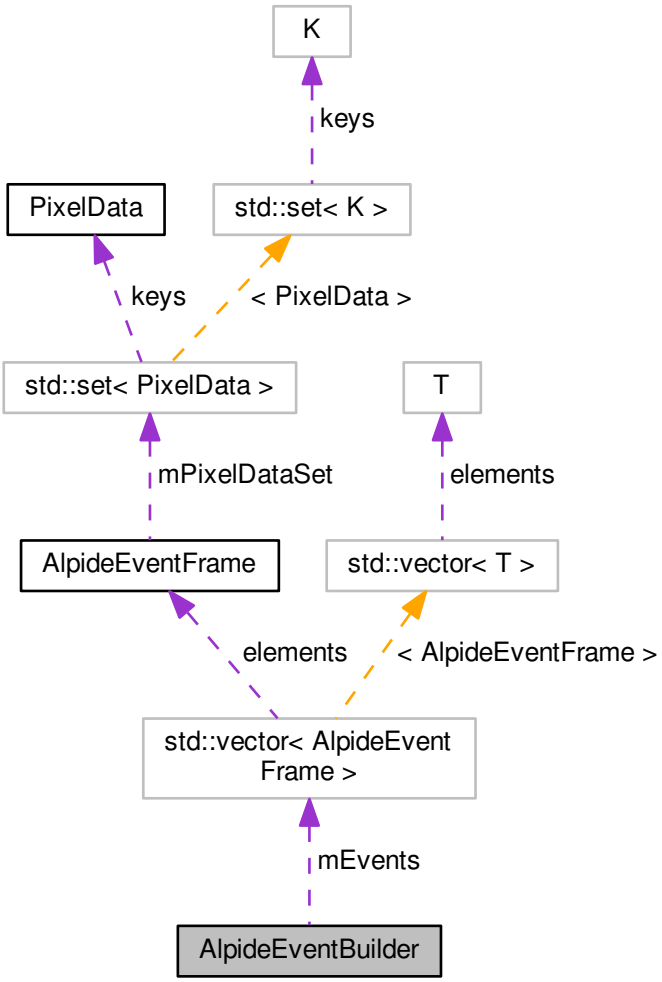
True if pixel is in event frame, false if not.

7.4.2.3 class AlpideEventBuilder

Inheritance diagram for AlpideEventBuilder:



Collaboration diagram for AlpideEventBuilder:



Public Member Functions

- unsigned int **getNumEvents** (void) const
- const AlpideEventFrame * getNextEvent (void) const

  Get a reference to the next event. This does not delete the event, and successive calls will return the same event unless popEvent() has been called.
- void popEvent (void)

  Pop/remove the oldest event (if there are any events, otherwise do nothing).
- void inputDataWord (AlpideDataWord dw)

  Takes a 3 byte Alpide data word as input, parses it, and depending on the data: 1) If this is a new Alpide data frame, a new AlpideEventFrame is created in mEvents 2) If this is data that belongs to the existing and most recent frame, hit data is added to that frame. 3) If these are just idle words etc., nothing is done with them.
- AlpideDataParsed parseDataWord (AlpideDataWord dw)

  Parse 3-byte Alpide data words the, and increase counters for the different types of data words. Note: the function only discovers what type of data word it is, it does nothing with the data word's parameters.

Private Member Functions

- AlpideDataTypes parseNonHeaderBytes (uint8_t data)

  Use this to parse the last 1-2 (least significant) bytes of a 24-bit Alpide data word, for words which are known to not utilize these bytes, e.g.: Data long uses all 3 bytes - don't use this function Data short uses first 2 bytes - use this function for the last byte Region header uses the first byte - use this function for the last two bytes The function will return either IDLE, BUSY_ON, BUSY_OFF, or UNKNOWN for these bytes. It will also increase counters for the corresponding words.

Private Attributes

- std::vector< AlpideEventFrame > **mEvents**
- unsigned int **mCurrentRegion** = 0
- long **mCommaCount**
- long **mIdleCount**
- long **mIdleByteCount**
- long **mBusyOnCount**
- long **mBusyOffCount**
- long **mDataShortCount**
- long **mDataLongCount**
- long **mRegionHeaderCount**
- long **mChipHeaderCount**
- long **mChipTrailerCount**
- long **mChipEmptyFrameCount**
- long **mUnknownDataWordCount**

7.4.2.3.1 Member Function Documentation

7.4.2.3.1.1 const AlpideEventFrame * AlpideEventBuilder::getNextEvent (void) const

Get a reference to the next event. This does not delete the event, and successive calls will return the same event unless popEvent() has been called.

Returns

Pointer to the next event if there are more events, nullptr if there are no events.

7.4.2.3.1.2 void AlpideEventBuilder::inputDataWord (AlpideDataWord *dw*)

Takes a 3 byte Alpide data word as input, parses it, and depending on the data: 1) If this is a new Alpide data frame, a new AlpideEventFrame is created in mEvents 2) If this is data that belongs to the existing and most recent frame, hit data is added to that frame. 3) If these are just idle words etc., nothing is done with them.

Parameters

in	*dw*	AlpideDataWord input to parse.

Todo Busy on here

Todo Busy off here

Todo Unknown Alpide data word received. Do something smart here?

7.4.2.3.1.3 AlpideDataParsed AlpideEventBuilder::parseDataWord (AlpideDataWord *dw*)

Parse 3-byte Alpide data words the, and increase counters for the different types of data words. Note: the function only discovers what type of data word it is, it does nothing with the data word's parameters.

Parameters

in	*dw*	AlpideDataWord input to parse.

Returns

AlpideDataParsed object with parsed data word type filled in for each byte

7.4.2.3.1.4 AlpideDataTypes AlpideEventBuilder::parseNonHeaderBytes (uint8_t *data*) [private]

Use this to parse the last 1-2 (least significant) bytes of a 24-bit Alpide data word, for words which are known to not utilize these bytes, e.g.: Data long uses all 3 bytes - don't use this function Data short uses first 2 bytes - use this function for the last byte Region header uses the first byte - use this function for the last two bytes The function will return either IDLE, BUSY_ON, BUSY_OFF, or UNKNOWN for these bytes. It will also increase counters for the corresponding words.

Parameters

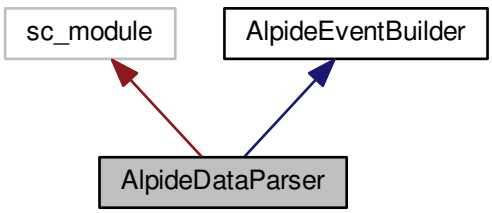
in	*data*	One of the "additional" bytes in a data word to parse

Returns

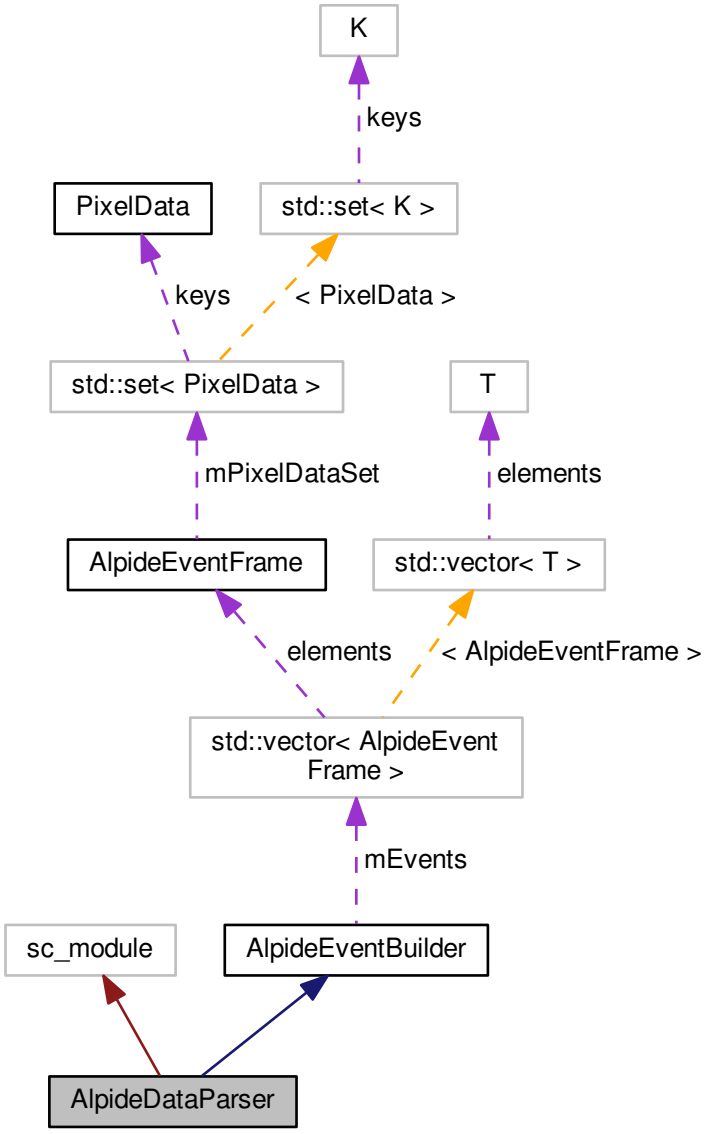
Data word type for "additional" byte provided in data argument

7.4.2.4 class AlpideDataParser

Inheritance diagram for AlpideDataParser:



Collaboration diagram for AlpideDataParser:



Public Member Functions

- **AlpideDataParser** (sc_core::sc_module_name name)
- void addTraces (sc_trace_file *wf, std::string name_prefix) const

  Add SystemC signals to log in VCD trace file.

Public Attributes

- sc_in< sc_uint< 24 > > **s_serial_data_in**
- sc_in_clk **s_clk_in**

Private Member Functions

- void parserInputProcess (void)

  Matrix readout SystemC method. Expects a 3-byte word input on each clock edge. The 3-byte data word is passed to the underlying base class for processing and event frame generation.

7.4.2.4.1 Member Function Documentation

7.4.2.4.1.1 void AlpideDataParser::addTraces (sc_trace_file * *wf*, std::string *name_prefix*) const

Add SystemC signals to log in VCD trace file.

Parameters

in,out	*wf*	Pointer to VCD trace file object
in	*name_prefix*	Name prefix to be added to all the trace names

Here is the call graph for this function:

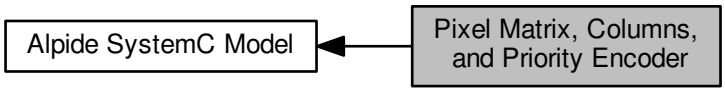
7.4.3 Enumeration Type Documentation

7.4.3.1 enum AlpideDataTypes

Enumerations used to identify the meaning of the different bytes in the data stream from the Alpide chip. Not to be confused with the definitions in alpide_data_format.h which are used to initialize the 24-bit FIFO words.

7.5 Pixel Matrix, Columns, and Priority Encoder

Collaboration diagram for Pixel Matrix, Columns, and Priority Encoder:



Classes

- class PixelData

  A struct that indicates a hit in a region, at the pixel identified by the col and row variables. For each hit an object of this type will be inserted into the std::set container in regionDataVector. For the pixels that don't have hits there will not be an object of this type inserted. Column should be 0 or 1. Row can be any value from 0 to N_PIXEL_ROWS-1. More...
- class PixelPriorityEncoder

  Comparator class/function for use with the PixelData class in the std::set container, which allows the container to sort the PixelData entries in a meaningful way. The picture below is from the ALPIDE operations manual, and shows: More...
- class PixelDoubleColumn
- class PixelMatrix

Functions

- const PixelData **NoPixelHit** (-1,-1)

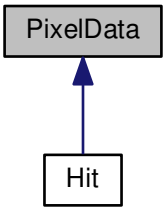
7.5.1 Detailed Description

7.5.2 Class Documentation

7.5.2.1 class PixelData

A struct that indicates a hit in a region, at the pixel identified by the col and row variables. For each hit an object of this type will be inserted into the std::set container in regionDataVector. For the pixels that don't have hits there will not be an object of this type inserted. Column should be 0 or 1. Row can be any value from 0 to N_PIXEL_ROWS-1.

Inheritance diagram for PixelData:



Public Member Functions

- **PixelData** (int col=0, int row=0)
- PixelData (int region, int pri_enc, int addr)

  Constructor for pixel data based on region number, priority encoder number in region, and pixel address in priority encoder. This is how the data are specified when they are transmitted as data long/short words.
- **PixelData** (const PixelData &p)
- bool **operator==** (const PixelData &rhs) const
- bool **operator**> (const PixelData &rhs) const
- bool **operator**< (const PixelData &rhs) const
- bool **operator**>**=** (const PixelData &rhs) const
- bool **operator**<**=** (const PixelData &rhs) const
- int **getCol** (void) const
- int **getRow** (void) const
- void **setCol** (const int col)
- void **setRow** (const int row)
- unsigned int getPriEncPixelAddress (void) const

  Get the "address" of this pixel within it's double column, that is the priority that this pixel has in the priority encoder would.
- unsigned int getPriEncNumInRegion (void) const

  Get the priority encoder that this pixel (column) belongs to, within the column's region. Hardcoded for 16 double columns per region.

Private Attributes

- int **mCol**
- int **mRow**

Friends

- class **PixelPriorityEncoder**

7.5.2.1.1 Constructor & Destructor Documentation

7.5.2.1.1.1 PixelData::PixelData (int *region,* int *pri_enc,* int *addr*)

Constructor for pixel data based on region number, priority encoder number in region, and pixel address in priority encoder. This is how the data are specified when they are transmitted as data long/short words.

Parameters

in	*region*	Region number
in	*pri_enc*	Priority encoder number in region (ie. double column number in region).
in	*addr*	Prioritized address in priority encoder

7.5.2.1.2 Member Function Documentation

7.5.2.1.2.1 unsigned int PixelData::getPriEncNumInRegion (void) const [inline]

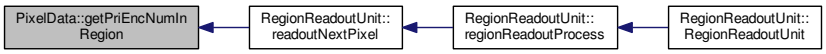
Get the priority encoder that this pixel (column) belongs to, within the column's region. Hardcoded for 16 double columns per region.

Returns

Priority encoder number.

Referenced by RegionReadoutUnit::readoutNextPixel().

Here is the caller graph for this function:



7.5.2.1.2.2 unsigned int PixelData::getPriEncPixelAddress (void) const [inline]

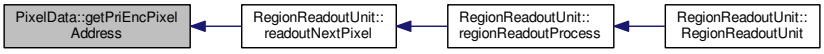
Get the "address" of this pixel within it's double column, that is the priority that this pixel has in the priority encoder would.

Returns

Pixel's priority encoder address/priority

Referenced by RegionReadoutUnit::readoutNextPixel().

Here is the caller graph for this function:



7.5.2.2 class PixelPriorityEncoder

Comparator class/function for use with the PixelData class in the std::set container, which allows the container to sort the PixelData entries in a meaningful way. The picture below is from the ALPIDE operations manual, and shows:

- Left: 512 rows x 1024 columns of pixels, divided into 32 regions
- Middle: 32 columns (16 double columns) x 512 rows in a region
- Right: Index/numbering/address of pixels within a double column, and the priority encoder between the columns. The priority encoder starts with the pixel that has the lowest address, and prioritizes them in increasing order.

The regionDataVector, which is declared as **std::vector<std::set<PixelData, PixelComparer> > regionData↩ Vector**; attempts to implement the priority encoder to reflect what is on the ALPIDE chip. Only pixels that have hits will be stored in the set, the pixels in the set are read out in increasing order (starting with index 0), and the PixelComparer type implements the actual prioritization of the pixels in the set.

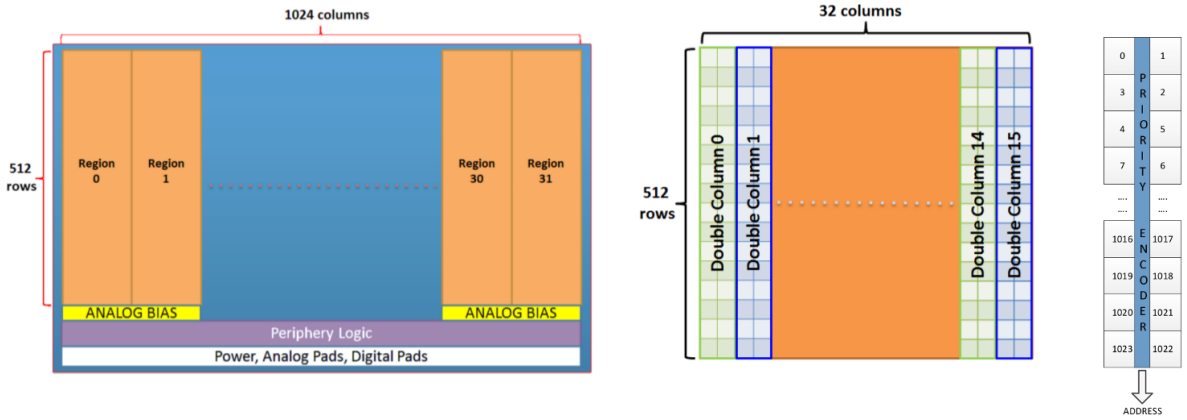


Figure 7.1 Overview of how regions and columns are indexed, and how pixels are indexed in double columns, in the Alpide chip.

