Continuous-Readout Simulation with FairRoot on the Example of the $\overline{\text{P}}$ANDA Experiment

Tobias Stockmanns for the $\overline{\text{P}}$ANDA collaboration and the FairRoot developer team

Forschungszentrum Jülich GmbH, Wilhelm-Johnen-Strasse, 52428 Jülich

E-mail: t.stockmanns@fz-juelich.de

Abstract. Future particle physics experiments are searching more and more for rare decays which have similar signatures in the detector as the huge background. For those events usually simple selection criteria do not exist, which makes it impossible to implement a hardware-trigger based on a small subset of detector data. Therefore, all the detector data is read out continuously and processed on-the-fly to achieve a data reduction suitable for permanent storage and detailed analysis. To cope with these requirements of a triggerless readout, also the simulation software has to be adopted to add a continuous data production with pile-up effects and event overlapping in addition to the event-wise simulation. This simulated data is of utmost importance to get a realistic detector simulation, to develop event-building algorithms and to determine the hardware requirements for the DAQ system of the experiments. The possibility to simulate a continuous data stream was integrated into the FairRoot simulation framework. This running mode is called time-based simulation and a lot of effort was taken that one can switch seamlessly between the event-based and the time-based simulation mode. One experiment, which is using this new feature, is the PANDA experiment. It utilizes a quasi-continuous antiproton beam with a mean time between interactions of 50 ns. Because of the unbunched structure of the beam the interaction time follows a Poisson statistic with a high probability of events with short time distances. Depending on the time resolution of the sub-detectors this leads to an overlap of up to 20 events inside a sub-detector. This makes it an ideal test candidate for the time-based simulation. In the following text an overview of the implementation of the time-based simulation mode in FairRoot is given and some examples for the $\overline{\text{P}}$ANDAexperiment are shown.

1. Motivation

Modern particle physics experiments are searching for rare physical events which require high luminosities to create sufficient statistics to find and precisely measure new states. As a consequence the amount of data measured by the detector systems are beyond any capability of permanent storage. Therefore, an efficient system to filter the interesting data out of the huge uninteresting background is needed. In current detector systems a multi-stage triggering system is typically used which starts to select events based on a small subset of data and requests more and more data for the events selected in the previous stage. This approach works well for signal signatures which can easily be distinguished from background by fast detectors but fails if the signatures of background and signal events are very similar. In addition, the staged approach limits the flexibility to modify the selection criteria to search for different events. Thus, a strong tendency by newer detector systems and upgrades can be seen to reduce the trigger stages or

even run the data selection on the complete data set. This approach leads to huge requirements on the data transmission and processing capabilities of the readout and data selection systems which have to analyze the data online.

To develop the needed hardware and the reconstruction algorithms, a realistic simulation of the data stream of the detector is needed. For this purpose an extension of the *FairRoot* framework [1] was developed. This extension allows to move over from an event-based simulation, where each event is independent from each other, to a time-based simulation, which takes beam structures and event overlaps into account. The change between these two modes can be done individually for each sub-detector and a mixed operation is possible.

2. Implementation

The FairRoot framework uses a simulation chain subdivided into several steps. The first step is the *event generation* usually done by an external physics simulator like Pythia [8], EvtGen [7] or others. They simulate the primary interaction between particles without taking any detector layout into account and generate position and four-momentum vectors for all created particles. These particles are than propagated through the detector by e.g. Geant4 [5] in the *Monte-Carlo simulation* stage. Here all interactions with the detector material and possible decays of the particles are simulated and the position, time and energy loss of the particles in the detector are recorded. In the *digitization* stage, the detector response is simulated with the goal to reproduce the real data stream of the detector as realistically as possible. After the digitization of the hits in the detector, the *reconstruction* stage converts the digitized detector information back into physical information. This usually contains a set of different algorithms first operating locally on the data from one sub-detector and then combining this data to global information like complete track information. In the last *analysis* stage, the measured particles are combined to the physics channels under study and the signals are extracted out of the background.

Each of the stages is usually done event-based, meaning that each physical interaction is completely independent of the previous one and the time between two events is not important. This is also reflected in the data structure of *ROOT* [2] which is the underlying framework of *FairRoot*. The data handling in *ROOT* is organized in a *tree*. Each data object has its own *branch* in the *tree*. The *branch* is subdivided into *entries*, where one *entry* contains all the data of one event of the data object. The same *entries* in all branches contain the data from the same event. This treatment of the simulation does not match with the real situation inside experiments where event mixing and pile-up happens, which are both dependent on the time between events and the time resolution of the different sub-detectors.

