Object Oriented simulation of the L1 trigger system of a CMS muon chamber

C. Grandi

Università di Bologna and INFN-Bologna, Italy

Abstract

The level 1 trigger system of the CMS experiment will be implemented using ad hoc designed hardware. It is essential to have a detailed simulation of the system in order to effectively define data analysis strategies. Object Oriented technology is well suited to reproduce the behavour of and the relations among the different electronic devices. We present the design of the code which simulates the behavour of the level 1 trigger system of a CMS muon chamber. The code is fully integrated with the ORCA (Object oriented Reconstruction for CMS Analysis) program. On demand reconstruction is widely used in order to optimize CPU and memory usage, and to provide an user-friendly interface. Persistency in an Objectivity database is introduced in order to save the products of the most CPU consuming processes.

Keywords: CMS,trigger,simulation

1 ORCA, the Object oriented Reconstruction program for CMS Analysis

Since September 1998 CMS has built and deployed an OO reconstruction program under the acronym ORCA. This software is built on the CARF[1] architecture which supplies persistent storage of data, and the framework for reconstruction and analysis. CARF is used to manage the reconstruction utilizing a system of reconstruction on demand. Modules register themselves at creation time and are invoked when required (*implicit invocation architecture*). This is acheaved using the *Dispatcher-Observer* pattern. Management of the data of the different sub-detectors is provided by the sub-systems (Tracker, Calorimetry, Muon, etc...) which implement the algorithms to perform the different steps of the digitization and of the reconstruction. More details on ORCA may be fond in [1].

2 Level 1 Trigger of a CMS muon chamber

Each CMS muon chamber is made up by 3 sets of drift tubes. Each set is composed by 4 layers of tubes (quadruplet). Two quadruplets are sensible to the R-Φ coordinate and the third one to the R-θ coordinate. The first step of the trigger system is the BTI (Bunch and Track Identifier). The BTI finds alignments of hits in each quadruplet and determines the LHC bunch crossing in which the muon was produced, using a mean-timer technique. The second step is the TRACO (TRAck COrrelator) which associates two segments found by the BTI's sensible to the R-Φ coordinate. The last step is the TS (Trigger Server) which selects the two best segments in a chamber and passes them to the Regional Trigger. The segments found by the BTI's in the R-θ quadruplet are sent to the TST (Trigger Server Theta) which packs them in bit streams that are then used by the Regional Trigger to find alignments in the longitudinal view. The R-θ segments are also used by

[1] CARF: CMS Analysis and Reconstruction Framework

the TRACO to validate its R-Φ candidates. Details on the CMS muon detector and muon trigger system may be found in [2].

3 Bit-wise algebra

Bit-wise algebra has been used whenever possible in the simulation of the Level 1 trigger, to be as close as possible to the real electronic circuit behavour. For this pourpose the class `BitArray` has been developed. Negative numbers have been implemented using the usual 2-complement notation.

4 Simulation of the Level 1 Trigger of a CMS muon chamber within ORCA

The code which performs the simulation of the Level 1 Trigger of a CMS muon chamber is in the ORCA Trigger subsystem. The involved packages with their dependencies are shown in figure 1.

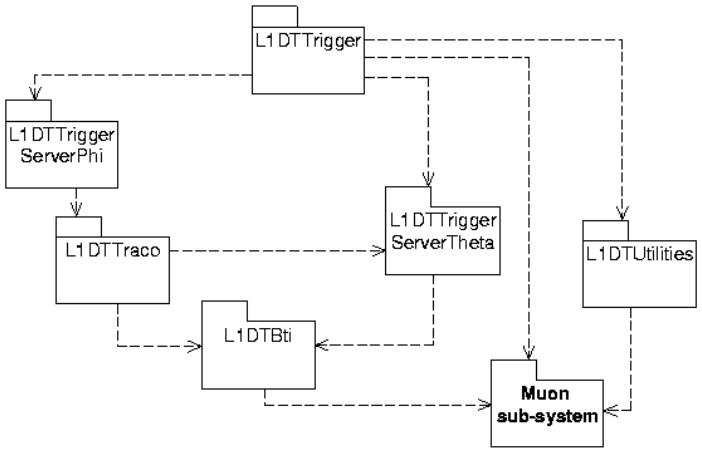

Figure 1: Packages involved in the simulation of the Level 1 Trigger of a CMS muon chamber.