Public Member Functions

- bool operator() (const PixelData &leftIn, const PixelData &rightIn)

  Overloaded () function, allows the std::set to use this function to compare two PixelData classes in the set, and determine which of them should come first when sorting them. The prioritization works like this:

7.5.2.2.1 Member Function Documentation

7.5.2.2.1.1 bool PixelPriorityEncoder::operator() (const PixelData & *leftIn*, const PixelData & *rightIn*) [inline]

Overloaded () function, allows the std::set to use this function to compare two PixelData classes in the set, and determine which of them should come first when sorting them. The prioritization works like this:

- Lower rows prioritized first
- For even rows, the column 0 pixel comes first
- For odd rows, the column 1 pixel comes first

Parameters

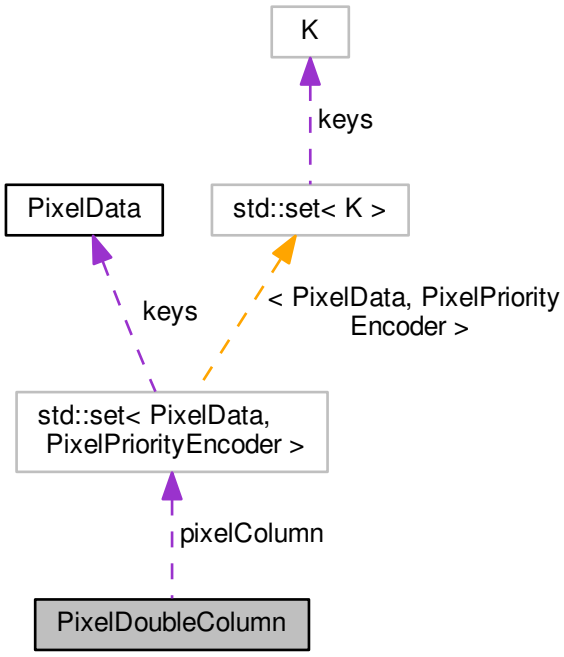
leftIn	Left side argument
rightIn	Right side argument

Returns

True if leftIn has highest priority, false if rightIn has higest priority

7.5.2.3 class PixelDoubleColumn

Collaboration diagram for PixelDoubleColumn:



Public Member Functions

- void setPixel (unsigned int col_num, unsigned int row_num)

  Set a pixel in a pixel double column object.
- void clear (void)

  Clear (flush) contents of double column.
- bool inspectPixel (unsigned int col_num, unsigned int row_num)

  Check if there is a hit or not for the pixel specified by col_num and row_num, without deleting the pixel from the MEB.
- PixelData readPixel (void)

  Read out the next pixel from this double column, and erase it from the MEB. Pixels are read out in an order corresponding to that of the priority encoder in the Alpide chip.
- unsigned int pixelHitsRemaining (void)

  Returns how many pixel hits (in this double column) that have not been read out from the MEBs yet.

Private Attributes

- std::set< PixelData, PixelPriorityEncoder > **pixelColumn**

7.5.2.3.1 Member Function Documentation

7.5.2.3.1.1 bool PixelDoubleColumn::inspectPixel (unsigned int *col_num,* unsigned int *row_num*)

Check if there is a hit or not for the pixel specified by col_num and row_num, without deleting the pixel from the MEB.

Parameters

in	*col_num*	column number of pixel, must be 0 or 1.
in	*row_num*	row number of pixel, must be in the range 0 to N_PIXEL_ROWS-1

Returns

True if there is a hit, false if not.

Exceptions

std::out_of_range	if col_num or row_num is not in the specified range.

7.5.2.3.1.2 PixelData PixelDoubleColumn::readPixel (void)

Read out the next pixel from this double column, and erase it from the MEB. Pixels are read out in an order corresponding to that of the priority encoder in the Alpide chip.

Returns

PixelData with hit coordinates. If no pixel hits exist, NoPixelHit is returned (PixelData object with coords = (-1,-1)).

7.5.2.3.1.3 void PixelDoubleColumn::setPixel (unsigned int *col_num,* unsigned int *row_num*)

Set a pixel in a pixel double column object.

Parameters

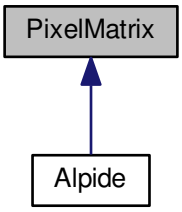
in	*col_num*	column number of pixel, must be 0 or 1.
in	*row_num*	row number of pixel, must be in the range 0 to N_PIXEL_ROWS-1

Exceptions

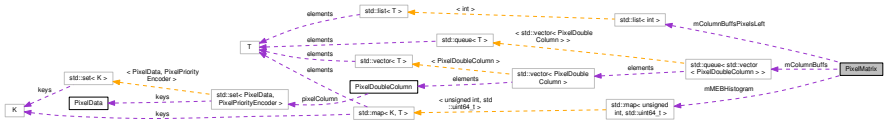
std::out_of_range	if col_num or row_num is not in the specified range.

7.5.2.4 class PixelMatrix

Inheritance diagram for PixelMatrix:



Collaboration diagram for PixelMatrix:



Public Member Functions

- PixelMatrix (bool continuous_mode)

  PixelMatrix Constructor.
- void newEvent (uint64_t event_time)

  Indicate to the Alpide that we are starting on a new event. If the call is successful a new MEB slice is created, and the next calls to setPixel will add pixels to the new event.
- void deleteEvent (uint64_t event_time)

  Delete the oldest event from the MEB (if there are any events at all, calling this function with no events is fine).
- void flushOldestEvent (void)

  Flush the oldest event by clearing all double columns and setting its size to zero.
- void setPixel (unsigned int col, unsigned int row)

  Set the pixel (ie. the pixel is hit) specified by col_num and row_num, in the most recent event buffer.
- bool regionEmpty (int start_double_col, int stop_double_col)

  Check if the region denoted by start_double_col and stop_double_col is empty.
- bool regionEmpty (int region)

  Check if a region of the pixel matrix is empty.
- PixelData readPixel (uint64_t time_now, int start_double_col=0, int stop_double_col=N_PIXEL_COLS/2)

  Read out the next pixel from the pixel matrix, and erase it from the MEB. This member function will read out pixels from the oldest event buffer. The pixels will be by default be read out from the double columns in consecutive order from 0 to (N_PIXEL_COLS/2)-1, or optionally from the double column range specified by start_double_col and stop↩ _double_col. Regions are not read out in parallel with this function. But note that within a double column the pixels will be read out with the order used by the priority encoder in the Alpide chip.
- PixelData readPixelRegion (int region, uint64_t time_now)

Read out the next pixel from the specified region in the pixel matrix, and erase it from the MEB. This member function will read out pixels from the oldest event buffer. The pixels in the desired region will be read out from the double columns in consecutive order from 0 to N_PIXEL_DOUBLE_COLS_PER_REGION-1. Note that within a double column the pixels will be read out with the order used by the priority encoder in the Alpide chip.

- int **getNumEvents** (void)
- int getHitsRemainingInOldestEvent (void)

  Return the number of hits in the oldest of the events stored in multi event buffers.
- int getHitTotalAllEvents (void)

  Get total number of hits in all Multi Event Buffers.
- std::map< unsigned int, std::uint64_t > **getMEBHisto** (void) const

Protected Attributes

- bool mContinuousMode

  True: Continuous, False: Triggered.

Private Attributes

- std::queue< std::vector< PixelDoubleColumn > > mColumnBuffs

  mColumnBuffs holds multi event buffers of pixel columns The queue represent the MEBs, and the vector the pixel columns.
- std::list< int > mColumnBuffsPixelsLeft

  Each entry here corresponds to one entry in mColumnBuffs. This variable keeps track of the number of pixel left in the columns in each entry in mColumnBuffs.
- std::map< unsigned int, std::uint64_t > mMEBHistogram

  This map contains histogram values over MEB usage. The key is the number of MEBs in use, and the value is the total time duration for that key.
- uint64_t mMEBHistoLastUpdateTime = 0

  Last time the MEB histogram was updated.

7.5.2.4.1 Constructor & Destructor Documentation

7.5.2.4.1.1 PixelMatrix::PixelMatrix (bool *continuous_mode*)

PixelMatrix Constructor.

Parameters

in	*continuous_mode*	True: continuous mode, false: triggered mode

7.5.2.4.2 Member Function Documentation

7.5.2.4.2.1 void PixelMatrix::deleteEvent (uint64_t *time_now*)

Delete the oldest event from the MEB (if there are any events at all, calling this function with no events is fine).

Parameters

in	*time_now*	Simulation time when this readout is occuring

Exceptions

out_of_range	If there are no events, or if col or row is outside the allowed range

Referenced by Alpide::frameReadout().

Here is the caller graph for this function:

7.5.2.4.2.2 int PixelMatrix::getHitsRemainingInOldestEvent (void)

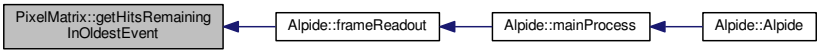
Return the number of hits in the oldest of the events stored in multi event buffers.

Returns

Number of hits in oldest event. If there are no events left, return zero.

Referenced by Alpide::frameReadout().

Here is the caller graph for this function:



7.5.2.4.2.3 int PixelMatrix::getHitTotalAllEvents (void)

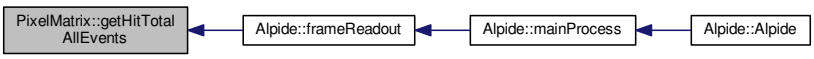
Get total number of hits in all Multi Event Buffers.

Returns

Total number of hits.

Referenced by Alpide::frameReadout().

Here is the caller graph for this function:



7.5.2.4.2.4 void PixelMatrix::newEvent (uint64_t *event_time*)

Indicate to the Alpide that we are starting on a new event. If the call is successful a new MEB slice is created, and the next calls to setPixel will add pixels to the new event.

Parameters

in	*event_time*	Simulation time when the event is pushed/latched into MEB (use current simulation time).

Referenced by Alpide::strobeInput().

Here is the caller graph for this function:

7.5.2.4.2.5 PixelData PixelMatrix::readPixel (uint64_t *time_now*, int *start_double_col* = 0, int *stop_double_col* = N_PIXEL_COLS/2)

Read out the next pixel from the pixel matrix, and erase it from the MEB. This member function will read out pixels from the oldest event buffer. The pixels will be by default be read out from the double columns in consecutive order from 0 to (N_PIXEL_COLS/2)-1, or optionally from the double column range specified by start_double_col and stop_double_col. Regions are not read out in parallel with this function. But note that within a double column the pixels will be read out with the order used by the priority encoder in the Alpide chip.

Parameters

in	*time_now*	Simulation time when this readout is occuring
in	*start_double_col*	Start double column to start searching for pixels to readout from
in	*stop_double_col*	Stop searching for pixels to read out when reaching this column

Returns

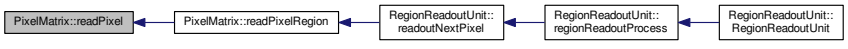
PixelData with hit coordinates. If no pixel hits exist, NoPixelHit is returned (PixelData object with coords = (-1,-1)).

Exceptions

std::out_of_range	if start_double_col is less than zero, or larger than (N_PIXEL_COLS/2)-1.
std::out_of_range	if stop_double_col is less than one, or larger than N_PIXEL_COLS/2.
std::out_of_range	if stop_double_col is greater than or equal to start_double_col

Referenced by readPixelRegion().

Here is the caller graph for this function:



7.5.2.4.2.6 PixelData PixelMatrix::readPixelRegion (int *region*, uint64_t *time_now*)

Read out the next pixel from the specified region in the pixel matrix, and erase it from the MEB. This member function will read out pixels from the oldest event buffer. The pixels in the desired region will be read out from the double columns in consecutive order from 0 to N_PIXEL_DOUBLE_COLS_PER_REGION-1. Note that within a double column the pixels will be read out with the order used by the priority encoder in the Alpide chip.

Parameters

in	*region*	The region number to read out a pixel from
in	*time_now*	Simulation time when this readout is occuring. Required for updating histogram data in case an MEB is done reading out.

Returns

PixelData with hit coordinates. If no pixel hits exist, NoPixelHit is returned (PixelData object with coords = (-1,-1)).

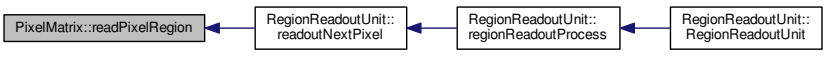
Exceptions

std::out_of_range	if region is less than zero, or greater than N_REGIONS-1

Referenced by RegionReadoutUnit::readoutNextPixel().

Here is the call graph for this function:

Here is the caller graph for this function:



7.5.2.4.2.7 bool PixelMatrix::regionEmpty (int *start_double_col*, int *stop_double_col*)

Check if the region denoted by start_double_col and stop_double_col is empty.

Parameters

in	*start_double_col*	Start of region in terms of double columns
in	*stop_double_col*	End of region in terms of double columns

Returns

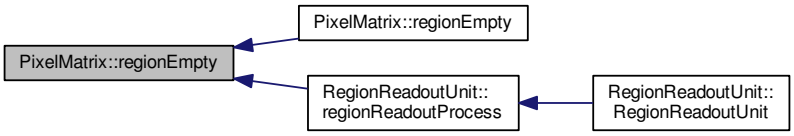
True if empty

Exceptions

std::out_of_range	if start_double_col is less than zero, or larger than (N_PIXEL_COLS/2)-1.
std::out_of_range	if stop_double_col is less than one, or larger than N_PIXEL_COLS/2.
std::out_of_range	if stop_double_col is greater than or equal to start_double_col

Referenced by regionEmpty(), and RegionReadoutUnit::regionReadoutProcess().

Here is the caller graph for this function:



7.5.2.4.2.8 bool PixelMatrix::regionEmpty (int *region*)

Check if a region of the pixel matrix is empty.

Parameters

in	*region*	The region number to check

Returns

PixelData with hit coordinates. If no pixel hits exist, NoPixelHit is returned (PixelData object with coords = (-1,-1)).

Exceptions

std::out_of_range	if region is less than zero, or greater than N_REGIONS-1

Here is the call graph for this function:

7.5.2.4.2.9 void PixelMatrix::setPixel (unsigned int *col,* unsigned int *row*)

Set the pixel (ie. the pixel is hit) specified by col_num and row_num, in the most recent event buffer.

Parameters

in	*col*	Column (0 to N_PIXEL_COLS-1).
in	*row*	Row (0 to N_PIXEL_ROWS-1).

Exceptions

out_of_range	If there are no events, or if col or row is outside the allowed range

Referenced by TriggerEvent::feedHitsToChip().

Here is the caller graph for this function:

7.5.2.4.3 Member Data Documentation

7.5.2.4.3.1 std::queue< std::vector<**PixelDoubleColumn**> > PixelMatrix::mColumnBuffs [private]

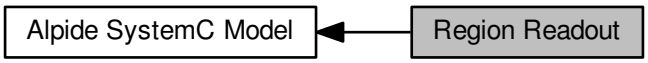
mColumnBuffs holds multi event buffers of pixel columns The queue represent the MEBs, and the vector the pixel columns.

Todo Implement event ID somewhere. Maybe make an MEB class, and use it as the datatype for this queue?

Referenced by deleteEvent(), flushOldestEvent(), getHitsRemainingInOldestEvent(), getHitTotalAllEvents(), newEvent(), readPixel(), regionEmpty(), and setPixel().

7.6 Region Readout

Collaboration diagram for Region Readout:



Classes

- class RegionReadoutUnit

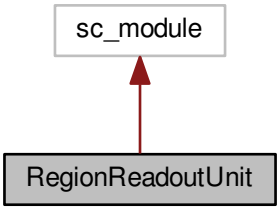
7.6.1 Detailed Description

7.6.2 Class Documentation

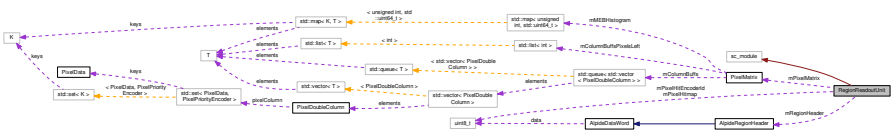
7.6.2.1 class RegionReadoutUnit

The RegionReadoutUnit class is a simple representation of the RRU in the Alpide chip. It has a member function that accepts pixel hits inputs, the RRU class will hold on to these pixels to determine if there are several pixels in the same cluster, and then put DATA_LONG or DATA_SHORT words into a SystemC FIFO.

Inheritance diagram for RegionReadoutUnit:



Collaboration diagram for RegionReadoutUnit:



Public Member Functions

- RegionReadoutUnit (sc_core::sc_module_name name, PixelMatrix *matrix, unsigned int region_num, unsigned int fifo_size, bool matrix_readout_speed, bool cluster_enable)

  Constructor for RegionReadoutUnit class.
- void regionReadoutProcess (void)

  SystemC process/method that implements the logic in the Region Readout Unit (RRU). NOTE: Should run at system clock frequency (40MHz).
- void **regionMatrixReadoutFSM** (void)
- void regionValidFSM (void)

  SystemC process/method that implements the state machine that determines if the region is valid (has data this frame) Note: should run on Alpide system clock frequency.
- void regionHeaderFSM (void)

  SystemC process/method that implements the state machine that determines when the region header should be outputted Note: should run on Alpide system clock frequency.
- void addTraces (sc_trace_file *wf, std::string name_prefix) const

  Add SystemC signals to log in VCD trace file.