To convert an event-based simulation to a more realistic time-based simulation four modifications of the *FairRoot* framework have been done, which are explained in the following sections.

2.1. Event Mixing and Time Association

FairRoot has the built-in functionality for mixing different signal channels with background data. This is done after the event generation and the Monte-Carlo simulation stage and allows for reuse of background data for other signal channels. The ratio of signal data to background data can be set either by the number of background events per signal event or by the time gap between two signal events.

Additionally, a global time is assigned to each event. The time difference between two events is calculated following a random distribution which can be user-defined via a ROOT `TF1` function. Built in methods are a Poisson distribution with a user given mean time and a uniform distribution between a minimum and maximum value. Furthermore, a time gap can be set where no events take place e.g. for regular gaps in the beam.

2.2. Data Buffering

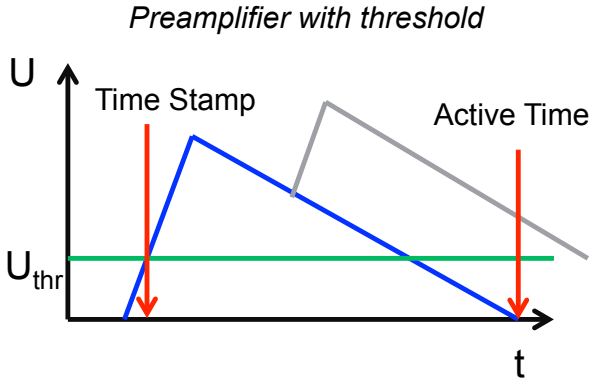

Figure 1. A simplified preamplifier signal as an example for the two time marks *time stamp* and *active time* used in the time-based simulation.

To be able to simulate pile-up effects and have interferences beyond event boundaries, a dedicated data buffer was developed. This so called *FairWriteoutBuffer* gets two pieces of information; the data object itself and an *active time*. The *active time* is the absolute time up to which consecutive hits in the same detector element can interfere. This is illustrated in Fig. 1. Here, a schematic output signal of a preamplifier is shown. When a particle passes through the detector, the deposited energy is integrated over time. Once the signal is higher than a threshold, a *time stamp* is associated to the hit, which is needed to match this hit with hits from other sub-detectors. After the energy is fully integrated, the signal returns back to its baseline with a certain slope. The time over threshold (ToT) is a measure of the amplitude of the signal. As long as the signal is above the baseline it can influence any other signal hitting the same detector element. Therefore the *active time* for this signal is the time the signal has not returned to its baseline.

If this detector element is hit before the active time is over, the method `Modify()` is called. What happens inside `Modify()` is sub-detector specific and strongly depends on the type of the detector and the connected electronics. In the shown example in Fig. 1, a pile-up signal in gray is superimposed over the first signal in blue. As a consequence, the ToT of the first hit is extended by the charge of the pile-up hit, the active time is extended until the combined signal would return to baseline and the second hit is not recorded.

When an event is read in, all hits with an active time lower than the event time of the new event are removed from the buffer and written out for further processing or storage to disk. As a result a data stream is created, which is ordered by the *active time* of a hit and not by the *time stamp*. This feature is necessary to create an output data stream which is similar to the data coming out of a realistic detector. As an example the simulated output of the Micro Vertex Detector (MVD) data of the $\overline{\text{P}}$ANDA experiment is shown in Fig. 2 (left). In the diagram the time stamp vs. the position in the data stream is plotted. A correlation between the time stamp and the position can be seen but this correlation is smeared depending on the readout characteristics of the Micro Vertex Detector.

2.3. Time Stamp Sorting

For reading back the data in a time-based manner, it is important to sort the data according to their *time stamps*. For this purpose, a ring buffer called *FairRingBuffer* was developed which uses the characteristics of the data stream that the data is not randomized over the full data set

