A trigger simulation framework for the ALICE experiment

F Antinori\textsuperscript{1,2}, F Carminati\textsuperscript{2}, A Gheata\textsuperscript{2}, M Gheata\textsuperscript{3}
For the ALICE Collaboration
\textsuperscript{1} INFN Padova, Italy
\textsuperscript{2} CERN Geneva, Switzerland
\textsuperscript{3} Institute for Space Sciences, Bucharest, Romania

E-mail: andrei.gheata@cern.ch

Abstract. A realistic simulation of the trigger system in a complex HEP experiment is essential for performing detailed trigger efficiency studies. The ALICE trigger simulation is evolving towards a framework capable of replaying the full trigger chain starting from the input to the individual trigger processors and ending with the decision mechanisms of the ALICE central trigger processor. This paper describes the new ALICE trigger simulation framework that is being tested and deployed. The framework handles details like trigger levels, signal delays and busy signals, implementing the trigger logic via customizable trigger device objects managed by a robust scheduling mechanism. A big advantage is the high flexibility of the framework, which is able to mix together components described with very different levels of detail. The framework is being gradually integrated within the ALICE simulation and reconstruction frameworks.

1. The ALICE trigger system
The ALICE trigger system operates in several different running modes, being designed to deal with very high multiplicity environment, detectors with fast and slow response time, and many different physics requirements. The main design features of the trigger were imposed by the technical features of the ALICE detector [1].

\textbf{Figure 1.} Overview of the ALICE trigger system.
- **Use of global trigger selections.** Trigger signals correspond to global features of the event, and trigger conditions are restricted to Boolean combinations of these. More detailed analysis is left to the High Level Trigger (HLT).

- **Continuous checking of time separation of events.** The trigger takes into account all the interactions occurring in the detector, and can veto all incoming events that are too close in time with respect to the current interaction. This feature, called *past–future protection*, ensures that spurious tracks from out–of–time events are avoided.

- **Faster trigger rates for some detectors.** While the TPC (main tracking detector) is constrained to relatively low event rates (300 Hz for Pb-Pb, 1 kHz for p-p), there are other detectors like ZDC, ITS SPD, PMD, V0, T0 or FMD) which can sustain much higher event rates. Where it makes sense to do so in order to improve statistics for physics, groups of detectors, called detector *clusters*, are read out at higher rates. In this way ALICE can make use of the full capability of its faster detectors to acquire useful data samples for physics.

- **Dynamic suppression of common triggers.** To avoid saturating the DAQ front end data storage bandwidth without affecting recording of the rare processes, high rate triggers are inhibited when the occupancy exceeds a given threshold.

2. **Offline trigger simulation**

The main purpose of the offline simulation is to be able to replay the full ALICE trigger chain, based on different configurations of the central trigger processor (CTP) reflecting the different features mentioned above. This allows to take into account the decision of the trigger system in the global detector simulation, but above all it allows detailed trigger efficiency studies for optimizing the data acquisition with respect to several physics observables.

2.1. **Representation for a generic trigger processor**

The initial ALICE trigger simulation was based on a simplified representation for the Central Trigger Processor. Each trigger detector provided the algorithm to compute its specific outputs for the CTP starting from the detector’s digitized signal. The trigger simulation was done in a simple loop over the trigger detectors filling all CTP inputs. The CTP algorithm was decoding this information based on the trigger partition for a given run (configuration of trigger detectors and their modes). This simplified view of the trigger system could not reproduce well the dependencies between different trigger processors in certain configurations, or certain pre-trigger conditions. Electronics and cable delays or other time-dependent effects affecting sequencing (like the BUSY time of the trigger detectors) were also ignored, so the simulation of the trigger could not actually replay the details of the full chain. We have chosen to take into account all these dependencies and to introduce the time component in a new framework.

The implementation of the trigger simulation is using C++ and the offline framework of ALICE [3] (*AliRoot* package). The schema is based on a simple representation for a generic trigger processor device, as illustrated in figure 2. The base C++ class representing such trigger processor allows for an arbitrary number of inputs and outputs slots and defines the API for the trigger specific algorithms. The data handled by the processors are generally Boolean, but certain inputs can be configured to use data arrays or custom objects in order to simplify the implementation of the actual trigger detector. The most common use is to group several trigger inputs corresponding to separate wires of the actual device in a single stream handled by one input slot of the simulated device.
Figure 2. Trigger processors are mapped to object-oriented representations that have a number of inputs and outputs. Their main functionality is to implement all relevant trigger response functions (outputs as function of inputs and configuration parameters) and to schedule their sequence according to the trigger logic [2].

The main functionality to be implemented by the trigger processor classes is the device response function. These response functions are class methods that give the output as function of the input values and possibly other configuration parameters. Complex trigger devices may define several response functions. An output becomes active only when the corresponding response function is fired. An output of a given device can be connected to compatible inputs of another device. The framework does not forbid an output to be connected to the input of the same device. This allows simulating some more complex response of a given processor or chaining responses of separate trigger detectors which have a given sequence in time. Simulated trigger devices are self contained, i.e. they define their structure and response functions without explicitly depending on other devices. The actual dependencies and the trigger schema are reproduced by connecting the components of the trigger chain via I/O slots at run time.

The framework general design is the following: several C++ objects representing trigger processors are interconnected via I/O slots. Every such object may have a reader that fills the inputs for each event from a data stream (raw or digitized). The response functions of all the trigger processors are replayed by a manager class (scheduler) that will be described in the next subsection. The final trigger decision is simulated by the CTP representation at the end of this chain.