Public Attributes

- sc_in_clk s_system_clk_in

  40MHz LHC clock
- sc_in< bool > s_frame_readout_start_in
- sc_in< bool > **s_readout_abort_in**
- sc_in< bool > s_region_event_start_in

  This comes from TRU, when reaodut of next frame from region FIFO to TRU FIFO should start.
- sc_in< bool > s_region_event_pop_in

  This comes from TRU, when reaodut of next frame from region FIFO to TRU FIFO should start.
- sc_in< bool > **s_region_data_read_in**
- sc_out< bool > **s_frame_readout_done_out**
- sc_out< bool > **s_region_fifo_empty_out**
- sc_out< bool > **s_region_valid_out**
- sc_out< AlpideDataWord > **s_region_data_out**

Private Member Functions

- bool readoutNextPixel (PixelMatrix &matrix)

  Read out the next pixel from this region's priority encoder. NOTE: This function should be called from a process that runs at the priority encoder readout clock. The function here will look for pixel clusters and generate DATA LONG words when possible if clustering is enabled, otherwise it will only send DATA SHORT words. See the flowchart for a better explanation of how this function works.
- void flushRegionFifo (void)

  Flush the region fifo. Used in data overrun mode. The function assumes that the fifo can be flushed in one clock cycle.

Private Attributes

- sc_signal< sc_uint< 8 > > **s_rru_readout_state**
- sc_signal< sc_uint< 8 > > **s_rru_valid_state**
- sc_signal< sc_uint< 1 > > **s_rru_header_state**
- sc_signal< bool > **s_generate_region_header**
- sc_signal< bool > s_region_matrix_empty_debug

  Delayed one clock cycle compared to when it is used..
- sc_signal< sc_uint< 2 > > **s_matrix_readout_delay_counter**
- tlm::tlm_fifo< AlpideDataWord > **s_region_fifo**
- sc_signal< sc_uint< 8 > > **s_region_fifo_size**
- AlpideRegionHeader **mRegionHeader**
- unsigned int mRegionId

  The region handled by this RRU.
- bool mMatrixReadoutSpeed
- bool mMatrixReadoutCounter

  Used with mMatrixReadoutSpeed to implement a delay when readout out pixel matrix.
- std::uint16_t mPixelHitBaseAddr

  Corresponds to pixel address in DATA SHORT/LONG words, in priority encoder order.
- std::uint8_t mPixelHitEncoderId
- std::uint8_t mPixelHitmap

  Corresponds to hitmap in DATA LONG word.
- unsigned int **mFifoSizeLimit**
- bool **mFifoSizeLimitEnabled**
- bool **mBusySignaled**
- bool **mClusteringEnabled**
- bool mClusterStarted

  Used in conjunction with mClusteringEnabled. Indicates that we have already received the first pixel in a potential cluster (stored in mPixelHitBaseAddr), and should continue building this cluster with subsequent hits that fall into the same pixel cluster range.
- PixelMatrix * **mPixelMatrix**

7.6.2.1.1 Constructor & Destructor Documentation

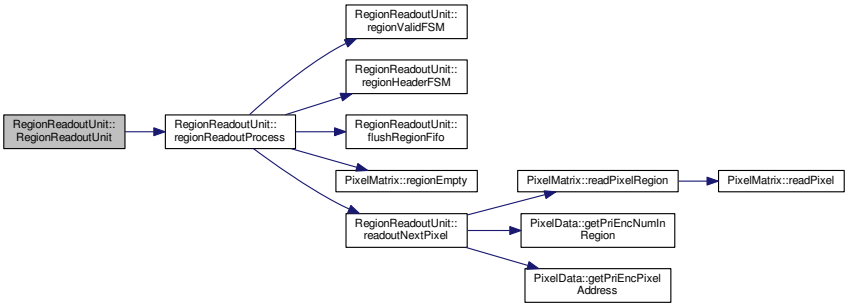
7.6.2.1.1.1 RegionReadoutUnit::RegionReadoutUnit (sc_core::sc_module_name *name*, PixelMatrix * *matrix*, unsigned int *region_num*, unsigned int *fifo_size*, bool *matrix_readout_speed*, bool *cluster_enable*)

Constructor for RegionReadoutUnit class.

Parameters

in	*name*	SystemC module name
in	*matrix*	Reference to pixel matrix
in	*region_num*	The region number that this RRU is assigned to
in	*fifo_size*	Size limit on the RRU's FIFO. 0 for no limit.
in	*matrix_readout_speed*	True for fast readout (2 clock cycles), false is slow (4 cycles).
in	*cluster_enable*	Enable/disable clustering and use of DATA LONG data words

Here is the call graph for this function:



7.6.2.1.2 Member Function Documentation

7.6.2.1.2.1 void RegionReadoutUnit::addTraces (sc_trace_file * *wf*, std::string *name_prefix*) const

Add SystemC signals to log in VCD trace file.

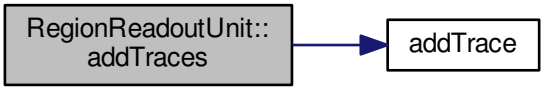
Parameters

in,out	*wf*	Pointer to VCD trace file object
in	*name_prefix*	Name prefix to be added to all the trace names

Todo Probably need to a stream << operator to allow values from fifo to be printed to trace file

Todo Probably need to a stream << operator to allow values from fifo to be printed to trace file

Here is the call graph for this function:



7.6.2.1.2.2 bool RegionReadoutUnit::readoutNextPixel (PixelMatrix & *matrix*) [private]

Read out the next pixel from this region's priority encoder. NOTE: This function should be called from a process that runs at the priority encoder readout clock. The function here will look for pixel clusters and generate DATA LONG words when possible if clustering is enabled, otherwise it will only send DATA SHORT words. See the flowchart for a better explanation of how this function works.

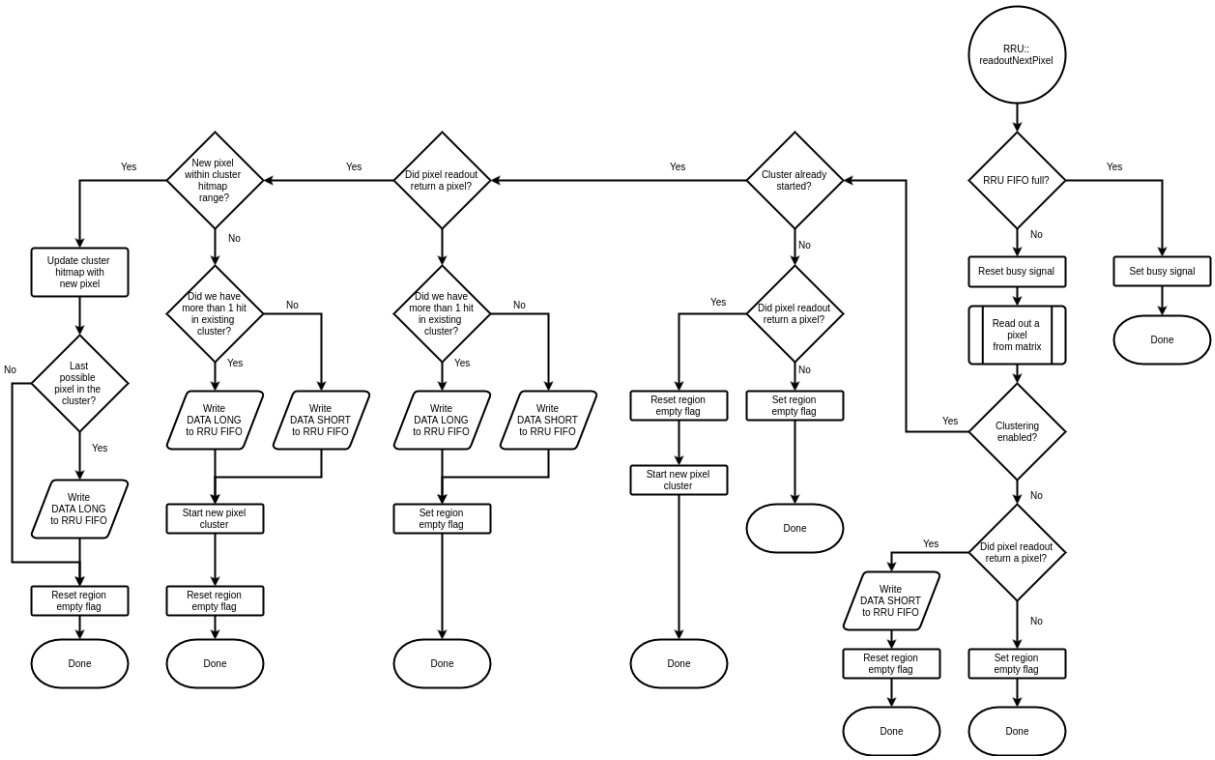


Figure 7.2 Flowchart for pixel readout and clustering in readoutNextPixel()

Parameters

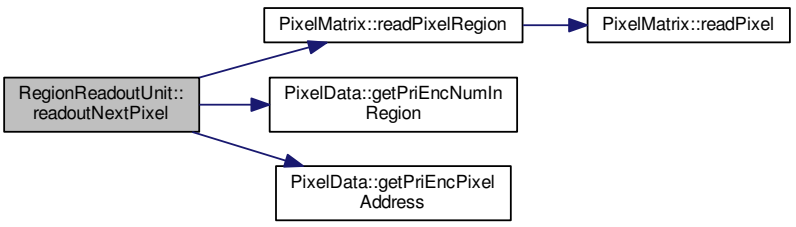
in	*matrix*	Reference to pixel matrix

Returns

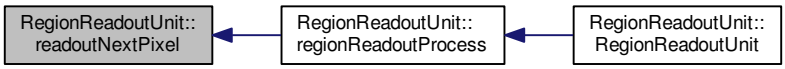
False if matrix is empty and no pixel was read out

Referenced by regionReadoutProcess().

Here is the call graph for this function:



Here is the caller graph for this function:



7.6.2.1.3 Member Data Documentation

7.6.2.1.3.1 bool RegionReadoutUnit::mMatrixReadoutSpeed [private]

Corresponds to Matrix Readout Speed bit in 0x0001 Mode Control register in Alpide chip. True: 20MHz readout. False: 10MHz readout.

Referenced by regionReadoutProcess().

7.6.2.1.3.2 std::uint8_t RegionReadoutUnit::mPixelHitEncoderId [private]

Corresponds to priority encoder id in DATA SHORT/LONG words, which is the priority encoder id (within the current region) that the current pixel belongs to.

Referenced by readoutNextPixel().

7.6.2.1.3.3 sc_in<bool> RegionReadoutUnit::s_frame_readout_start_in

This signal comes from FROMU, on deassertion of trigger, and indicates that start of readout from current pixel matrix event buffer to region FIFO can start

Referenced by addTraces(), and regionReadoutProcess().

7.7 Event Generation

Classes

- class EventGenerator

  A simple event generator for Alpide SystemC simulation model. More...
- class Hit
- class TriggerEvent

Macros

- #define **N_CHIPS** 108

Variables

- const TriggerEvent NoTriggerEvent

  A TriggerEvent that equals NoTriggerEvent is returned by some of the EventGenerator's functions which return a reference to an event, when there is no TriggerEvent to return.

7.7.1 Detailed Description

7.7.2 Class Documentation

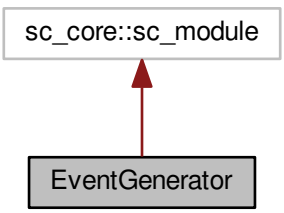
7.7.2.1 class EventGenerator

A simple event generator for Alpide SystemC simulation model.

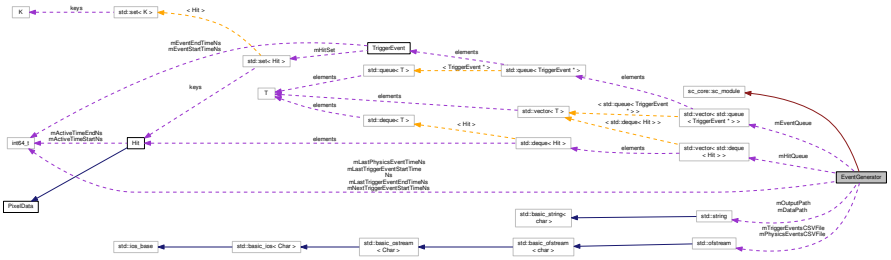
Physics events are generated at a rate that has an exponential distribution, with Lambda = 1 / average rate. The number of hits generated (hit multiplicity) per event can be based on a gaussian distribution or a user-defined discrete distribution. The ROOT macro export_multiplicity_data.cxx found under process/Multiplicity_distribution can be used to generate a discrete distribution based on real multiplicity data from ALICE.

The hits will currently be disributed randomly (with a flat/uniform distribution) among the different chips and over a chip's x/y coordinates. For each hit a fixed 2x2 pixel cluster is generated on the chip (this might be replaced with a more advanced random distribution in the future).

Inheritance diagram for EventGenerator:



Collaboration diagram for EventGenerator:



Public Member Functions

- EventGenerator (sc_core::sc_module_name name, const QSettings *settings, std::string output_path)

  Constructor for EventGenerator.
- void **generateNextEvent** ()
- void **generateNextEvents** (int n_events)
- const TriggerEvent & getNextTriggerEvent (void)

  Get a reference to the next event (if there is one). Note: this function will keep returning the same event until it has been removed by removeOldestEvent().
- void setBunchCrossingRate (int rate_ns)

  Sets the bunch crossing rate, and recalculates the average crossing rate.
- void setRandomSeed (int seed)

  Sets the random seed used by random number generators.
- void initRandomNumGenerator (void)

  Initialize random number generators.
- void **setPath** (const std::string &path)
- void **enableWriteToDisk** (void)
- void **disableWriteToDisk** (void)
- void **setNumEventsInMemAllowed** (int n)
- int **getTriggerFilterTime** (void) const
- int **getEventsInMem** (void) const
- int **getPhysicsEventCount** (void) const
- int **getTriggerEventCount** (void) const
- void removeOldestEvent (void)

  Remove the oldest event from the event queue (if there are any events in the queue, otherwise do nothing).
- void physicsEventProcess (void)

  SystemC controled method, should be sensitive to the positive edge of the clock. Responsible for 1) Creating new physics events (hits) 2) Deleting old inactive hits.
- void triggerEventProcess (void)

  SystemC controlled method. It should be sensitive to the strobe signal, (both rising and falling edge) and is responsible for creating the triggerEvent objects after a STROBE pulse.

Public Attributes

- sc_in< bool > **s_strobe_in**
- sc_in_clk **s_clk_in**
- sc_event_queue_port **E_trigger_event_available**
- sc_out< bool > s_physics_event_out

Private Member Functions

- void **calculateAverageCrossingRate** (void)
- void eventMemoryCountLimiter (void)

  Limit the number of events stored in memory, as specified by mNumEventsInMemoryAllowed. The oldest events will be removed to bring the count below the threshold. If mWriteEventsToDisk is true, then the events that are removed will be written to disk.
- TriggerEvent * generateNextTriggerEvent (int64_t event_start, int64_t event_end, int chip_id)

  Create a new trigger event at the given start time. It checks if trigger event should be filtered or not, and updates trigger ID count.
- int64_t generateNextPhysicsEvent (void)

  Generate the next physics event (in the future). 1) Generate time till the next physics event 2) Generate hits for the next event, and put them on the hit queue 3) Update counters etc.
- void readDiscreteDistributionFile (const char *filename, std::vector< double > &dist_vector) const

  Read a discrete distribution from file and store it in a vector. The file format is a simple text file, with the following format: X0 Y0 X1 Y1 ... Xn Yn.
- void scaleDiscreteDistribution (std::vector< double > &dist_vector, double new_mean_value)

  Scale the x axis of a discrete distribution, so that the distribution gets a new mean value.
- unsigned int getRandomMultiplicity (void)

  Return a random number of hits (multiplicity) based on the chosen distribution for multiplicity.
- void addHitsToTriggerEvent (TriggerEvent &e)

  Iterate through the hit queue corresponding to the chip_id associated with the event referenced by e, and add the active hits to it.
- void removeInactiveHits (void)

  Remove old hits. Start at the front of the hit queue, and pop (remove) hits from the front while the hits are no longer active at current simulation time, and older than the oldest trigger event (so we don't delete hits that may be still be used in a trigger event that hasn't been processed yet).

