A data parallel digitizer for a time-based simulation of CMOS Monolithic Active Pixel Sensors with FairRoot

P. Sitzmann\textsuperscript{a}, S. Amar-Youcef\textsuperscript{a}, D. Doering\textsuperscript{a}, M. Deveaux\textsuperscript{a}, I. Fröhlich\textsuperscript{a}, M. Koziel\textsuperscript{a}, E. Krebs\textsuperscript{a}, B. Linnik\textsuperscript{a}, J. Michel\textsuperscript{a}, B. Milanovic\textsuperscript{a}, C. Müntz\textsuperscript{a}, Q. Li\textsuperscript{a,b}, J. Stroth\textsuperscript{a,c}, and T. Tischler\textsuperscript{a}

\textsuperscript{a} Goethe University Frankfurt, Max-von-Laue-Str. 1, 60438 Frankfurt/M, Germany
\textsuperscript{b} Central China Normal University, Luoyulu.152, 430079 Wuhan, China
\textsuperscript{c} GSI Helmholtzzentrum für Schwerionenforschung, 64291 Darmstadt, Germany

E-mail: deveaux@physik.uni-frankfurt.de, s.amar@gsi.de

Abstract. CMOS Monolithic Active Pixel Sensors (MAPS) demonstrated excellent performances in the field of charged particle tracking. They feature an excellent single point resolution of few $\mu$m, a light material budget of 0.05% $X_0$ in combination with a good radiation tolerance and time resolution. This makes the sensors a valuable technology for micro vertex detectors (MVD) of various experiments in heavy ion and particle physics like STAR and CBM. State of the art MAPS are equipped with a rolling shutter readout. Therefore, the data of one individual event is typically found in more than one data train generated by the sensor. This paper presents a concept to introduce this feature in both simulation and data analysis, taking profit of the sensor topology of the MVD. This topology allows to use for massive parallel data streaming and handling strategies within the FairRoot framework.

1. Introduction

Next generation heavy-ion physics experiments, like the CBM experiment at FAIR [1], aim for extremely high collision rates of up to $10^7$ collisions per second. Combining these high collision rates with complex event signatures like displaced vertices is out of the scope of traditional trigger concepts. In order to overcome this obstacle, CBM aims at using a free running DAQ concept: the particle hits will be recorded by self-triggered detector systems and complemented with a time information before the data are shipped to a central compute cluster. The latter performs event building and a partial event reconstruction before a trigger decision is taken.

Detector simulation and data analysis tools like CBM-/FairRoot [2] and the embedded GEANT [3] package handle data event-by-event. This is perfectly appropriate if the events are defined by a triggered DAQ but not optimized to represent potential timing issues of sensors and DAQ instances operating in a trigger-less mode. Timing effects have however, to be accounted for when studying questions of real-time event building and tracking on the future CBM computing cluster. This holds in particular for CMOS Monolithic Active Pixel Sensors (MAPS) as their rolling shutter readout imposes an exception in this concept. Accounting for the need of a fast, parallel and realistic simulation, we performed an exploratory study on how to build a related detector description software, which is compatible with CBM-/FairRoot.
To introduce our concept, we will discuss the rolling shutter readout of MAPS and its consequences on simulation and data analysis and remind the abilities and limits of the previous simulation software. Hereafter, we will introduce our strategy to extend this software to a version representing timing and being compatible with parallel and many-core computing. Finally we will give a status on the implementation of this concept and discuss its limits.

2. The rolling-shutter readout of MAPS

During our study, we aimed at representing the MAPS prototype MIMOSA-26 [4], which is shown in Figure 1, in FairRoot. This sensor is considered as a representative precursor of the future CBM-sensor. It hosts a matrix made from 1152 columns composed from \(N_{\text{line}} = 576\) pixels with 18.4 \(\mu\text{m}\) pitch. The signals of one line of pixels are sent to an array of 1152 discriminator blocks, which are located aside the sensor matrix. After discriminating the pixel signals, an on-chip zero-suppression logic scans the digital pattern obtained for groups of up to four consecutive fired pixels. The processing time for one line is \(t_l = 200\ \text{ns}\). Once all lines are consecutively processed, the \(t_{\text{int}} = t_l \cdot N_{\text{line}} = 115.2\ \mu\text{s}\) long readout cycle is restarted. Simultaneously, the data of the completed frame with number \(f\) is pushed via two digital 80 Mbps links from the output buffer of the sensor to the controlling FPGA. The absolute time of readout of line \(i\) is determined according to:

\[
t_r(f,i) = t_0 + f \cdot t_{\text{int}} + i \cdot t_l \quad (0 \leq i < N_{\text{line}}, \ f \geq 0, \ t_0 = \text{sensor start time}).
\]

(1)