`L1DTTrigger` Interface package. It is responsible for the instantiation of all the objects. It contains the objects which answer the outside world queries.

`L1DTUtilities` Configuration parameters and geometry of the chambers. It keeps the link with the corresponding chamber objects in the Muon sub-system.

`L1DTBti` Simulation of the BTI.

`L1DTTraco` Simulation of the TRACO.

`L1DTTriggerServerPhi` Simulation of the TS (R-Φ view).

`L1DTTriggerServerTheta` Simulation of the TST (R-θ view).

4.1 System setup

The `L1MuDTTrigSetup` object is instantiated by the user as a singleton. It is a `LazyObserver` of `Run` (see [1]). When a new run is read in it instantiates a new `L1MuDTTrig` object, which creates a `L1MuDTTrigUnit` for each muon chamber (`MuBarStat`). Each `L1MuDTTrigUnit` has a geometry (through a link to the associated muon chamber) and all the trigger card objects (`L1MuDTBtiCard`, `L1MuDTTracoCard`, `L1MuDTTSPhi`, `L1MuDTTSTheta`). The cards are `LazyObserver` of `Event` and activate themselves when their output is asked for.

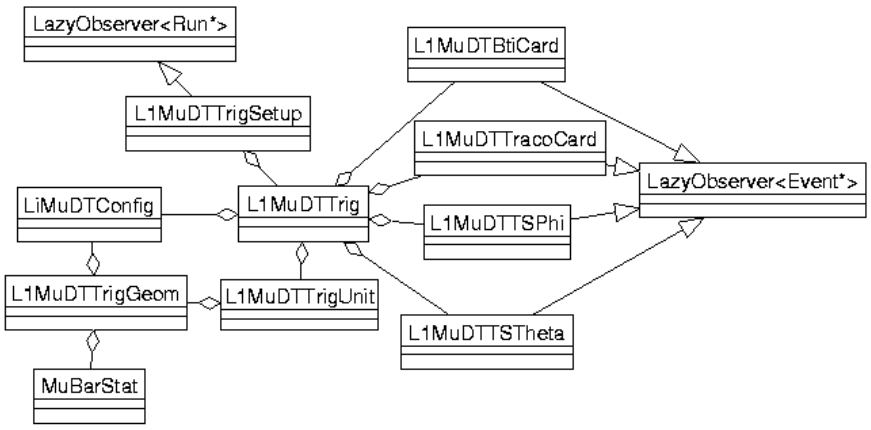

Figure 2: Class diagram for objects instantiated at setup.

4.2 Event processing

The system output is accessed through the `L1MuDTTrig` class. In the normal simulation flow the only use case is given by the Regional Trigger that asks for chamber segments (the output of the Trigger Server). For debugging pourposes methods are available to access the intermediate results from the trigger card objects. In the trigger card objects, the same methods used to pass results to the other trigger components are also used to pass intermediate results to the users for debugging pourposes.