Private Attributes

- std::vector< std::queue< TriggerEvent * > > mEventQueue
- std::vector< std::deque< Hit > > mHitQueue
- int **mNumChips**
- int **mBunchCrossingRateNs**
- int **mAverageEventRateNs**
- int mNumEventsInMemoryAllowed = 0

  Number of events to keep in memory at a time. 0 = infinite.
- int mPhysicsEventCount = 0

  Total number of physics and trigger events generated.
- int **mTriggerEventIdCount** = 0
- int64_t mLastPhysicsEventTimeNs = 0

  Time of the last physics event that was generated.
- int64_t mLastTriggerEventStartTimeNs = 0
- int64_t **mLastTriggerEventEndTimeNs** = 0
- bool **mStrobeActive** = false
- int64_t mNextTriggerEventStartTimeNs = 0
- int mNextTriggerEventChipId = 0
- int **mPixelDeadTime**
- int **mPixelActiveTime**
- int mTriggerFilterTimeNs
- bool **mTriggerFilteringEnabled** = false
- bool **mContinuousMode** = false
- std::string mDataPath = "data"

- std::string **mOutputPath**
- bool **mWriteEventsToDisk** = false
- bool **mCreateCSVFile** = true
- std::ofstream **mPhysicsEventsCSVFile**
- std::ofstream **mTriggerEventsCSVFile**
- int **mRandomSeed**
- boost::random::mt19937 **mRandHitGen**
- boost::random::mt19937 **mRandHitMultiplicityGen**
- boost::random::mt19937 **mRandEventTimeGen**
- boost::random::uniform_int_distribution< int > * mRandHitChipID

  Uniform distribution used generating hit coordinates.
- boost::random::uniform_int_distribution< int > * **mRandHitChipX**
- boost::random::uniform_int_distribution< int > * **mRandHitChipY**
- boost::random::discrete_distribution * mRandHitMultiplicityDiscrete
- boost::random::normal_distribution< double > * **mRandHitMultiplicityGauss**
- boost::random::exponential_distribution< double > * mRandEventTime

  Exponential distribution used for time between events.
- int **mHitMultiplicityGaussAverage**
- int **mHitMultiplicityGaussDeviation**

7.7.2.1.1 Constructor & Destructor Documentation

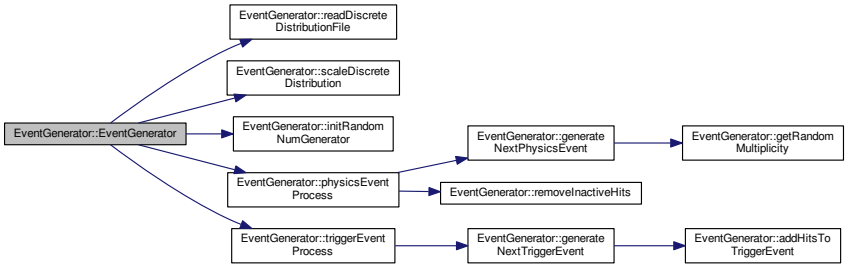
7.7.2.1.1.1 EventGenerator::EventGenerator (sc_core::sc_module_name *name*, const QSettings * *settings*, std::string *output_path*)

Constructor for EventGenerator.

Parameters

in	*name*	SystemC module name
in	*settings*	QSettings object with simulation settings.
in	*output_path*	Directory path to store simulation output data in

Here is the call graph for this function:



7.7.2.1.2 Member Function Documentation

7.7.2.1.2.1 void EventGenerator::addHitsToTriggerEvent (TriggerEvent & *e*) [private]

Iterate through the hit queue corresponding to the chip_id associated with the event referenced by e, and add the active hits to it.

Parameters

in,out	*e*	Event to add hits to.

Todo Is this check worth it performance wise, or is it better to just iterate through the whole list?

Referenced by generateNextTriggerEvent().

Here is the caller graph for this function:

7.7.2.1.2.2 int64_t EventGenerator::generateNextPhysicsEvent (void) [private]

Generate the next physics event (in the future). 1) Generate time till the next physics event 2) Generate hits for the next event, and put them on the hit queue 3) Update counters etc.

Returns

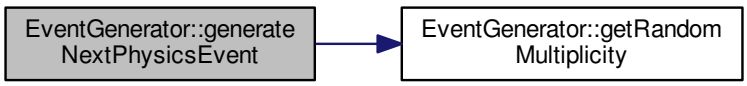
The number of clock cycles until this event will actually occur

Todo Account larger/bigger clusters here (when implemented)

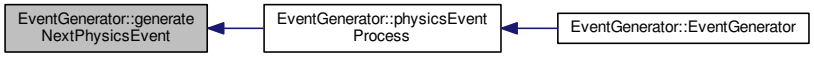
Todo Remove?

Referenced by physicsEventProcess().

Here is the call graph for this function:



Here is the caller graph for this function:



7.7.2.1.2.3 TriggerEvent * EventGenerator::generateNextTriggerEvent (int64_t *event_start*, int64_t *event_end*, int *chip_id*) [private]

Create a new trigger event at the given start time. It checks if trigger event should be filtered or not, and updates trigger ID count.

Parameters

in	*event_start*	Start time of trigger event (time when strobe signal went high).
in	*event_end*	End time of trigger event (time when strobe signal went low again).
in	*chip_id*	Chip ID to generate event for

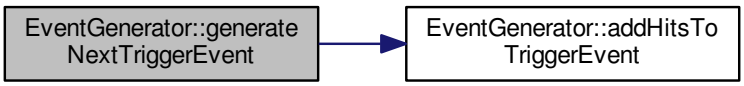
Returns

Pointer to new TriggerEvent object that was allocated on the stack. Caller must remember to delete it when done in order to free memory.

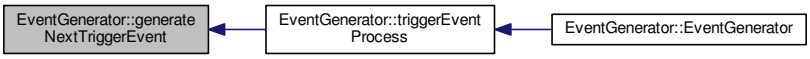
Todo Should I check distance between start time of two triggers? Or the distance in time between the end of the first trigger and the start of the next trigger?

Referenced by triggerEventProcess().

Here is the call graph for this function:



Here is the caller graph for this function:



7.7.2.1.2.4 const TriggerEvent & EventGenerator::getNextTriggerEvent (void)

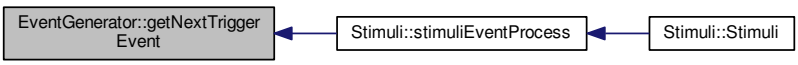
Get a reference to the next event (if there is one). Note: this function will keep returning the same event until it has been removed by removeOldestEvent().

Returns

Reference to next event. If there are no events, then a reference to NoTriggerEvent (with event id = -1) is returned.

Referenced by Stimuli::stimuliEventProcess().

Here is the caller graph for this function:



7.7.2.1.2.5 unsigned int EventGenerator::getRandomMultiplicity (void) [private]

Return a random number of hits (multiplicity) based on the chosen distribution for multiplicity.

Returns

Number of hits

Exceptions

runtime_error	if the EventGenerator for some reason does not have a multiplicity distribution initialized.

Referenced by generateNextPhysicsEvent().

Here is the caller graph for this function:

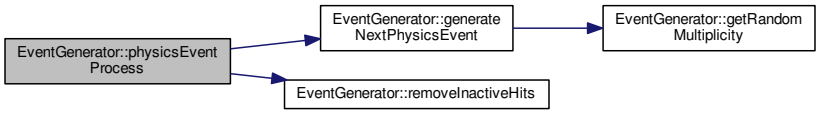
7.7.2.1.2.6 void EventGenerator::physicsEventProcess (void)

SystemC controled method, should be sensitive to the positive edge of the clock. Responsible for 1) Creating new physics events (hits) 2) Deleting old inactive hits.

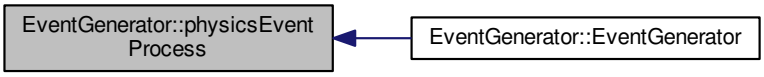
Todo Maybe do this only on strobe falling edge? Saves some CPU cycles that way?

Referenced by EventGenerator().

Here is the call graph for this function:



Here is the caller graph for this function:



7.7.2.1.2.7 void EventGenerator::readDiscreteDistributionFile (const char * *filename,* std::vector< double > & *dist_vector*) const [private]

Read a discrete distribution from file and store it in a vector. The file format is a simple text file, with the following format: X0 Y0 X1 Y1 ... Xn Yn.

Where X-values correspond to the possible range of values for the random distribution, and the Y-values correspond to probability for a given X-value. X and Y is separated by whitespace. All X-values must be unsigned integers, and Y-values are assumed to be (positive) floating point.

The boost::random::discrete_distribution expects a list of probability values, where the index in the list corresponds to the X-value. This function generates a vector to represent that list. Missing X-values is allowed in the file, for example: 0 0.12 1 0.23 3 0.45

In the above example, an entry for the X-value of 2 with probability (Y) 0.0 will be inserted to the vector by this function.

Parameters

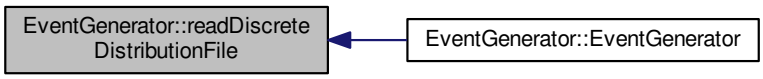
in	*filename*	Relative or absolute path and filename to open
out	*dist_vector*	Reference to vector to store the distribution in

Exceptions

runtime_error	If the file can not be opened
domain_error	If a negative x-value (hits) or y-value (probability) is encountered in the file

Referenced by EventGenerator().

Here is the caller graph for this function:



7.7.2.1.2.8 void EventGenerator::scaleDiscreteDistribution (std::vector< double > & *dist_vector,* double *new_mean_value*) [private]

Scale the x axis of a discrete distribution, so that the distribution gets a new mean value.

Parameters

in,out	*dist_vector*	Distribution to scale. The original distribution in this vector will be overwritten and replaced with the new, scaled, distribution.
in	*new_mean_value*	The desired mean value of the new distribution.

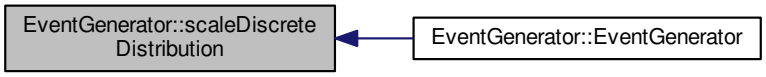
Exceptions

runtime_error	If dist_vector is empty, a runtime_error is thrown.

Todo This changes the mean value slightly.. and the sum isn't that far off 1.0 before this anyway...

Referenced by EventGenerator().

Here is the caller graph for this function:

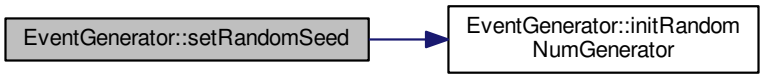


7.7.2.1.2.9 void EventGenerator::setRandomSeed (int *seed*)

Sets the random seed used by random number generators.

Todo More than one seed? What if seed is set after random number generators have been started?

Here is the call graph for this function:



7.7.2.1.3 Member Data Documentation

7.7.2.1.3.1 std::string EventGenerator::mDataPath = "data" [private]

Todo This is currently used.. remove or update code that uses it..

Referenced by removeOldestEvent().

7.7.2.1.3.2 std::vector<std::queue<TriggerEvent*> > EventGenerator::mEventQueue [private]

This is the trigger event queue (ie. the hits that occur between a strobe, which are fed to the Alpide chips). Each Alpide chip has its own queue (corresponding to an index in the vector).

Referenced by EventGenerator(), getNextTriggerEvent(), removeOldestEvent(), and triggerEventProcess().

7.7.2.1.3.3 std::vector<std::deque<Hit> > EventGenerator::mHitQueue [private]

New hits will be push at the back, and old (expired) hits popped at the front. We need to be able to iterate over the queue, so a normal std::queue would not work. And deque seems faster than a list for our purpose: http://stackoverflow.com/questions/14574831/stddeque-or-stdlist But that should probably be tested :) Each Alpide chip has its own queue (corresponding to an index in the vector).

Referenced by addHitsToTriggerEvent(), EventGenerator(), generateNextPhysicsEvent(), and removeInactiveHits().

7.7.2.1.3.4 int64_t EventGenerator::mLastTriggerEventStartTimeNs = 0 [private]

Time of the last trigger event that was generated (time of last strobe) Will not be updated if trigger was filtered out.

Referenced by generateNextTriggerEvent(), and triggerEventProcess().

7.7.2.1.3.5 int EventGenerator::mNextTriggerEventChipId = 0 [private]

Used by getNextTriggerEvent() so it doesn't have to start iterating from the beginning of the event queue vector each time it is called. Also used by removeOldestEvent().

Referenced by getNextTriggerEvent(), removeOldestEvent(), and triggerEventProcess().

7.7.2.1.3.6 int64_t EventGenerator::mNextTriggerEventStartTimeNs = 0 [private]

Start time of next trigger event (start time recorded on STROBE rising edge). Event actually created and hits assigned to it on STROBE falling edge.

Referenced by triggerEventProcess().

7.7.2.1.3.7 boost::random::discrete_distribution* EventGenerator::mRandHitMultiplicityDiscrete [private]

Choice of discrete distribution (based on discrete list of N_hits vs Probability), or gaussian distribution.

Referenced by EventGenerator(), and getRandomMultiplicity().

7.7.2.1.3.8 int EventGenerator::mTriggerFilterTimeNs [private]

Minimum time between two triggers/events. Triggers/events that come sooner than this will be filtered out (but their hits will still be stored).

Referenced by EventGenerator(), and generateNextTriggerEvent().

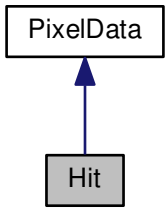
7.7.2.1.3.9 sc_out<bool> EventGenerator::s_physics_event_out

Active for one clock pulse every time we have a "physics event". Not really used for anything, just to indicate physics events in waveforms

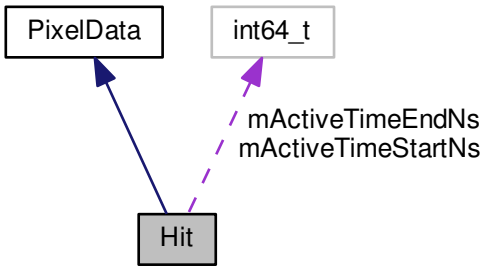
Referenced by physicsEventProcess(), and Stimuli::Stimuli().

7.7.2.2 class Hit

Inheritance diagram for Hit:



Collaboration diagram for Hit:



Public Member Functions

- Hit (int col, int row, int64_t time_now_ns, int dead_time_ns, int active_time_ns)

  Constructor that calculates active start and end times based on current simulation time, and dead times and active times.
- Hit (int col, int row, int64_t time_active_start_ns, int64_t time_active_end_ns)

  Constructor that takes active start and end times directly.
- **Hit** (const Hit &h)
- bool **operator==** (const Hit &rhs) const
- bool **operator>** (const Hit &rhs) const
- bool **operator<** (const Hit &rhs) const
- bool **operator>=** (const Hit &rhs) const
- bool **operator<=** (const Hit &rhs) const
- Hit & **operator=** (const Hit &rhs)
- int64_t **getActiveTimeStart** (void) const
- int64_t **getActiveTimeEnd** (void) const
- bool isActive (int64_t time_now_ns) const

Check if this hit is currently active (which is equivalent to when analog pulse shape is over threshold).

- bool isActive (int64_t strobe_start_time_ns, int64_t strobe_end_time_ns) const

  Check if this hit is active at any time during the specified time duration (between strobe_start_time_ns and strobe_↩ end_time_ns).

Private Attributes

- int64_t **mActiveTimeStartNs**
- int64_t **mActiveTimeEndNs**

7.7.2.2.1 Constructor & Destructor Documentation

7.7.2.2.1.1 Hit::Hit (int *col,* int *row,* int64_t *time_now_ns,* int *dead_time_ns,* int *active_time_ns*)

Constructor that calculates active start and end times based on current simulation time, and dead times and active times.

Parameters

in	*col*	Column number.
in	*row*	Row number.
in	*time_now_ns*	Time (in nanoseconds) when this hit occured (ie. current simulation time).
in	*dead_time_ns*	Dead time (in nanoseconds) before the hit "becomes active". This is equivalent to the time it takes for the analog signal to go above the threshold after a hit.
in	*active_time_ns*	Specifies (in nanoseconds) how long the hit stays active (ie. pixel is triggered) after the dead time has passed. This is equivalent to the amount time the analog pulse into the discriminator/comparator is over threshold.

7.7.2.2.1.2 Hit::Hit (int *col,* int *row,* int64_t *time_active_start_ns,* int64_t *time_active_end_ns*)

Constructor that takes active start and end times directly.

Parameters

in	*col*	Column number.
in	*row*	Row number.
in	*time_active_start_ns*	Absolute simulation time (in nanoseconds) for when the hit becomes active, which is equivalent to the analog signal going above the threshold after a hit.
in	*time_active_end_ns*	Absolute simulation time (in nanoseconds) for when the hit stops being active, which is equivalent to when the analog signal goes below the threshold again after having been active.

7.7.2.2.2 Member Function Documentation

7.7.2.2.2.1 bool Hit::isActive (int64_t *time_now_ns*) const [inline]

Check if this hit is currently active (which is equivalent to when analog pulse shape is over threshold).

Parameters

time_now_ns	Current simulation time (in nanoseconds).

Returns

True if active, false if not.

7.7.2.2.2 bool Hit::isActive (int64_t *strobe_start_time_ns*, int64_t *strobe_end_time_ns*) const [inline]

Check if this hit is active at any time during the specified time duration (between strobe_start_time_ns and strobe↩ _end_time_ns).

Parameters

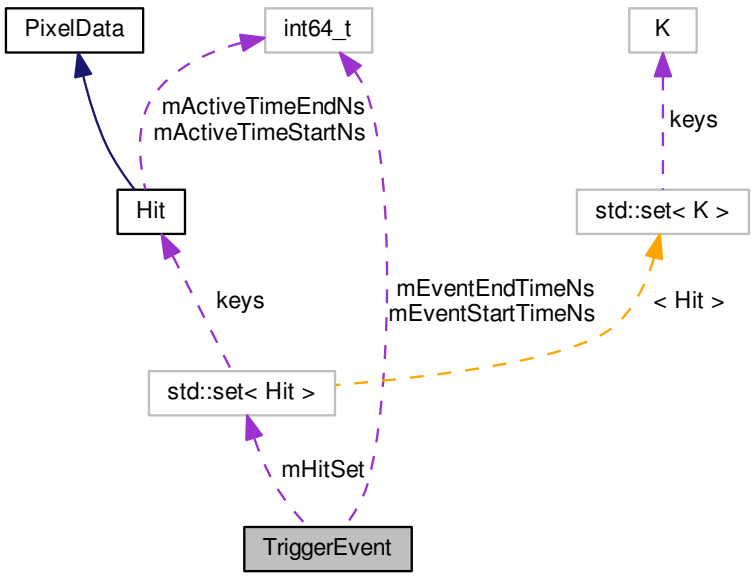
strobe_start_time_ns	Strobe start time
strobe_end_time_ns	Strobe end time

Returns

True if active, false if not.