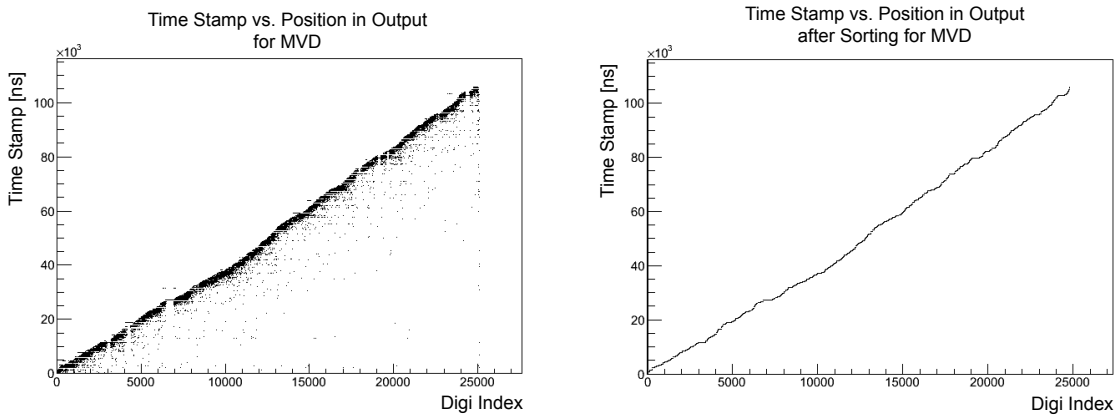

Figure 2. Assigned time stamp of Micro Vertex Detector hits versus position in data stream before (left) and after sorting (right) by the time stamps of the hits. In addition a pile-up signal in gray is shown.

but only over a limited time window depending on the sub-detector properties. The ring buffer is subdivided into cells which can contain multiple hits. The width of one cell corresponds to the time measurement precision of the sub-detector. The total number of cells corresponds to the randomization of hits in time and gives the total size of the buffer in time when multiplied with the cell widths. If new data would override old data in the buffer, the old data is removed from the ring sorter and given to further processing. In addition, a start pointer is moved to the position behind the new data. Fig. 2 (right) shows the result of the sorting of the data of the MVD. It can be seen that the sorting process worked as expected. The position in the data stream corresponds to the time-wise sorting of the data.

2.4. Data Readback

To read back the data which has been generated in the previous stages in a time-based manner, a `GetData()` method exists which does not operate on events any longer but on the *time stamps* of the hits. The `GetData()` method uses pairs of functors and values to determine how to retrieve the data out of the time ordered data set. Two versions of the `GetData()` method exists: one with one pair of functor/value and one with two pairs of functor/value. The `GetData()` method with one pair of functor/value always runs in one time direction through the data and returns data only once while the version with two pairs can return data multiple times and is used e.g. to return overlapping time windows.

Two predefined functors exist: The first is a so called `StopTime` functor which is used to extract data up to a given absolute time. The second functor is the `TimeGap` functor. It looks for gaps in time larger than a given length between two consecutive hits and extracts the data before the time gap. The `TimeGap` functor is very useful to do a fast event building for detectors with a time resolution much shorter than the average time between two events.

3. Examples from $\overline{\mathrm{P}}$ANDA

$\overline{\mathrm{P}}$ANDA is one of the key experiments of the future Facility for Antiproton and Ion Research (FAIR) currently under construction at Darmstadt, Germany. $\overline{\mathrm{P}}$ANDA will study the transition region between perturbative and non-perturbative QCD with a phase-space cooled antiproton beam on a fixed target with a maximum beam momentum of 15 GeV/c. One special feature of the experiment is the quasi-continuous beam. The time between two beam target interactions follows a Poisson distribution with a mean time between two events of 100 ns and with a high

probability of shorter time gaps between two events. In addition, $\overline{\text{P}}$ANDA will run with an online event filter operating on the complete data stream of 200 GByte/s and reducing it by a factor of 1000. The time-based simulation of the detector is essential for the experiment to develop the DAQ system and to determine its performance. Therefore, the simulation software of $\overline{\text{P}}$ANDA, PandaRoot [9], which is derived from FairRoot is modified to run both event and time-based.

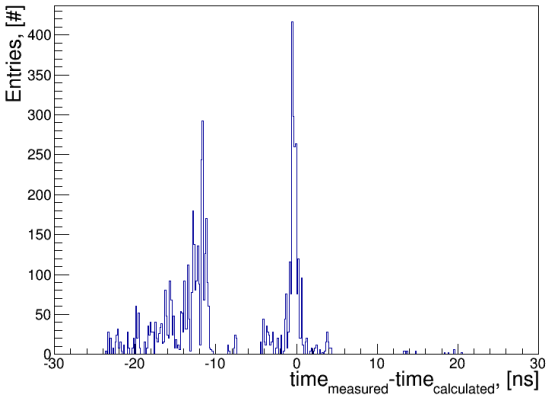

Figure 3. Photon arrival time in a DIRC bar for a pile-up event in a time-based simulation. [4]