Hits of particles impinging a line in between two readouts are sensed and kept in an on-pixel sample and hold circuit until the next pixel readout occurs. Hereafter, the data is added to the data train related to the frame under readout. Consequently, a data train related to a frame \(f\) contains data on hits, which imping at a time \(t_{\text{hit}}\) fulfilling the condition:

\[
t_r(f-1,i) < t_{\text{hit}} \leq t_r(f,i) \quad (0 \leq i < N_{\text{line}}).
\]

(2)

Therefore, the impact time of the hits found in this data array depends on the line number, which is referred to as rolling-shutter readout. The unintuitive consequence of this data handling is the fact that the hits created by one reaction are typically found in two consecutive “frames”\(^1\). To understand this, one has to assume that the frame with number \(f = f_i\) is partially readout once the event occurs and that the line \(i = i_r\) is under readout that precise moment\(^2\). Particles

\(^1\) For the sake of simplicity, the array of hits recorded by a MAPS during the readout of one frame will be called “frame” hereafter. This frame has to be distinguished from arrays containing data on all particles or hits related to a nuclear collision, which will be called “event”.

\(^2\) We neglect the time-of-flight of the particles, which is short as compared to \(t_l\).
impinging the lines $i > i_r$ fulfill condition 2 for frame $f = f_r$. Particles impinging the lines $i < i_r$ are sensed and stored in the pixel. However, as the lines are located behind the line under readout, they fulfill condition 2 for frame $f = f_r + 1$ only and are therefore stored in this data array.

3. Simulation and analysis software

3.1. Limits of the previous digitizer versions

The current version of our digitizer software is being developed for the CBM-/FairRoot framework [2]. The core of the simulation is a pixel response model for MAPS (see [5]), which was built by parameterizing the response of MIMOSA-17 (30 µm pixel pitch, standard epitaxial layer) to $\sim 120$ GeV pions from the CERN-SPS. After tuning the model parameters, a good agreement between experiment and simulation was observed (see Figure 3). Meanwhile, the model was extended to describe the sensors with a pitch of 10µm - 30µm and with the high-resistivity epitaxial layer, which will presumably be used in CBM-MVD.

So far, the pixel response model acts on “frames”, which have to be built by earlier instances of the simulation. This contradicts the native, “event” based data handling of FairRoot and GEANT due to the above-mentioned space-time character of a “frame” delivered by MAPS (rolling-shutter). While studying the feasibility of particle reconstructions, we circumvented this problem by setting a “frame” equal to an “event”. This is equivalent to assuming an ideal event building and a more elaborated solution is needed once one aims for studying event building strategies. Additional requirements on an improved code arose from the wish to provide a parallel software, which allows to test real-time analysis codes on the many-core computing environment of CBM. As the sensor technology and the computing concept are still under development, we aimed at a flexible software, which is easy to adapt to future technology choices.

3.2. The software concept

The core of the novel computing concept was derived from the ambitioned option for parallel processing. An event parallel approach was excluded: each “frame” overlaps two “events”, which destroys the related data parallelism during detector simulation. On the other hand, the sensors, their readout systems and data streams are separated and therefore parallel by hardware. Therefore, we decided to follow a sensor parallel concept and to encapsulate all data needed for processing into a so-called MVD-Sensor object as shown in Figure 4 (left). By adding copies of all geometry and technical information on the sensor, we avoid conflicting RAM accesses. The non-parallel concept of the ROOT-TTask\(^3\) [6] was translated to a plugin-concept:

\[^3\] The ROOT/FairRoot-class TTask provides an interface for an event-by-event data processing. The concept allows for an easy and efficient data processing in complex and flexible processing chains. See ROOT manual [6].
instead of building one task, which loops over the different sensors, each sensor holds an array of pointers to so-called MVD-task objects, which contain their input and output data together with methods for data processing. By looping over the array, MVD-Sensors execute a processing chain (see Figure 4, right). MVD-tasks are implemented as virtual base classes, which allows for an easy modification and extension of this processing chain. The data types and the coordinate transformations used by the MVD-tasks are kept flexible. Any future data types or coordinate systems, which can be expressed with ROOT - TClonesArrays\textsuperscript{4} and the TGeoMatrix\textsuperscript{5} may be accommodated.

To interface FairRoot, we build the so-called MVD-detector object. This singleton constructs an array containing all MVD-Sensors at start time and provides to them the required information from the central data base(s). During simulation, it distributes the input obtained from the GEANT-simulation to the different sensors. The data processing in the individual sensors may be triggered by calling the related methods of the MVD-sensors from a potentially parallel loop within the MVD-detector. To interface the FairRoot, the MVD-detector may be steered by a FairRoot - TTasks. However, once they received their input, the MVD-sensors may conceptually act independently of the framework until their output has to be collected.

