SBND code - Bug #24211

Undefined cast to size_t on AddSPE() function both on DigiArapucaSBNDAlg and DigiPMTSBNDAlg

03/23/2020 08:58 PM - Iker de Icaza Astiz

Status:	Feedback	**Start date:**	03/23/2020
Priority:	High	**Due date:**	
Assignee:	Iker de Icaza Astiz	**% Done:**	100%
Category:	Simulation	**Estimated time:**	0.00 hour
Target version:			
First Occurred:		**Occurs In:**	

Description

This is been around since those two files were added in revision c5a40f005c, and also on the file before that had a different name on 8b729591cbb9. It has only now became apparent when I refactored them.

The original function looks like this:

```
  void DigiPMTSBNDAlg::AddSPE(size_t time_bin, std::vector<double>& wave)
  {
    size_t min = 0;
    size_t max = 0;
    if(time_bin < wave.size()) {
      min = time_bin;
      max = time_bin + pulsesize < wave.size() ? time_bin + pulsesize : wave.size();
      for(size_t i = min; i < max; i++) {
        wave[i] += wsp[i - min];
      }
    }
  }
```

The problem is that an event can have a negative time if time 0 is defined to be the neutrino interaction time. size_t is an unsigned integer and its behaviour is undefined when trying to cast a negative number.

```
The value of time_bin passed as parameter:         -637688
The value of time_bin inside AddSPE():              18446744073708913929
```

So given the code above whatever event came with negative time it just didn't went through this stage of the simulation.

With the purposes of improving performance I've refactored the code using std::transform and with the if statement before the function call. So it looks like this

```
  void DigiPMTSBNDAlg::AddSPE(size_t time_bin, std::vector<double>& wave)
  {
    size_t max = time_bin + pulsesize < wave.size() ? time_bin + pulsesize : wave.size();
    auto min_it = std::next(wave.begin(), time_bin);
    auto max_it = std::next(wave.begin(), max);
    std::transform(min_it, max_it,
                   wsp.begin(), min_it,
                   std::plus<double>( ));
  }
```

I didn't catch these errors because I wasn't using out of time events.

Right now it's not entirely obvious to me how to handle a waveform with negative indices.

If this has been shared with another experiment or picked from another experiment, it would be worth to see how they handle it, if they have.

Associated revisions

Revision cf097143 - 03/24/2020 08:31 PM - Iker de Icaza Astiz

Handle the casts from doubles to integers, avoid computing waveforms from times before the acquisition window.

This commit is meant to deal with issue #24211.

Revision 713663ad - 03/26/2020 07:56 AM - Iker de Icaza Astiz

Move the check to allow digitization time spread to populate the readout window

Related to issue #24211.

History

#1 - 03/23/2020 09:18 PM - Iker de Icaza Astiz

I've pushed 65367dc0b36 to mitigate this issue for now so that there are no segfaults.

#2 - 03/24/2020 05:47 PM - Gray Putnam

To summarize a discussion Iker and I had about this issue:

The original state of this code does not represent a bug -- it enforces a good behavior in a rather roundabout way.

In the optical digitization, the digitization is run for a first pass over the full configured window -- usually the TPC readout window. We run G4 for 1 TPC drift window before the TPC turns on (to ensure that the front porch is populated with cosmics). For photons that arrive before the digitization starts (i.e. in this prior drift window), the "time_bin" input to this function will be negative. This is because "time_bin" represents the time since the beginning of the digitization window, so negative for photons that arrive in the first simulated drift window.

Previously the code would cast the negative time_bin to a very large value in the size_t, check that it was bigger than the waveform size, and not include that photon in the waveform. The net behavior of this is what we want: photons that arrive outside the optical digitization window are not included in the waveform. This should be changed though to an explicit check on whether the photon time is outside the digitization window.

#3 - 03/24/2020 09:12 PM - Iker de Icaza Astiz

I've pushed branch bugfix/icaza_digitizers to deal with this issue.

Running test_runner develop_test_sbndcode I get

```
6 tests passed (54%), 0 tests failed, 5 tests with warnings, 0 tests skipped, out of 11
```

The full output can be find at: /sbnd/data/users/icaza/work/sbndcode/developHEAD/tests

The warnings are somewhat expected as I've improved the performance of the code, changed the way we're sampling photons and also because of less overall waveforms needed to get digitised. I explain this last reason below.

Gray's point is true only some fraction of the time as casting a negative int to an unsigned int is undefined and will not always produce a number that would prevent further processing.

But even if it did there are still a small fraction of waveforms that shouldn't have been digitised.

The above function gets called after having simulated the time response from the optical detectors. That is, some photons that arrived before the acquisition window are being digitised because adding the detector response grants them a positive time again, that will get casted to ints more along the expected behaviour.

Hence the result of this bug is that there is more waveforms being produced.

I believe those photons shouldn't be digitised, and I've coded this fix to reflect that. Please do let me know if you believe otherwise.

#4 - 03/26/2020 08:49 AM - Iker de Icaza Astiz

- % Done changed from 0 to 100

- Assignee set to Iker de Icaza Astiz

- Status changed from New to Feedback

After discussing with Michelle and Laura it seems the check should allow the detectors response to factor in. Revision 713663ad does this.

I have now merged to develop. I got 5 warnings related to lower cpu usage and changes in the data products size, both of which are expected.