7.7.2.3 class TriggerEvent

Collaboration diagram for TriggerEvent:



Public Member Functions

- TriggerEvent (int64_t event_start_time_ns, int64_t event_end_time_ns, int chip_id, int event_id, bool filter_↩ event=false)

  Standard constructor.
- TriggerEvent (const TriggerEvent &e)

  Copy constructor.
- void **addHit** (const Hit &h)
- void feedHitsToChip (PixelMatrix &matrix) const

  Feed this event to the pixel matrix of the specified chip. If the trigger filter flag is set, or if there are no hits in the event, nothing will be sent to the chip, and a new event/MEB will not be created in the Alpide chip / pixel matrix object.
- void writeToFile (const std::string path="")

  Write this event to file, in XML format. The filename will be: "path/event<mEventId>.xml".
- void **setEventFilteredFlag** (bool value)
- int **getEventSize** (void) const
- int **getChipId** (void) const
- int **getEventId** (void) const
- int64_t **getEventStartTime** (void) const
- int64_t **getEventEndTime** (void) const
- bool **getEventFilteredFlag** (void) const

Private Attributes

- int64_t mEventStartTimeNs

  Absolute start time of event.
- int64_t mEventEndTimeNs

  Absolute end time of event.
- int **mEventId**
- int **mChipId**
- std::set< Hit > **mHitSet**
- bool mEventFilteredFlag

  This flag indicates that this event/trigger came too soon, and that it has been filtered out. The class object is still created to keep track of the pixels that are hit, but they will not be fed to the Alpide chip.

7.7.2.3.1 Constructor & Destructor Documentation

7.7.2.3.1.1 TriggerEvent::TriggerEvent (int64_t *event_start_time_ns*, int64_t *event_end_time_ns*, int *chip_id*, int *event_id*, bool *filter_event* = `false`)

Standard constructor.

Parameters

in	*event_start_time_ns*	Start time of trigger event (time when strobe was asserted)
in	*event_end_time_ns*	End time of trigger event (time when strobe was deasserted)
in	*chip_id*	Chip ID
in	*event_id*	Event ID
in	*filter_event*	Flag that indicates whether this trigger should be filtered or not (when trigger filtering is enabled, and trigger came too close to last event)

7.7.2.3.2 Member Function Documentation

7.7.2.3.2.1 void TriggerEvent::feedHitsToChip (PixelMatrix & *matrix*) const

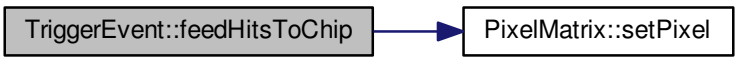
Feed this event to the pixel matrix of the specified chip. If the trigger filter flag is set, or if there are no hits in the event, nothing will be sent to the chip, and a new event/MEB will not be created in the Alpide chip / pixel matrix object.

Parameters

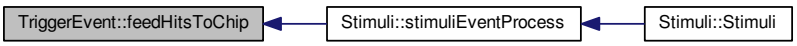
out	*matrix*	Pixel matrix for the chip

Referenced by Stimuli::stimuliEventProcess().

Here is the call graph for this function:



Here is the caller graph for this function:



7.7.2.3.2.2 void TriggerEvent::writeToFile (const std::string *path* = " ")

Write this event to file, in XML format. The filename will be: "path/event<mEventId>.xml".

Todo Note in use.. Revisit this function, since I have changed this class a lot...

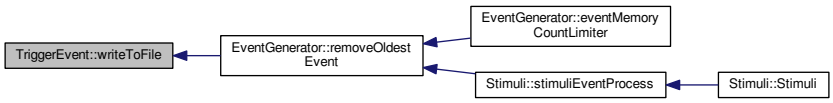
Parameters

in	*path*	Path to store file in.

Todo Implement layers etc.

Referenced by EventGenerator::removeOldestEvent().

Here is the caller graph for this function:



7.7.2.3.3 Member Data Documentation

7.7.2.3.3.1 bool TriggerEvent::mEventFilteredFlag [private]

This flag indicates that this event/trigger came too soon, and that it has been filtered out. The class object is still created to keep track of the pixels that are hit, but they will not be fed to the Alpide chip.

Todo With the new way of doing things, I don't need to have an Event object to keep track of hits, they are stored in the EventGenerator object. So I can get rid off this?

Referenced by feedHitsToChip(), and TriggerEvent().

7.8 Miscellaneous functions

Functions

- template<class T >
  static void addTrace (sc_trace_file *wf, std::string name_prefix, std::string signal_name, T &signal)

  Add a SystemC signal/trace to VCD file, with desired signal hierarchy given by name_prefix.

7.8.1 Detailed Description

7.8.2 Function Documentation

7.8.2.1 template<class T > static void addTrace (sc_trace_file * *wf*, std::string *name_prefix*, std::string *signal_name*, T & *signal*) [inline], [static]

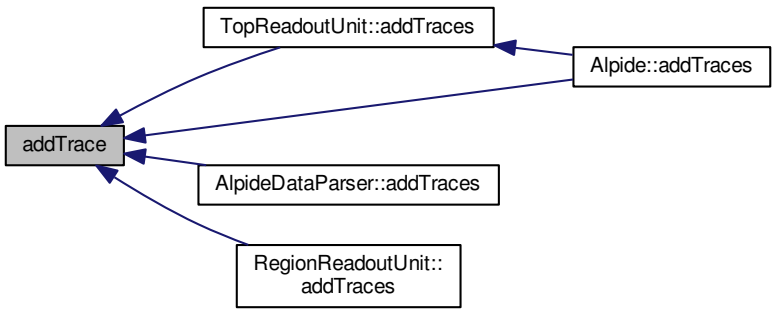
Add a SystemC signal/trace to VCD file, with desired signal hierarchy given by name_prefix.

Parameters

wf	VCD waveform file pointer
name_prefix	Prefix to be added before signal name, used for signal hierarchy. A period (.) separates levels of hierarchy.
signal_name	Name of the signal
signal	The SystemC signal object

Referenced by TopReadoutUnit::addTraces(), AlpideDataParser::addTraces(), Alpide::addTraces(), and RegionReadoutUnit::addTraces().

Here is the caller graph for this function:



7.9 Settings

Macros

- #define **DEFAULT_DATA_OUTPUT_WRITE_VCD** "true"
- #define **DEFAULT_DATA_OUTPUT_WRITE_VCD_CLOCK** "false"
- #define **DEFAULT_DATA_OUTPUT_WRITE_EVENT_CSV** "true"
- #define **DEFAULT_SIMULATION_N_CHIPS** "25000"
- #define **DEFAULT_SIMULATION_N_EVENTS** "10000"
- #define **DEFAULT_SIMULATION_CONTINUOUS_MODE** "false"
- #define **DEFAULT_SIMULATION_RANDOM_SEED** "0"
- #define **DEFAULT_EVENT_HIT_MULTIPLICITY_DISTRIBUTION_TYPE** "discrete"
- #define **DEFAULT_EVENT_HIT_MULTIPLICITY_DISTRIBUTION_FILE** "multipl_dist_raw_bins.txt"
- #define **DEFAULT_EVENT_HIT_MULTIPLICITY_GAUSS_AVG** "2000"
- #define **DEFAULT_EVENT_HIT_MULTIPLICITY_GAUSS_STDDEV** "350"
- #define **DEFAULT_EVENT_HIT_DENSITY_MIN_BIAS_PER_CM2** "19"
- #define **DEFAULT_EVENT_BUNCH_CROSSING_RATE_NS** "25"
- #define **DEFAULT_EVENT_AVERAGE_EVENT_RATE_NS** "2500"
- #define **DEFAULT_EVENT_TRIGGER_DELAY_NS** "1000"
- #define **DEFAULT_EVENT_TRIGGER_FILTER_TIME_NS** "10000"
- #define **DEFAULT_EVENT_TRIGGER_FILTER_ENABLE** "true"
- #define **DEFAULT_EVENT_STROBE_ACTIVE_LENGTH_NS** "4800"
- #define **DEFAULT_EVENT_STROBE_INACTIVE_LENGTH_NS** "200"
- #define **DEFAULT_ALPIDE_CLUSTERING_ENABLE** "true"
- #define **DEFAULT_ALPIDE_REGION_FIFO_SIZE** "128"
- #define **DEFAULT_ALPIDE_DMU_FIFO_SIZE** "64"
- #define **DEFAULT_ALPIDE_REGION_SIZE** "32"
- #define **DEFAULT_ALPIDE_PIXEL_SHAPING_DEAD_TIME_NS** "200"
- #define **DEFAULT_ALPIDE_PIXEL_SHAPING_ACTIVE_TIME_NS** "6000"
- #define **DEFAULT_ALPIDE_MATRIX_READOUT_SPEED_FAST** "true"

Functions

- QSettings ∗ getSimSettings (const char ∗fileName="settings.txt")

  Open a file with simulation settings. If the file does not exist, it will be created. If any settings are missing, they will be initialized with default values. If no filename is specified, the default settings.txt file is used in the current directory.
- void setDefaultSimSettings (QSettings ∗readoutSimSettings)

  Set default settings for each setting that is missing in the QSettings object.

7.9.1 Detailed Description

7.9.2 Function Documentation

7.9.2.1 QSettings∗ getSimSettings (const char ∗ *fileName*)

Open a file with simulation settings. If the file does not exist, it will be created. If any settings are missing, they will be initialized with default values. If no filename is specified, the default settings.txt file is used in the current directory.

Parameters

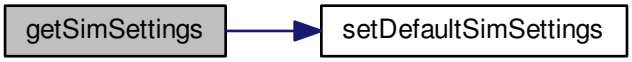
in	*fileName*	File to open, relative to current directory. Defaults to settings.txt if not supplied.

Returns

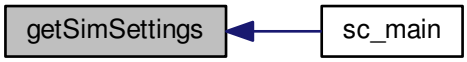
Pointer to QSettings object initialized with all settings, either from settings file or with default settings if any settings were missing.

Referenced by sc_main().

Here is the call graph for this function:



Here is the caller graph for this function:



7.9.2.2 void setDefaultSimSettings (QSettings * *readoutSimSettings*)

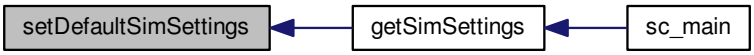
Set default settings for each setting that is missing in the QSettings object.

Parameters

in,out	*readoutSimSettings*	Pointer to QSettings object.

Referenced by getSimSettings().

Here is the caller graph for this function:



7.10 Main Alpide Simulation Testbench

Classes

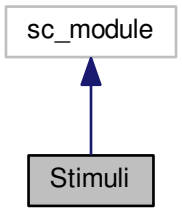
- class Stimuli

7.10.1 Detailed Description

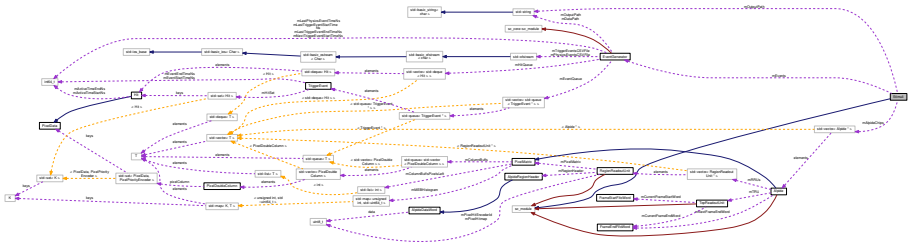
7.10.2 Class Documentation

7.10.2.1 class Stimuli

Inheritance diagram for Stimuli:



Collaboration diagram for Stimuli:



Public Member Functions

- Stimuli (sc_core::sc_module_name name, QSettings ∗settings, std::string output_path)

  Constructor for stimuli class. Instantiates and initializes the EventGenerator and Alpide objects, connects the SystemC ports.
- void stimuliMainProcess (void)

  Main control of simulation stimuli, which mainly involves controlling the strobe signal and stop the simulation after the desired number of events.
- void stimuliEventProcess (void)

  SystemC controlled method. Waits for EventGenerator to notify the E_trigger_event_available notification queue that a new trigger event is available. When a trigger event is available it is fed to the Alpide chip(s).
- void addTraces (sc_trace_file ∗wf) const

  Add SystemC signals to log in VCD trace file.
- void writeDataToFile (void) const

  Write simulation data to file. Histograms for MEB usage from the Alpide chips, and trigger event statistics (number of accepted/rejected) in the chips are recorded here.

Public Attributes

- sc_in_clk **clock**
- sc_signal< bool > **s_strobe_n**
- sc_signal< bool > **s_physics_event**
- sc_signal< bool > **s_chip_ready** [100]
- sc_signal< sc_uint< 24 > > **s_alpide_serial_data** [100]
- sc_event_queue **E_trigger_event_available**

Private Attributes

- EventGenerator * **mEvents**
- std::vector< Alpide * > **mAlpideChips**
- const QSettings * **mSettings**
- std::string **mOutputPath**
- bool **simulation_done** = false
- bool **mContinuousMode**
- int mNumEvents
- int **mNumChips**
- int **mStrobeActiveNs**
- int **mStrobeInactiveNs**
- int **mTriggerDelayNs**

7.10.2.1.1 Constructor & Destructor Documentation

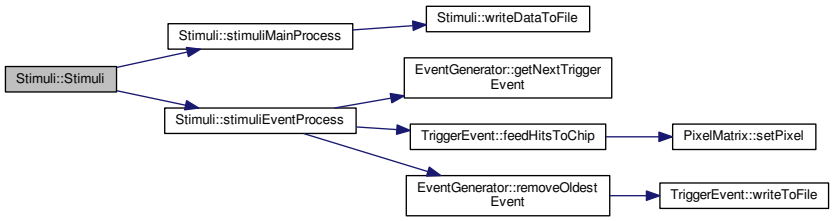
7.10.2.1.1.1 Stimuli::Stimuli (sc_core::sc_module_name *name,* QSettings * *settings,* std::string *output_path*)

Constructor for stimuli class. Instantiates and initializes the EventGenerator and Alpide objects, connects the SystemC ports.

Parameters

in	*name*	SystemC module name
in	*settings*	QSettings object with simulation settings.
in	*output_path*	Path to store output files generated by the Stimuli class

Here is the call graph for this function:



7.10.2.1.2 Member Function Documentation

7.10.2.1.2.1 void Stimuli::addTraces (sc_trace_file * *wf*) const

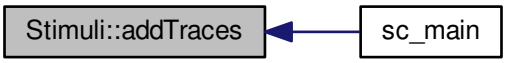
Add SystemC signals to log in VCD trace file.

Parameters

in,out	*wf*	VCD waveform file pointer

Referenced by sc_main().

Here is the caller graph for this function:



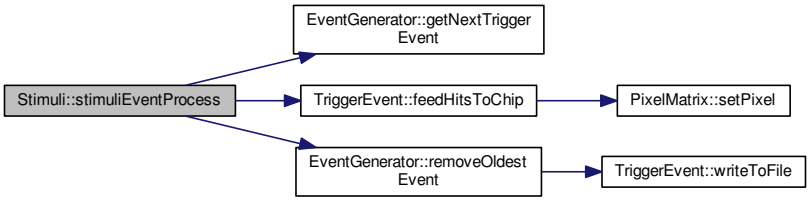
7.10.2.1.2.2 void Stimuli::stimuliEventProcess (void)

SystemC controlled method. Waits for EventGenerator to notify the E_trigger_event_available notification queue that a new trigger event is available. When a trigger event is available it is fed to the Alpide chip(s).

Todo Check if there are actually events? Throw an error if we get notification but there are not events?

Referenced by Stimuli().

Here is the call graph for this function:



Here is the caller graph for this function:

7.10.2.1.3 Member Data Documentation

7.10.2.1.3.1 int Stimuli::mNumEvents [private]

Todo Make it a 64-bit int?

Referenced by Stimuli(), stimuliMainProcess(), and writeDataToFile().