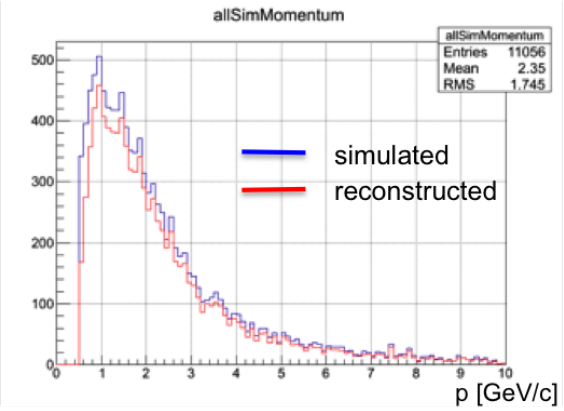

Figure 4. Track finding efficiency in the $\overline{\text{P}}$ANDA GEM detector for different track momenta in a time-based simulation. [6]

One example of the results of such a simulation is shown in Fig. 3. It shows the arrival time of photons in a DIRC bar. Clearly two peaks are visible. The first is coming from a previous event, the second is from the current event generated via the time-based simulation. With the time-based simulation, it was shown that 4% of the background events generate such a pile-up effect but 90% of these events can be separated by studying the time-distribution in detail. [3][4]

Another example for the importance of time-based simulation is shown in Fig. 4. Here, the complete reconstruction chain for the $\overline{\text{P}}$ANDA GEM detector was performed in a time-based simulation up to the track reconstruction. The graph shows the simulated tracks in blue and the reconstructed tracks in red. The reconstruction efficiency is larger than 87% for tracks above 1 GeV/c momentum. This compares to 95% track finding efficiency for an event-based simulation. [6]

The time-based simulation of the $\overline{\text{P}}$ANDA detector shows that the performance of the detector degrades with event overlapping and the time to reconstruct events rises because of higher multiplicities. Furthermore, the reconstruction algorithms have to be adopted to operate with *time* as an additional parameter. These results show the importance of time-based simulation to construct a successfully running detector.

4. Summary

An extension of the *FairRoot* framework was developed which allows to simulate events in a time-based way. The extension takes sub-detector specific dead-times and pile-up behavior into account which is needed to come to a realistic simulation of the data stream. This data stream serves as a basis for the layout of the DAQ and the development of new reconstruction algorithms. One key feature of the time-based extension is the possibility to seamlessly change between time-based and event-based simulation on a sub-detector level.

The time-based simulation is heavily used for the simulation of the $\overline{\text{P}}$ANDA experiment which strongly depends on this tool because of its specific beam characteristics and the new feature

of an online event filter. First simulation results show the influence on the performance of the detector when using a time-based simulation and reconstruction chain. The modification of all sub-detectors of $\overline{\mathrm{P}}$ANDA to the time-based simulation is almost finished and current efforts focus on the development of suitable reconstruction algorithms.

Bibliography

[1] M. Al-Turany, D. Bertini, R. Karabowicz, D. Kresan, P. Malzacher, T. Stockmanns, and F. Uhlig. The FairRoot framework. *Journal of Physics: Conference Series*, 396(2):022001, 2012.

[2] Rene Brun and Fons Rademakers. ROOT — An Object Oriented Data Analysis Framework. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 389(1):81–86, 1997.

[3] Roman Dzhygadlo. Status of the Barrel DIRC software. PANDA Internal Report, Jun 2014.

[4] Roman Dzhygadlo. Status of the time-based sim and reco. PANDA Internal Report, Feb 2015.

[5] S. Agostinelli et.al. Geant4 – a simulation toolkit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 506(3):250 – 303, 2003.

[6] Radoslaw Karabowicz. An Event Building scenario in the trigger-less PANDA experiment. *J.Phys.Conf.Ser.*, 513:012016, 2014.

[7] David J. Lange. The EvtGen particle decay simulation package. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 462(1):152–155, 2001.

[8] Torbjörn Sjöstranda, Stefan Askb, Jesper R. Christiansena, Richard Corkea, Nishita Desaic, Philip Iltend, Stephen Mrennae, Stefan Prestelf, Christine O. Rasmussena, and Peter Z. Skandsh. An introduction to PYTHIA 8.2. *Computer Physics Communications*, Feb 2015. In Press, Accepted Manuscript.

[9] Stefano Spataro. Event Reconstruction in the PandaRoot framework. *Journal of Physics: Conference Series*, 396(2):022048, 2012.