4.2.1 BTI

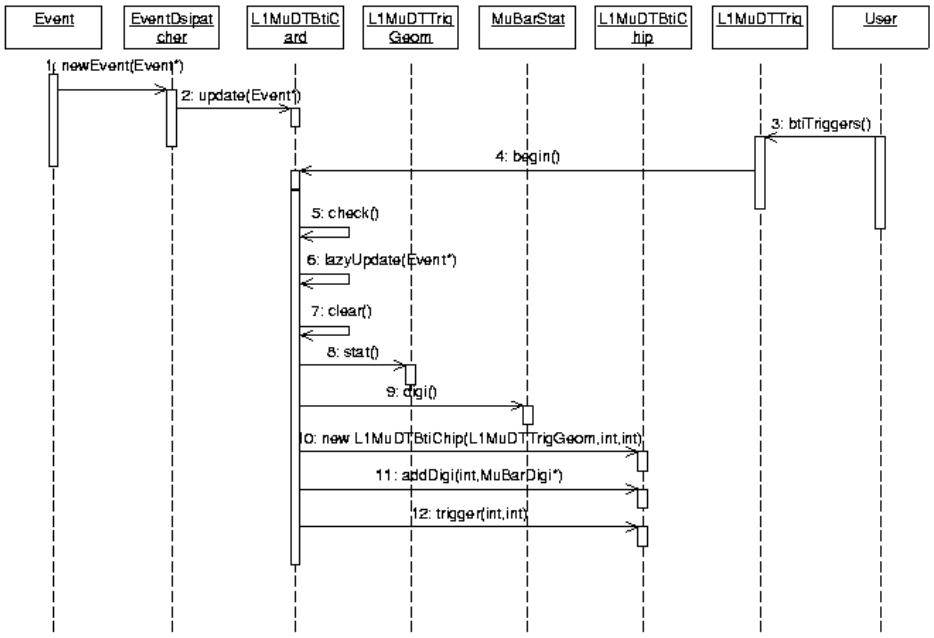

Figure 3: Interaction diagram for BTI simulation.

There are about 450000 BTI chips in the real CMS detector and on average only 35 BTI's have non null data for each muon. It would be a memory waste to instantiate all of the BTI chips at setup time. When the BTI output is asked for, the `L1MuDTBtiCard` checks if the event data has already been analyzed. If not it loads the `MuBarDigi` (i.e. the drift times) from the `MuBarStat` to the appropriate `L1MuDTBtiChip`, which represents the single BTI chip. If the chip is not instantiated yet, the `L1MuDTBtiCard` creates it with the appropriate geometry and configuration. After loading the drift times, all the non-empty `L1MuDTBtiChip` are activated. The output is stored in a cache on the `L1MuDTBtiCard` ready to be delivered. The interaction diagram for the BTI reconstruction is shown in figure 3.

In order to reproduce the mean-timer behavour, the BTI algorithm has to be run several times, simulating the situation in the chip registers as it would appear in the real detector before and after the expected triggering time. In the simulation code this is done 26 times in steps of 25 ns. All the triggers found at the correct and wrong times are stored, with also the triggering time information.

4.2.2 TRACO

There are about 120000 TRACO chips in the real CMS detector. The situation is similar to the one presented for the BTI. The `L1MuDTTracoChip` objects are instantiated by the `L1MuDTTracoCard` only if they have non-null input from the BTI. The algorithm is run 26 times (once every 25 ns) using as input the BTI output found at the corresponding triggering time.

4.2.3 Trigger Server

All the 250 `L1MuDTTSPhi` and 250 `L1MuDTTSTheta` (one of each per chamber) are instantiated at setup time. As for the previous steps the algorithms are run 26 times. The main task of the R-Φ view algorithm is a sorting of the triggers provided by the TRACO chips. Depending on the configuration, the significance of the bits which represent the TRACO output is different. This is acheaved both in the hardware and in the simulation by a re-ordering of the bits in the BTI-data string. The sorting is performed by means of the `operator<` of the `BitArray` class. For the R-θ view the algorithm is basically a packing of the bits representing the output of the BTI chips in the R-θ view, and it is also performed using the tools provided by the `BitArray` class.

5 Persistency

In order to save processing time, it is foreseen that the output of the different steps of the trigger simulation are made persistent in Objectivity/DB databases. At the end of an event processing the cache of each trigger card object is copied to its persistent representation. In reading from the database, simple C++ pointers to the persistent objects are loaded into the cache of the trigger cards. CARF takes care of copying the objects to memory and to keep the pointers valid until the end of the event processing. The use of persistent or transient data is completely transparent to the end user. More details on object persistency in ORCA may be found in [3].

References

1 D.Stickland, "The Design, Implementation and Deployment of Functional Prototype OO Reconstruction Software for CMS. The ORCA project", paper submitted to this conference.

2 CMS collaboration, "The Muon Project Technical Design Report", CERN/LHCC 97-32.

3 L.Silvestris, "An ODBMS approach for the persistency in CMS", paper submitted to this conference.