Chapter 8

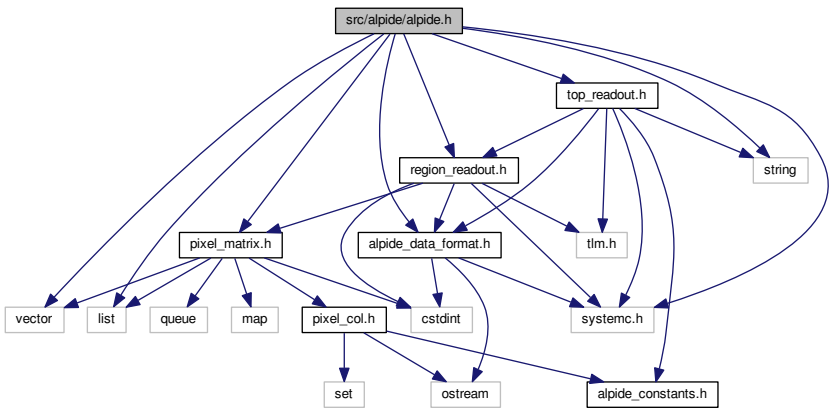
File Documentation

8.1 src/alpide/alpide.h File Reference

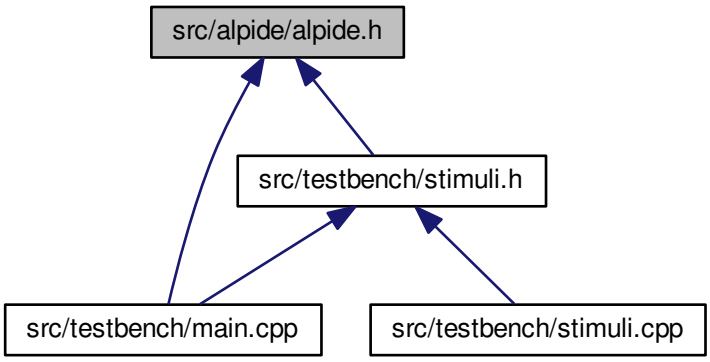
Source file for Alpide class.

```
#include "alpide_data_format.h"
#include "pixel_matrix.h"
#include "region_readout.h"
#include "top_readout.h"
#include <systemc.h>
#include <vector>
#include <list>
#include <string>
```

Include dependency graph for alpide.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Alpide

8.1.1 Detailed Description

Source file for Alpide class.

Header file for Alpide class.

Author

Simon Voigt Nesbo

Date

December 12, 2016

Author

Simon Voigt Nesbo

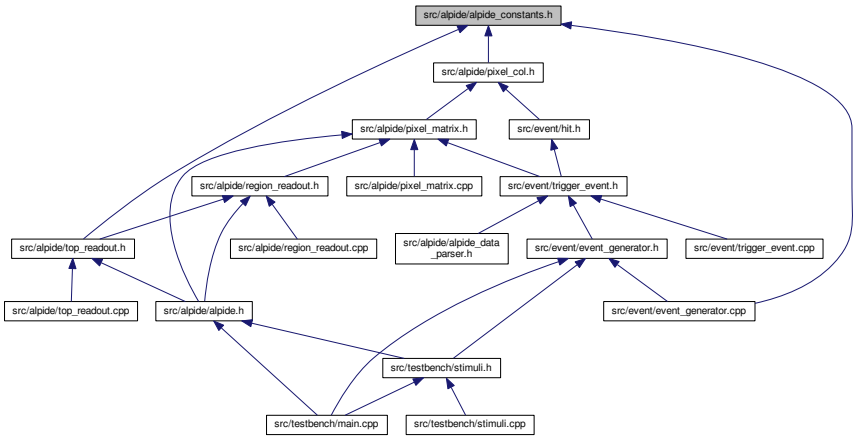
Date

December 11, 2016

8.2 src/alpide/alpide_constants.h File Reference

Various constants for alpide chip, such as pixel matrix width and heigh, fifo depths, etc.

This graph shows which files directly or indirectly include this file:



Macros

- #define **N_REGIONS** 32
- #define **N_PIXEL_ROWS** 512
- #define **N_PIXEL_COLS** 1024
- #define **N_PIXEL_COLS_PER_REGION** (N_PIXEL_COLS/N_REGIONS)
- #define **N_PIXEL_DOUBLE_COLS_PER_REGION** (N_PIXEL_COLS_PER_REGION/2)
- #define **N_PIXELS_PER_REGION** (N_PIXEL_COLS/N_REGIONS)
- #define **TRU_FRAME_FIFO_ALMOST_FULL1** 48
- #define **TRU_FRAME_FIFO_ALMOST_FULL2** 56
- #define **TRU_FRAME_FIFO_SIZE** 64
- #define **DATA_LONG_PIXMAP_SIZE** ((unsigned int) 7)
- #define **LHC_ORBIT_BUNCH_COUNT** 3564
- #define **CHIP_WIDTH_CM** 3
- #define **CHIP_HEIGHT_CM** 1.5

8.2.1 Detailed Description

Various constants for alpide chip, such as pixel matrix width and heigh, fifo depths, etc.

Author

Simon Voigt Nesbo

Date

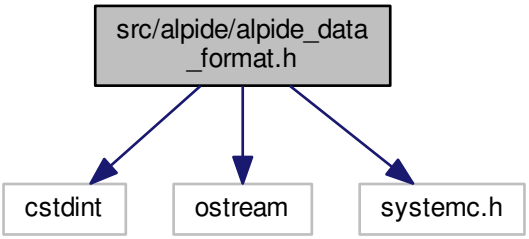
November 27, 2016

8.3 src/alpide/alpide_data_format.h File Reference

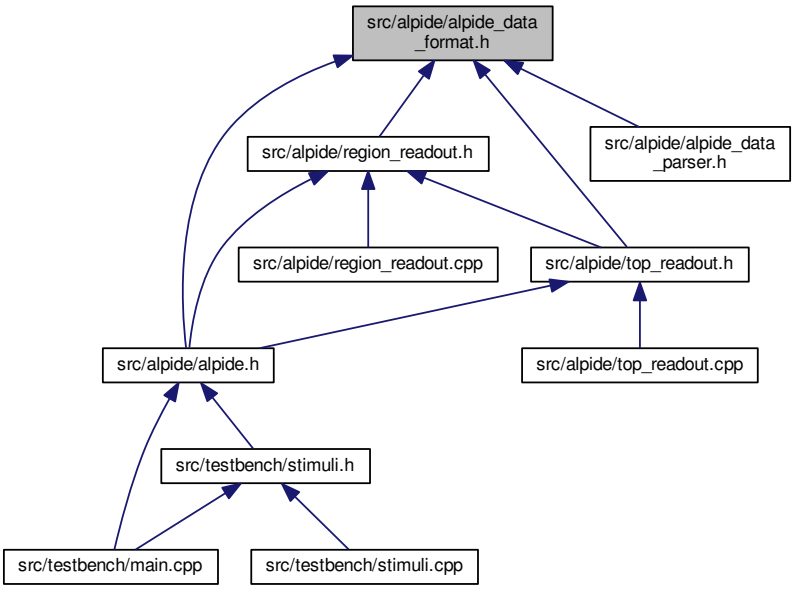
Definitions for data format used in Alpide chip.

```
#include <cstdint>
#include <ostream>
#include <systemc.h>
```

Include dependency graph for alpide_data_format.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct FrameStartFifoWord

Data word stored in FRAME START FIFO. More...

- struct FrameEndFifoWord

  Data word stored in FRAME END FIFO. More...

- class AlpideDataWord

  The FIFOs in the Alpide chip are 24 bits, or 3 bytes, wide. This is a base class for the data words that holds 3 bytes, and is used as the data type in the SystemC FIFO templates. This class shouldn't be used on its own, the various types of data words are implemented in derived classes. More...

- class AlpideIdle
- class AlpideChipHeader
- class AlpideChipTrailer
- class AlpideChipEmptyFrame
- class AlpideRegionHeader
- class AlpideRegionTrailer
- class AlpideDataShort
- class AlpideDataLong
- class AlpideBusyOn
- class AlpideBusyOff
- class AlpideComma

Variables

- const uint8_t DW_IDLE = 0b11111111
- const uint8_t **DW_CHIP_HEADER** = 0b10100000
- const uint8_t **DW_CHIP_TRAILER** = 0b10110000
- const uint8_t **DW_CHIP_EMPTY_FRAME** = 0b11100000
- const uint8_t **DW_REGION_HEADER** = 0b11000000
- const uint8_t **DW_REGION_TRAILER** = 0b11110011
- const uint8_t **DW_DATA_SHORT** = 0b01000000
- const uint8_t **DW_DATA_LONG** = 0b00000000
- const uint8_t **DW_BUSY_ON** = 0b11110001
- const uint8_t **DW_BUSY_OFF** = 0b11110000
- const uint8_t DW_COMMA = 0b11111110
- const uint8_t **READOUT_FLAGS_BUSY_VIOLATION** = 0b00001000
- const uint8_t **READOUT_FLAGS_FLUSHED_INCOMPLETE** = 0b00000100
- const uint8_t **READOUT_FLAGS_STROBE_EXTENDED** = 0b00000010
- const uint8_t **READOUT_FLAGS_BUSY_TRANSITION** = 0b00000001
- const uint8_t MASK_IDLE_BUSY_COMMA = 0b11111111

  Mask for busy, idle and comma words.

- const uint8_t MASK_CHIP = 0b11110000

  Mask for chip header/trailer/empty frame words.

- const uint8_t MASK_REGION_HEADER = 0b11100000

  Mask for region header word.

- const uint8_t MASK_DATA = 0b11000000

  Mask for data short/long words.

8.3.1 Detailed Description

Definitions for data format used in Alpide chip.

Author

Simon Voigt Nesbo

Date

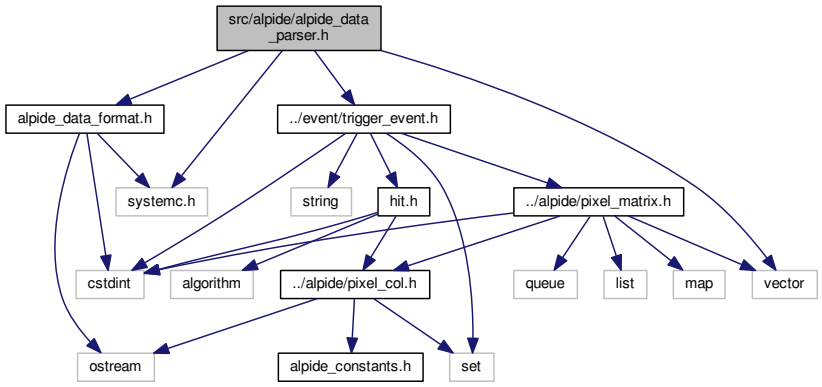
February 20, 2017

8.4 src/alpide/alpide_data_parser.h File Reference

Classes for parsing serial data from Alpide chip, and building/reconstructing events/frames from the data.

```
#include "alpide_data_format.h"
#include "../event/trigger_event.h"
#include <vector>
#include <systemc.h>
```

Include dependency graph for alpide_data_parser.h:



Classes

- struct AlpideDataParsed
- class AlpideEventFrame
- class AlpideEventBuilder
- class AlpideDataParser

Enumerations

- enum AlpideDataTypes {
  ALPIDE_IDLE, **ALPIDE_CHIP_HEADER1**, **ALPIDE_CHIP_HEADER2**, **ALPIDE_CHIP_TRAILER**,
  ALPIDE_CHIP_EMPTY_FRAME1, **ALPIDE_CHIP_EMPTY_FRAME2**, **ALPIDE_REGION_HEADER**, **AL↩**
  PIDE_DATA_SHORT1,
  ALPIDE_DATA_SHORT2, **ALPIDE_DATA_LONG1**, **ALPIDE_DATA_LONG2**, **ALPIDE_DATA_LONG3**,
  ALPIDE_BUSY_ON, **ALPIDE_BUSY_OFF**, **ALPIDE_COMMA**, **ALPIDE_UNKNOWN** }

8.4.1 Detailed Description

Classes for parsing serial data from Alpide chip, and building/reconstructing events/frames from the data.

Author

Simon Voigt Nesbo

Date

March 6, 2017

Todo Move this class to a separate directory/module. Don't mix it with the Alpide simulation model.

Author

Simon Voigt Nesbo

Date

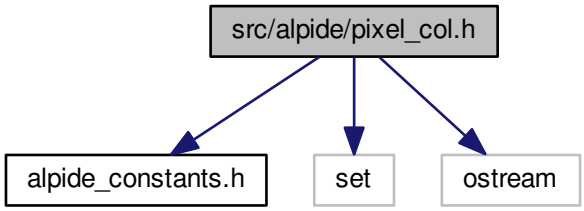
March 3, 2017

8.5 src/alpide/pixel_col.h File Reference

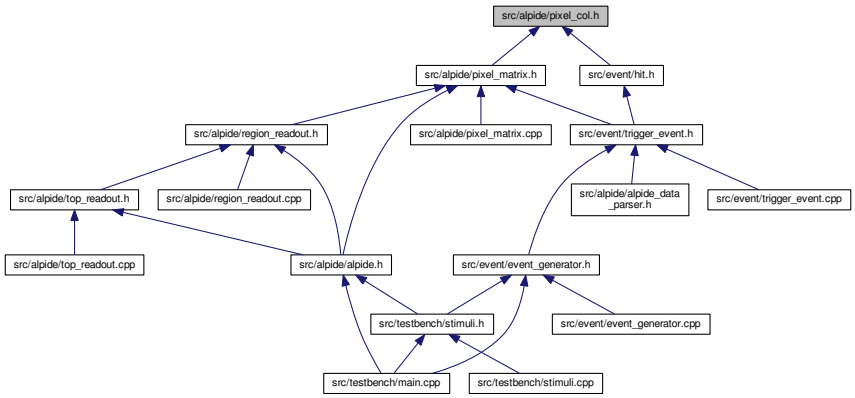
Source file for pixel column, double column, and priority encoder classes.

```
#include "alpide_constants.h"
#include <set>
#include <ostream>
```

Include dependency graph for pixel_col.h:



This graph shows which files directly or indirectly include this file:



Classes

- class PixelData

  A struct that indicates a hit in a region, at the pixel identified by the col and row variables. For each hit an object of this type will be inserted into the std::set container in regionDataVector. For the pixels that don't have hits there will not be an object of this type inserted. Column should be 0 or 1. Row can be any value from 0 to N_PIXEL_ROWS-1. More...

- class PixelPriorityEncoder

  Comparator class/function for use with the PixelData class in the std::set container, which allows the container to sort the PixelData entries in a meaningful way. The picture below is from the ALPIDE operations manual, and shows: More...

- class PixelDoubleColumn

Functions

- const PixelData **NoPixelHit** (-1,-1)

8.5.1 Detailed Description

Source file for pixel column, double column, and priority encoder classes.

Pixel column, double column, and priority encoder classes.

Author

Simon Voigt Nesbo

Date

November 27, 2016

Author

Simon Voigt Nesbo

Date

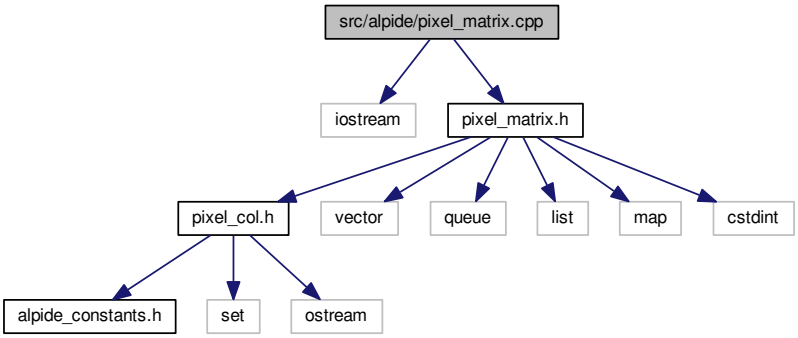
November 27, 2016 Detailed description of file.

8.6 src/alpide/pixel_matrix.cpp File Reference

Source file for pixel matrix class.

```
#include <iostream>
#include "pixel_matrix.h"
```

Include dependency graph for pixel_matrix.cpp:



8.6.1 Detailed Description

Source file for pixel matrix class.

Author

Simon Voigt Nesbo

Date

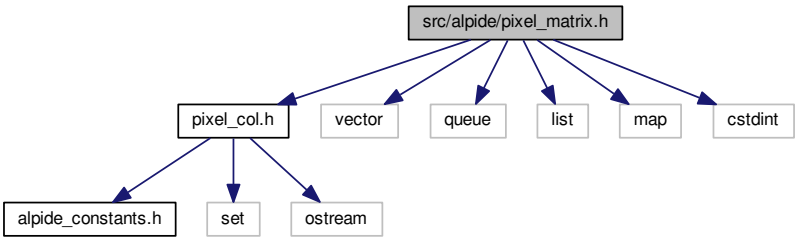
December 11, 2016 Pixel matrix class comprises all the pixel regions, which allows to interface in terms of absolute coordinates with the pixel matrix. Special version for the Alpide Dataflow SystemC model.

8.7 src/alpide/pixel_matrix.h File Reference

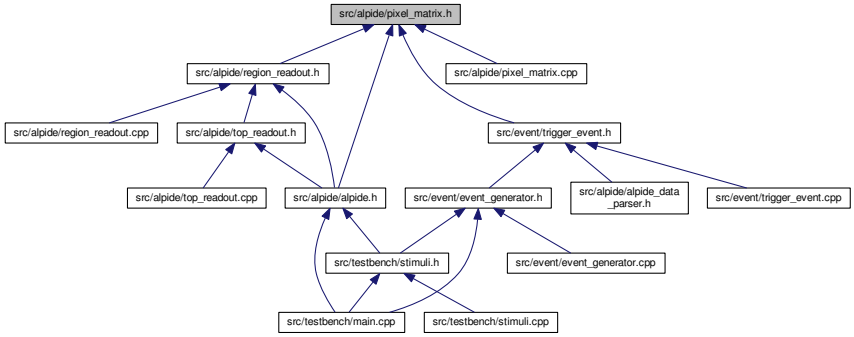
Header file for pixel matrix class.

```
#include "pixel_col.h"
#include <vector>
#include <queue>
#include <list>
#include <map>
#include <cstdint>
```

Include dependency graph for pixel_matrix.h:



This graph shows which files directly or indirectly include this file:



Classes

- class PixelMatrix

8.7.1 Detailed Description

Header file for pixel matrix class.