2.2. Replaying the trigger chain

The way in which data are treated in the simulation is a simplified model of the hardware trigger. The goal is actually to reproduce the sequence of the trigger decisions and to emulate the response functions as well as possible. There is no need for a detailed signal treatment but it is important to compute all responses in the appropriate order so that all the needed inputs are available while executing the sequence. A simple loop over the trigger detector objects (as in the previous implementation) cannot achieve in some particular cases the correct sequencing that end-up with the final decision. This is the case for instance with the transition radiation detector (TRD) which can be pre-triggered by the forward detectors or by the faster time of flight detector (TOF). Another example is the detector BUSY signals that are delaying the CTP final decision. For more accuracy in the trigger sequencing, one needs also to take into account the signal delays in the electronics and cables. All these cases could not be handled by the old trigger framework.

In the new implementation, the scheduling mechanism is managed by a dedicated scheduler class. Every simulated trigger device has a scheduler attached to it which has to be configured to replay the given device. The scheduler holds a list of scheduled entries, representing a group of one or more response functions to be fired sequentially. Any such entry may be assigned a relative start time or priority. The list of scheduled groups can be sorted with respect to these variables or some custom criteria, creating a sequence. The sequence defined for a device is the recipe for replaying the trigger response functions in the desired order. This is detailed in figure 3 which shows a configurable scheduler that holds more than one possible sequence.
Figure 3. Trigger device output response functions are fired by schedulers. One can group outputs like: $G_1(O_1, O_2, O_3)$, $G_2(O_4, O_8)$, $G_3(O_5, O_6, O_7)$. Any device must schedule all groups in at least one sequence. The full trigger device replay is managed by a global scheduler.

The description above applies only for a single trigger device but cannot replay a complex schema with trigger processors that co-operate for giving the full CTP input. To cope with this, the new framework provides a global scheduler that is managing the whole trigger chain. An entire device can become an entry for the global scheduler so all device schedulers can be executed using the sequences mentioned before. In case a given trigger device requires inputs coming from other devices, a sequence can combine any type of scheduled entries – single outputs, groups or devices – so that any possible schema can be accommodated.

3. Implementation of the new ALICE trigger simulation

The usage of the new trigger simulation in the ALICE offline framework is sketched in figure 4. The input for the trigger simulation chain can be raw or digitized data. Currently not all real trigger processor hardware inputs are available in the raw data format, but a detailed simulation of the FPGA algorithms is not always needed. A first level of trigger readers is decoding the trigger inputs from the digital detector output. For example, the trigger reader for the silicon layers of the inner tracking system (ITS) is getting from data the Fast-OR signals which are used by this detector to provide an efficient trigger on track multiplicity [4].

The trigger readers are implemented in the simulation framework as simple device objects taking a single input (the current event) and providing a single output to the client trigger processor objects. In case of the ITS reader an array of 1200 Fast-OR pulses is retrieved in this way.

The next level of devices ($TP_1$, $TP_2$ …) implement the detector specific trigger algorithms as response functions connected to as many outputs as provided by the specific detector trigger to the CTP. To follow the given example, the pixel trigger system (PIT) processes the array of FO signals and provides 10 inputs to CTP.

The ALICE central trigger processor has a more complex software implementation of the trigger decision. The different detector trigger processor inputs are grouped into clusters that fire-up predefined trigger classes [1]. The CTP module and certain TP’s are configurable devices, getting the configuration from a file or a database entry.
Figure 4. The connection between trigger device objects in the offline simulation closely reproduces the actual ALICE trigger system. The inputs for different trigger processors are de-serialized from real or simulated data, while the response of different triggers is simulated as a sequence replayed by the global scheduler, based on the requested configuration. The ALICE central trigger processor is simulated in the same manner and its decision is embedded in the event summary data (ESD) as a special object.

4. Current status and future plans
We have described the new ALICE trigger simulation framework, which is currently in the commissioning phase. We are running extensive tests using a simplified version of ALICE CTP while the current offline description for all trigger detectors are being migrated to the new framework. The plan is to provide a setup for testing and optimizing the ALICE trigger simulation which was using the simplified version for the first year of data taking. Another step in the development is to serialize and store not only the trigger decision but also the inputs and the configuration in the event summary data format [3]. This will allow comparing offline the simulated trigger decision with the actual one for fine-tuning purposes.

After the full validation of the framework, the next step will be to perform detailed trigger efficiency studies using this new tool.

The new ALICE trigger simulation framework was designed and optimized for the specific experiment needs, being implemented as a library within the ALICE offline framework AliRoot. The core classes of the framework offer however a generic approach that is suitable for describing any complex trigger schemas. This framework is quite flexible in accommodating raw or detailed descriptions of trigger detectors and it introduces time as component of the trigger simulation. The framework is likely to evolve while implementing more accurate descriptions for the ALICE trigger detectors, which may introduce extra requirements.

References
[1] ALICE TDR on Trigger, DAQ, HLT and Control system, 2004 CERN-LHCC-2003-062
[2] ALICE: Physics Performance Report, Volume I, ALICE Collaboration et al 2004 J. Phys. G: Nucl. Part. Phys. 30 1668-1673
[3] http://aliweb.cern.ch/Offline/AliRoot/Manual.html
[4] The pixel Fast-OR signal for the ALICE trigger in p-p collisions, 2007 nucl-ex/0703019