A MVD-tasks may also act as an unpacker receiving data streams from real sensors. If so, the detector response model is replaced by this instance and the more downstream MVD-tasks may handle experimental data in a fully parallel way. Doing so provides the option of using MVD-related reconstruction codes in both, simulation and data analysis, which is a central requirement of the CBM analysis software.

3.3. Time-based simulation of MAPS
In order to represent the rolling-shutter readout of MAPS in the simulation, we designed a second class of plug-ins, the so-called MVD-buffers. These buffers are to extract the content of

\textsuperscript{4} The ROOT-class TClonesArray organizes complex data sets as an array of identical data objects. Among the strong points are efficient memory and file management. The class forms the standard for data handling in FairRoot. See ROOT manual \textsuperscript{6}.

\textsuperscript{5} The ROOT-class TGeoMatrix defines the ROOT standard interface for geometrical operations like coordinate transformations. See ROOT manual \textsuperscript{6}.
MAPS “frames” from the “events” provided by GEANT and to build arrays according to the time representation used in the DAQ from the “frames”. The first buffer receives its input via the MVD-detector from FairRoot. Particle hits are extracted from the “events” and moved to the “frames” of the MVD-sensors based on the position and impact time of the individual hits. To account for the possible case of events following each other such rapidly that the fast particles of the second event arrive before the slow particles of the first event have left the detector, a frame is considered completed only, once three consecutive incoming “events” do not contain hits for this “frame” anymore. Hereafter, the “frame” is sent to the MVD-tasks modeling the pixel and sensor response. As the related hardware does indeed handle “frames”, the arrays may be processed in a straightforward way within those instances. The output MVD-buffer finally collects the hits in the output format of the simulation chain. Once the buffer is filled, the data can be requested by downstream instances simulating / executing event building and tracking.

3.4. Implementation and limits
The above mentioned software concept was implemented and tested in CbmRoot in the absence of other detector description codes and downstream processing instances. Tests confirmed the validity of the code and first attempts to perform parallel code execution by means of OpenMP were carried out. Based on the successful tests, we aim at integrating the event based version of the software to CbmRoot. As a next step, we intend to add further MVD-tasks to represent for example the native data format of the sensors and of intermediate preprocessing instances.

Concerning the representation of the rolling-shutter readout in the simulation, we have identified a number of issues, which cannot be solved within an event-based version of the FairRoot-framework. In particular, the input and output buffers of the MVD-software must be filled before any output is created. This causes a latency of few events, which does not exist in tasks describing other sub-detectors and does therefore destroy the event-based synchronization of the framework. To overcome this point, one requires a central instance for buffering data and handling latencies of the individual detector simulation packages on the framework level. Related software packages became available only recently and we intend to test their compatibility to our concept soon.

4. Summary and conclusion
CMOS Monolithic Active Pixel Sensors like the ones foreseen in the MVD of the future CBM-experiment are read out by means of a rolling-shutter. Hence, data generated by a single reaction is typically distributed to two consecutively recorded data arrays (frames) submitted by the readout electronics of the sensors. Simulations aiming at testing and optimizing the event building network and the tracking of CBM have to consider this effect.

In order to extend the existing detector simulation package of the CBM-MVD to time-based simulations, we aimed at building a data parallel software package, which represents the rolling-shutter and which can be extended for parallel computing as required for performing software tests in the future CBM many core computing environment. Our exploratory study has already satisfied major parts of the goals by exploiting the sensor parallelism of the future hardware. This was done by encapsulating the data to be processed, the necessary methods and the required information on the sensor of interest into a dedicated object, which can be processed in parallel. We found that this solution is well suited to perform sensor parallel processing within an event-based simulation framework. However, the ambitioned goal to simulate the rolling-shutter readout was not reached as the necessary input and output buffers break the synchronization concept of event-based simulations. To overcome this issue, we will exploit the central buffer concepts, which became available during a recent, dedicated update of FairRoot.
Acknowledgements
This work was supported by HIC for FAIR and BMBF (06FY9099I and 05P12RFFC7).

References
[1] B. Friman et al. (eds), *The CBM physics book*, Springer (2011), ISBN 978-3-642-13293-3
[2] M. Al-Turany et al., *The FairRoot framework*, J. Phys.: Conf Ser 396 (2012) 022001, http://fairroot.gsi.de/
[3] S. Agostinelli et al., GEANT4 - a simulation toolkit, NIM-A, Vol 506 (2003), Iss. 3 P. 250-303
[4] M. Deveaux et al., *Radiation tolerance of a column parallel CMOS sensor with high resistivity epitaxial layer*, 2011 JINST 6 C02004
[5] C. Dritsa et al., *A detector response model for CMOS Monolithic Active Pixel Sensors*, PoS(Bormio2010)015
[6] R. Brun, F. Rademakers, *ROOT - An object oriented data analysis framework*, NIM-A, Vol 389 (1997), Iss. 1-2, P. 81-86, detailed description and manuals: root.cern.ch