Author

Simon Voigt Nesbo

Date

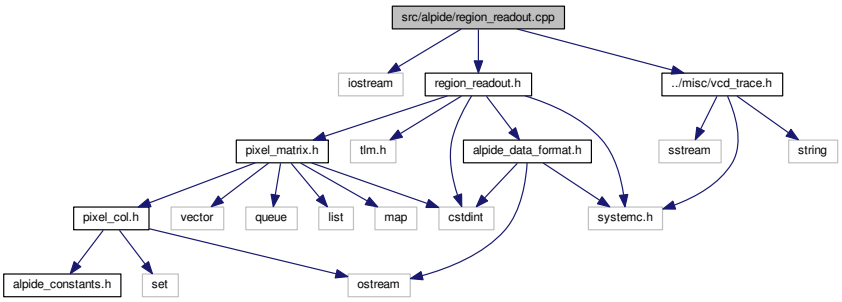
December 11, 2016 Pixel matrix class comprises all the pixel regions, which allows to interface in terms of absolute coordinates with the pixel matrix. Special version for the Alpide Dataflow SystemC model.

8.8 src/alpide/region_readout.cpp File Reference

Class for implementing the Region Readout Unit (RRU) in the Alpide chip.

```
#include <iostream>
#include "region_readout.h"
#include "../misc/vcd_trace.h"
```

Include dependency graph for region_readout.cpp:



Functions

- **SC_HAS_PROCESS** (RegionReadoutUnit)

8.8.1 Detailed Description

Class for implementing the Region Readout Unit (RRU) in the Alpide chip.

Author

Simon Voigt Nesbo

Date

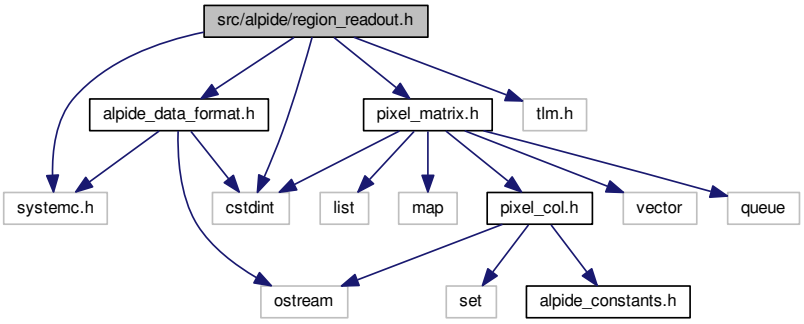
February 20, 2017

8.9 src/alpide/region_readout.h File Reference

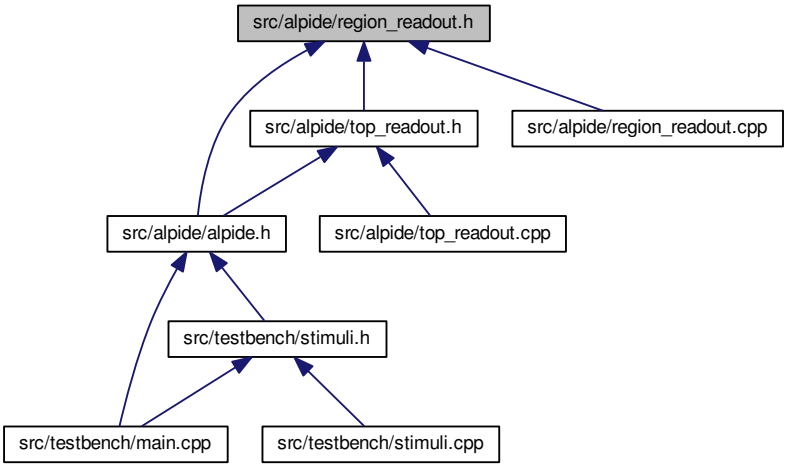
Class for implementing the Region Readout Unit (RRU) in the Alpide chip.

```
#include "alpide_data_format.h"
#include "pixel_matrix.h"
#include <cstdint>
#include <systemc.h>
#include <tlm.h>
```

Include dependency graph for region_readout.h:



This graph shows which files directly or indirectly include this file:



Classes

- class RegionReadoutUnit

Enumerations

- enum { **IDLE** = 0, **START_READOUT** = 1, **READOUT_AND_CLUSTERING** = 2, **REGION_TRAILER** = 3 }
- enum { **IDLE** = 0, **EMPTY** = 1, **VALID** = 2, **POP** = 3 }
- enum { **HEADER** = 0, **DATA** = 1 }

8.9.1 Detailed Description

Class for implementing the Region Readout Unit (RRU) in the Alpide chip.

Author

Simon Voigt Nesbo

Date

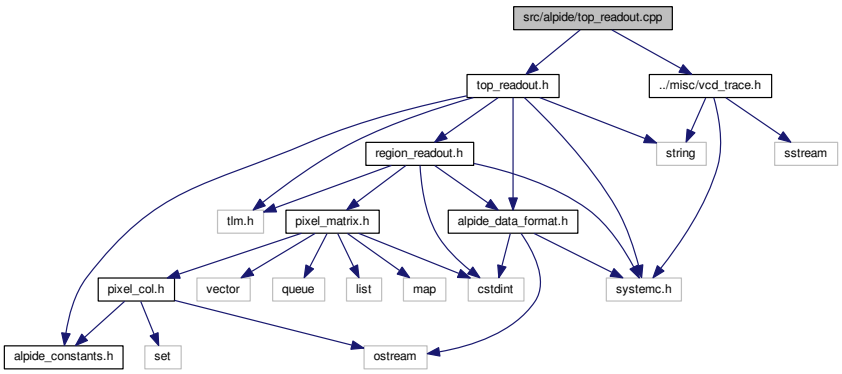
February 20, 2017

8.10 src/alpide/top_readout.cpp File Reference

Class for implementing the Top Readout Unit (TRU) in the Alpide chip.

```
#include "top_readout.h"
#include "../misc/vcd_trace.h"
```

Include dependency graph for top_readout.cpp:



Functions

- **SC_HAS_PROCESS** (TopReadoutUnit)

8.10.1 Detailed Description

Class for implementing the Top Readout Unit (TRU) in the Alpide chip.

Author

Simon Voigt Nesbo

Date

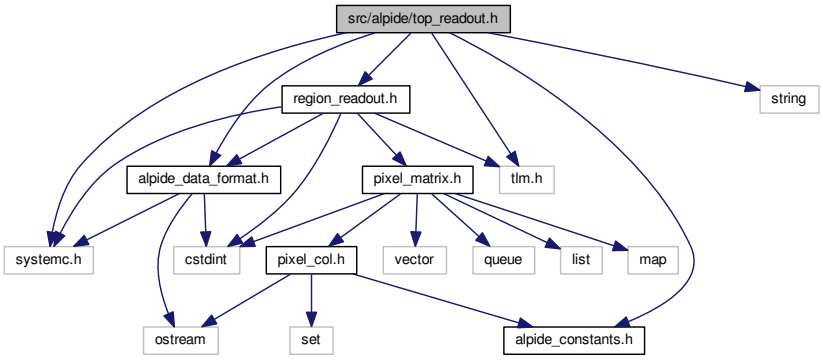
February 20, 2017

8.11 src/alpide/top_readout.h File Reference

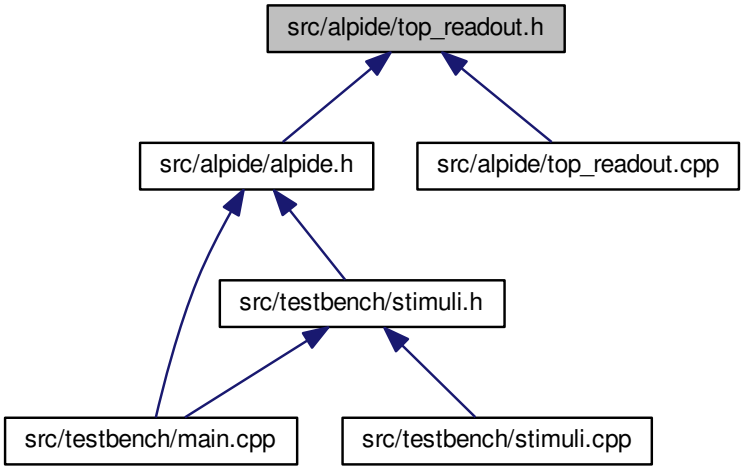
Class for implementing the Top Readout Unit (TRU) in the Alpide chip.

```
#include "region_readout.h"
#include "alpide_constants.h"
#include "alpide_data_format.h"
#include <string>
#include <systemc.h>
#include <tlm.h>
```

Include dependency graph for top_readout.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TopReadoutUnit

8.11.1 Detailed Description

Class for implementing the Top Readout Unit (TRU) in the Alpide chip.

Author

Simon Voigt Nesbo

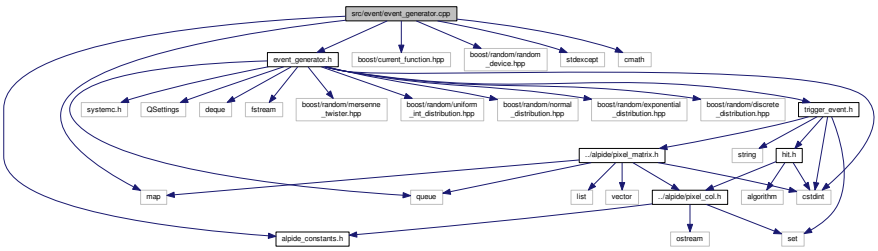
Date

February 20, 2017

8.12 src/event/event_generator.cpp File Reference

```
#include "event_generator.h"
#include "../alpide/alpide_constants.h"
#include <boost/current_function.hpp>
#include <boost/random/random_device.hpp>
#include <stdexcept>
#include <cmath>
#include <map>
```

Include dependency graph for event_generator.cpp:



Macros

- #define **print_function_timestamp**()

Functions

- **SC_HAS_PROCESS** (EventGenerator)

8.12.1 Detailed Description

Author

Simon Voigt Nesbo

Date

December 22, 2016

A simple event generator for Alpide SystemC simulation model.

8.12.2 Macro Definition Documentation

8.12.2.1 #define print_function_timestamp( )

Value:

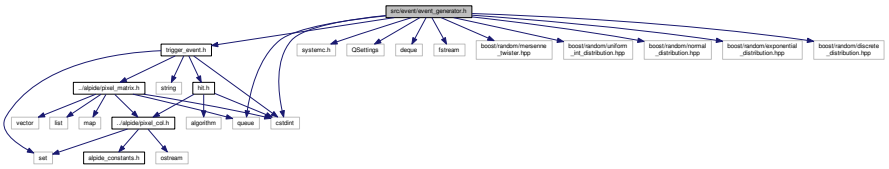
```
std::cout << std::endl << "@ " << sc_time_stamp().value() << " ns\t";  \
  std::cout << BOOST_CURRENT_FUNCTION << ":" << std::endl; \
  std::cout << "-------------------------------------------"; \
  std::cout << "-------------------------------------------" << std::endl;
```

8.13 src/event/event_generator.h File Reference

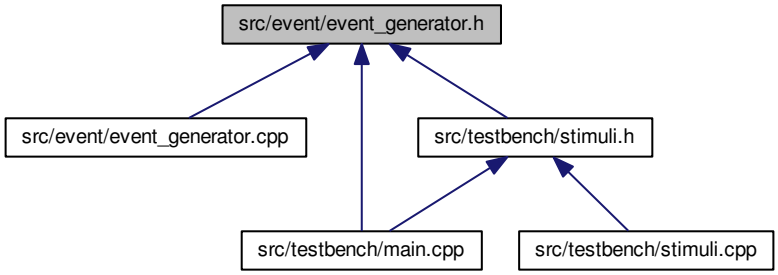
A simple event generator for Alpide SystemC simulation model.

```
#include "trigger_event.h"
#include <systemc.h>
#include <QSettings>
#include <queue>
#include <deque>
#include <fstream>
#include <boost/random/mersenne_twister.hpp>
#include <boost/random/uniform_int_distribution.hpp>
#include <boost/random/normal_distribution.hpp>
#include <boost/random/exponential_distribution.hpp>
#include <boost/random/discrete_distribution.hpp>
#include <cstdint>
```

Include dependency graph for event_generator.h:



This graph shows which files directly or indirectly include this file:



Classes

- class EventGenerator

  A simple event generator for Alpide SystemC simulation model. More...

Macros

- #define **N_CHIPS** 108

8.13.1 Detailed Description

A simple event generator for Alpide SystemC simulation model.

Author

Simon Voigt Nesbo

Date

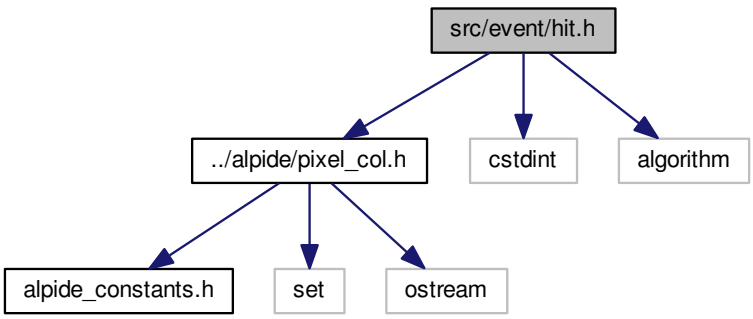
December 22, 2016

8.14 src/event/hit.h File Reference

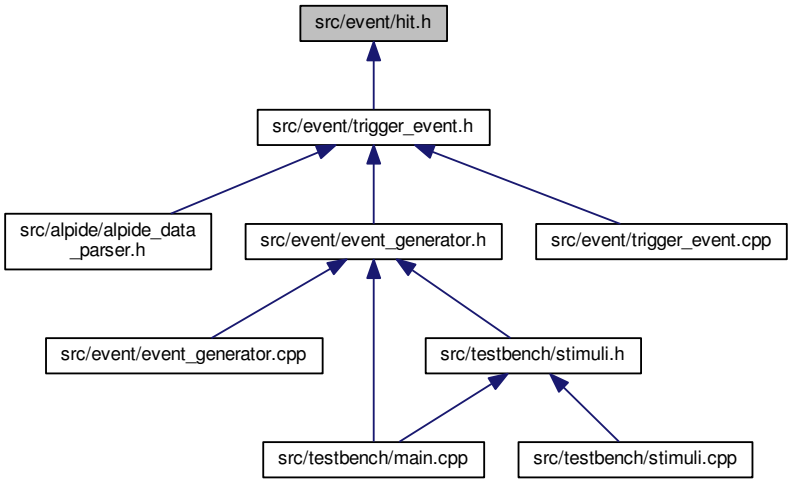
Source file for PixelData and Hit classes. These classes hold the coordinates for a discrete hit in the Alpide chip, along with information about when the hit is active (equivalent to when the analog pulse out of the amplifier and shaping stage in the analog front end goes above the threshold).

```
#include "../alpide/pixel_col.h"
#include <cstdint>
#include <algorithm>
```

Include dependency graph for hit.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Hit

8.14.1 Detailed Description

Source file for PixelData and Hit classes. These classes hold the coordinates for a discrete hit in the Alpide chip, along with information about when the hit is active (equivalent to when the analog pulse out of the amplifier and shaping stage in the analog front end goes above the threshold).

Header file for PixelData and Hit classes. These classes hold the coordinates for a discrete hit in the Alpide chip, along with information about when the hit is active (equivalent to when the analog pulse out of the amplifier and shaping stage in the analog front end goes above the threshold).

Author

Simon Voigt Nesbo

Date

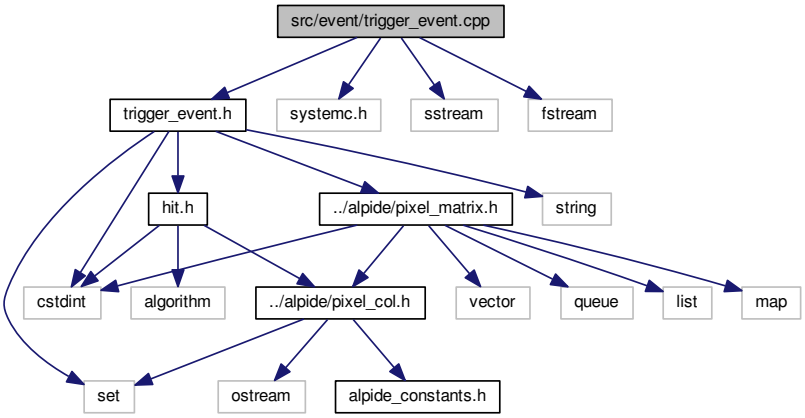
December 12, 2016

8.15 src/event/trigger_event.cpp File Reference

Event class for Alpide SystemC simulation model. This class holds all the pixel hits for an event for the whole detector. The philosophy behind this class is that the shaping etc. is performed by this class and the EventGenerator class, and that the pixel hits here can be fed directly to the Alpide chip at the given time.

```
#include "trigger_event.h"
#include <systemc.h>
#include <sstream>
#include <fstream>
```

Include dependency graph for trigger_event.cpp:



Variables

- const TriggerEvent NoTriggerEvent (0, 0,-1,-1)

  A TriggerEvent that equals NoTriggerEvent is returned by some of the EventGenerator's functions which return a reference to an event, when there is no TriggerEvent to return.

8.15.1 Detailed Description

Event class for Alpide SystemC simulation model. This class holds all the pixel hits for an event for the whole detector. The philosophy behind this class is that the shaping etc. is performed by this class and the EventGenerator class, and that the pixel hits here can be fed directly to the Alpide chip at the given time.

Author

Simon Voigt Nesbo

Date

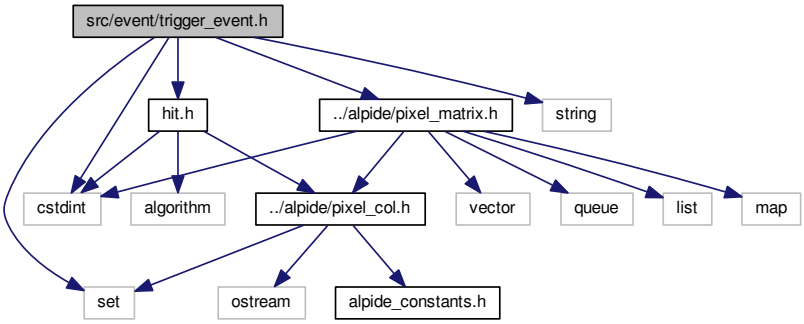
December 12, 2016

8.16 src/event/trigger_event.h File Reference

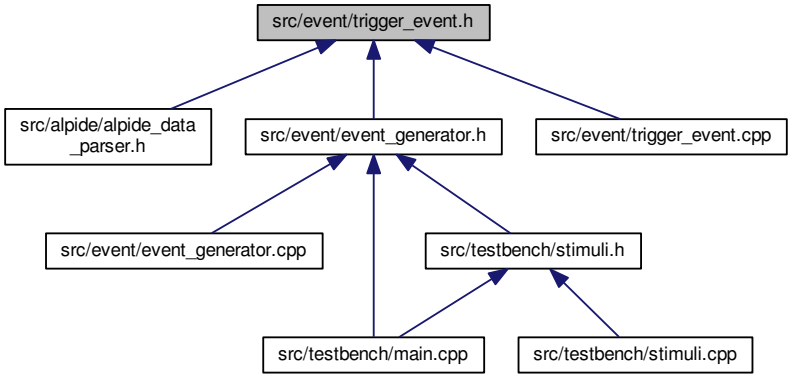
Trigger event class for Alpide SystemC simulation model. This class holds all the pixel hits for a trigger event for the whole detector. The philosophy behind this class is that the shaping etc. is performed by this class and the EventGenerator class, and that the pixel hits here can be fed directly to the Alpide chip at the given time.

```
#include "hit.h"
#include "../alpide/pixel_matrix.h"
#include <string>
#include <set>
#include <cstdint>
```

Include dependency graph for trigger_event.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TriggerEvent

Variables

- const TriggerEvent NoTriggerEvent

  A TriggerEvent that equals NoTriggerEvent is returned by some of the EventGenerator's functions which return a reference to an event, when there is no TriggerEvent to return.

8.16.1 Detailed Description

Trigger event class for Alpide SystemC simulation model. This class holds all the pixel hits for a trigger event for the whole detector. The philosophy behind this class is that the shaping etc. is performed by this class and the EventGenerator class, and that the pixel hits here can be fed directly to the Alpide chip at the given time.

Author

Simon Voigt Nesbo

Date

January 2, 2017

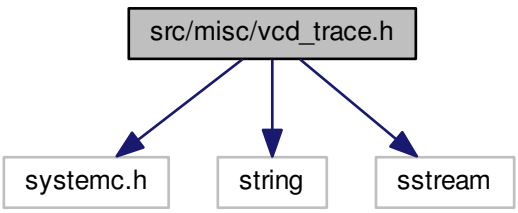
Todo Use SystemC time data type instead of int64_t?

8.17 src/misc/vcd_trace.h File Reference

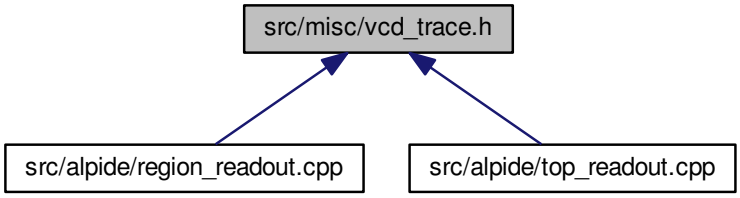
Common function for adding SystemC signals etc. to Value Change Dump (VCD) file.

```
#include <systemc.h>
#include <string>
#include <sstream>
```

Include dependency graph for vcd_trace.h:



This graph shows which files directly or indirectly include this file:



Functions

- template<class T >
  static void addTrace (sc_trace_file *wf, std::string name_prefix, std::string signal_name, T &signal)

  Add a SystemC signal/trace to VCD file, with desired signal hierarchy given by name_prefix.

8.17.1 Detailed Description

Common function for adding SystemC signals etc. to Value Change Dump (VCD) file.

Author

Simon Voigt Nesbo

Date

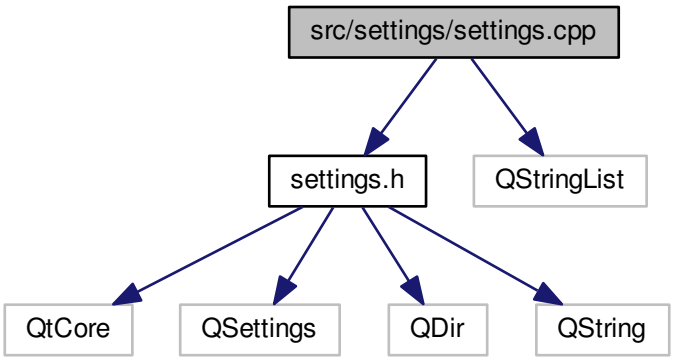
March 3, 2017

8.18 src/settings/settings.cpp File Reference

Source file for simulation settings file.

```
#include "settings.h"
#include <QStringList>
```

Include dependency graph for settings.cpp:



Functions

- QSettings * getSimSettings (const char *fileName)

  Open a file with simulation settings. If the file does not exist, it will be created. If any settings are missing, they will be initialized with default values. If no filename is specified, the default settings.txt file is used in the current directory.
- void setDefaultSimSettings (QSettings *readoutSimSettings)

  Set default settings for each setting that is missing in the QSettings object.

8.18.1 Detailed Description

Source file for simulation settings file.

Author

Simon Voigt Nesbo svn@hib.no

Date

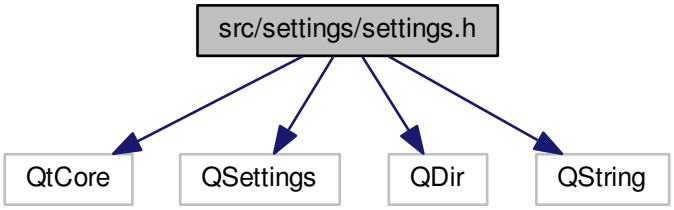
November 3, 2016 Some functions for reading the simulation settings file, and for initializing default settings if the settings file, or certain settings, are missing.

8.19 src/settings/settings.h File Reference

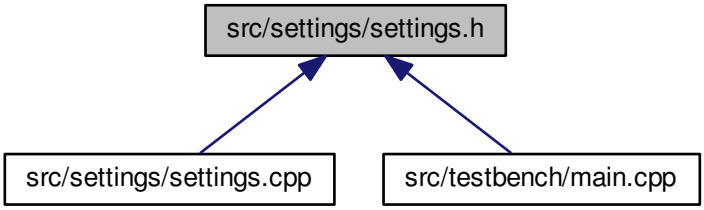
Header file for simulation settings file handling.

```
#include <QtCore>
#include <QSettings>
#include <QDir>
#include <QString>
```

Include dependency graph for settings.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **DEFAULT_DATA_OUTPUT_WRITE_VCD** "true"
- #define **DEFAULT_DATA_OUTPUT_WRITE_VCD_CLOCK** "false"
- #define **DEFAULT_DATA_OUTPUT_WRITE_EVENT_CSV** "true"
- #define **DEFAULT_SIMULATION_N_CHIPS** "25000"
- #define **DEFAULT_SIMULATION_N_EVENTS** "10000"
- #define **DEFAULT_SIMULATION_CONTINUOUS_MODE** "false"
- #define **DEFAULT_SIMULATION_RANDOM_SEED** "0"
- #define **DEFAULT_EVENT_HIT_MULTIPLICITY_DISTRIBUTION_TYPE** "discrete"
- #define **DEFAULT_EVENT_HIT_MULTIPLICITY_DISTRIBUTION_FILE** "multipl_dist_raw_bins.txt"
- #define **DEFAULT_EVENT_HIT_MULTIPLICITY_GAUSS_AVG** "2000"
- #define **DEFAULT_EVENT_HIT_MULTIPLICITY_GAUSS_STDDEV** "350"
- #define **DEFAULT_EVENT_HIT_DENSITY_MIN_BIAS_PER_CM2** "19"
- #define **DEFAULT_EVENT_BUNCH_CROSSING_RATE_NS** "25"
- #define **DEFAULT_EVENT_AVERAGE_EVENT_RATE_NS** "2500"
- #define **DEFAULT_EVENT_TRIGGER_DELAY_NS** "1000"

- #define **DEFAULT_EVENT_TRIGGER_FILTER_TIME_NS** "10000"
- #define **DEFAULT_EVENT_TRIGGER_FILTER_ENABLE** "true"
- #define **DEFAULT_EVENT_STROBE_ACTIVE_LENGTH_NS** "4800"
- #define **DEFAULT_EVENT_STROBE_INACTIVE_LENGTH_NS** "200"
- #define **DEFAULT_ALPIDE_CLUSTERING_ENABLE** "true"
- #define **DEFAULT_ALPIDE_REGION_FIFO_SIZE** "128"
- #define **DEFAULT_ALPIDE_DMU_FIFO_SIZE** "64"
- #define **DEFAULT_ALPIDE_REGION_SIZE** "32"
- #define **DEFAULT_ALPIDE_PIXEL_SHAPING_DEAD_TIME_NS** "200"
- #define **DEFAULT_ALPIDE_PIXEL_SHAPING_ACTIVE_TIME_NS** "6000"
- #define **DEFAULT_ALPIDE_MATRIX_READOUT_SPEED_FAST** "true"

Functions

- QSettings * getSimSettings (const char *fileName="settings.txt")

  Open a file with simulation settings. If the file does not exist, it will be created. If any settings are missing, they will be initialized with default values. If no filename is specified, the default settings.txt file is used in the current directory.
- void setDefaultSimSettings (QSettings *readoutSimSettings)

  Set default settings for each setting that is missing in the QSettings object.

8.19.1 Detailed Description

Header file for simulation settings file handling.

Author

Simon Voigt Nesbo svn@hib.no

Date

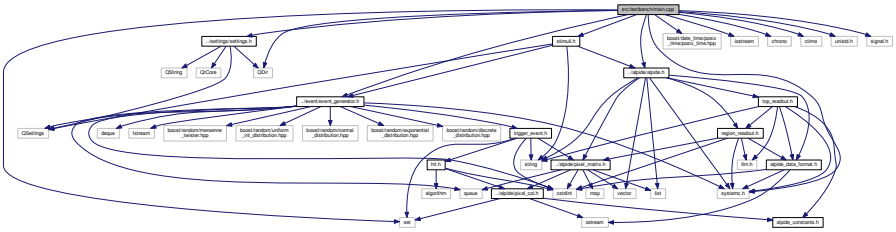
November 3, 2016 This file has definitions for default simulation settings, which can be used as default values or for generating the settings file if it is missing.

8.20 src/testbench/main.cpp File Reference

Main source file for Alpide Dataflow SystemC simulation testbench.

```
#include "../settings/settings.h"
#include "../event/event_generator.h"
#include "../alpide/alpide.h"
#include "stimuli.h"
#include <systemc.h>
#include "boost/date_time/posix_time/posix_time.hpp"
#include <set>
#include <iostream>
#include <chrono>
#include <ctime>
#include <QDir>
#include <unistd.h>
#include <signal.h>
```

Include dependency graph for main.cpp:



Functions

- void signal_callback_handler (int signum)

  Callback function for CTRL+C (SIGINT) signal, used for exiting the simulation nicely and not lose data if the user presses CTRL+C on the command line.
- std::string create_output_dir (const QSettings ∗settings)

  Create output directory "$PWD/sim_output/Run <timestamp>". Also writes a copy of the settings file used for the simulation to this path.
- int sc_main (int argc, char ∗∗argv)

Variables

- volatile bool **g_terminate_program** = false

8.20.1 Detailed Description

Main source file for Alpide Dataflow SystemC simulation testbench.

Author

Simon Voigt Nesbo

Date

December 11, 2016

8.20.2 Function Documentation

8.20.2.1 std::string create_output_dir (const QSettings ∗ *settings*)

Create output directory "$PWD/sim_output/Run <timestamp>". Also writes a copy of the settings file used for the simulation to this path.

Returns

Output directory path string

Referenced by sc_main().

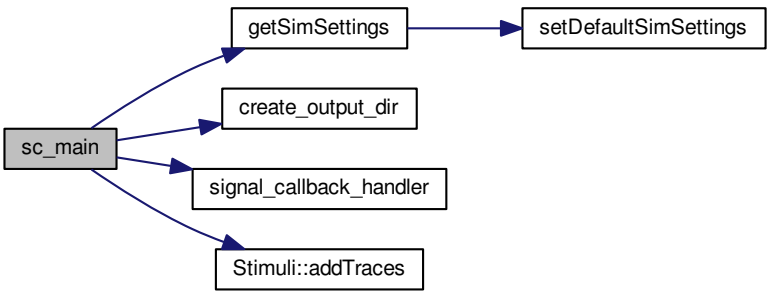
Here is the caller graph for this function:

8.20.2.2 int sc_main (int *argc*, char ** *argv*)

Todo Pass vcd trace object to constructor of Stimuli class and Alpide classes?

Todo Add a warning here if user tries to simulate over 1000 events with this option enabled, because it will consume 100s of megabytes

Here is the call graph for this function:

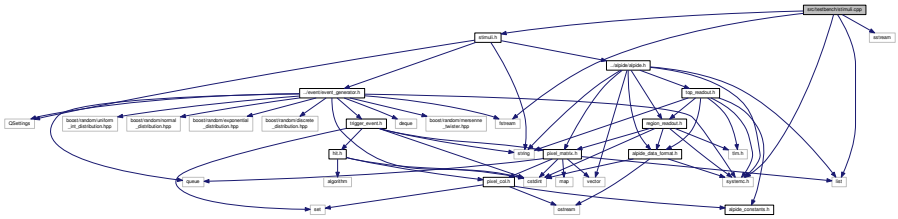


8.21 src/testbench/stimuli.cpp File Reference

Source file for stimuli function for Alpide Dataflow SystemC model.

```
#include "stimuli.h"
#include <systemc.h>
#include <list>
#include <sstream>
#include <fstream>
```

Include dependency graph for stimuli.cpp:



Functions

- void print_event_rate (const std::list< int > &t_delta_queue)

  Takes a list of t_delta values (time between events) for the last events, calculates the average event rate over those events, and prints it to std::cout. The list must be maintained by the caller.
- **SC_HAS_PROCESS** (Stimuli)

Variables

- volatile bool **g_terminate_program**

8.21.1 Detailed Description

Source file for stimuli function for Alpide Dataflow SystemC model.

Author

Simon Voigt Nesbo

Date

December 12, 2016

8.21.2 Function Documentation

8.21.2.1 void print_event_rate (const std::list< int > & *t_delta_queue*)

Takes a list of t_delta values (time between events) for the last events, calculates the average event rate over those events, and prints it to std::cout. The list must be maintained by the caller.

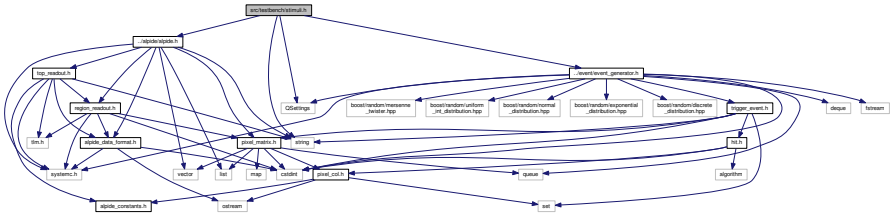
Todo Update/fix/remove this function.. currently not used..

8.22 src/testbench/stimuli.h File Reference

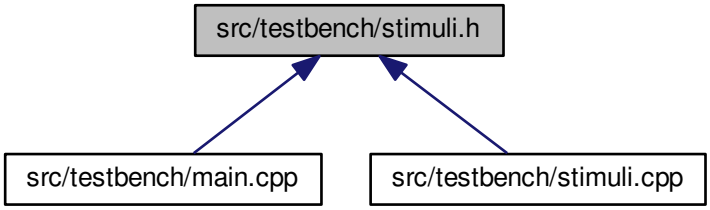
Header file for stimuli function for Alpide Dataflow SystemC model.

```
#include "../alpide/alpide.h"
#include "../event/event_generator.h"
#include <QSettings>
#include <string>
```

Include dependency graph for stimuli.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Stimuli

8.22.1 Detailed Description

Header file for stimuli function for Alpide Dataflow SystemC model.

Author

Simon Voigt Nesbo

Date

December 12, 2016

Index