September 25, 1998

Performance Evaluation of the CMS DAQ System Using Simulations

T.Ladzinski, N.J.Sinanis, S.Tether*
for the CMS DAQ Group
CERN, CH-1211 Geneva 23
***MIT**

Abstract

The data acquisition system of the Compact Muon Solenoid (CMS) experiment must run at the very high trigger rates expected at the LHC. A 512x512 data switch with an aggregate bandwidth of 500 Gb/s is planned as the core of the event builder, which will feed events in at least two stages to the computing farm running the post level-1 triggers in software. Currently small scale hardware setups are built to test various hardware architectures. In order to see how the test bed results apply to the full scale system simulation activities have started.This paper describes the modular discrete event simulation developed for the CMS data acquisition system. Module interfaces are defined as sets of messages, represented as objects that know how to “send themselves” to the correct destination modules. Implementation was done in C++ with the underlying event scheduler from the CNCL class library. Results of first simulation runs are presented and discussed.

To appear in the proceedings of
CHEP ‘98 (August 31 - September 4, 1998) Chicago, Illinois, USA

Performance Evaluation of the CMS DAQ System Using Simulations

T.Ladzinski, N.J.Sinanis, S.Tether*
for CMS DAQ Group
CERN, CH-1211 Geneva 23
{t.ladzinski | n.sinanis}@cern.ch
*MIT, tether@mitlns.mit.edu

Abstract. The data acquisition system of the Compact Muon Solenoid (CMS) experiment must run at the very high trigger rates expected at the LHC. A 512x512 data switch with an aggregate bandwidth of 500 Gb/s is planned as the core of the event builder, which will feed events in at least two stages to the computing farm running the post level-1 triggers in software. Currently small scale hardware setups are built to test various hardware architectures. In order to see how the test bed results apply to the full scale system simulation activities have started.This paper describes the modular discrete event simulation developed for the CMS data acquisition system. Module interfaces are defined as sets of messages, represented as objects that know how to “send themselves” to the correct destination modules. Implementation was done in C++ with the underlying event scheduler from the CNCL class library. Results of first simulation runs are presented and discussed.

Introduction

In the CMS detector the collision rate of Large Hadron Collider (LHC) of 40 MHz will be reduced to 75kHz by the Level-1 trigger system, which is based on specially designed hardware modules. The average data volume generated by the CMS detector is expected to be of the order of 1 MByte per event. To match the capabilities of the mass storage (TByte/day) and off-line computing systems, the final output of the experiment should not exceed few 100 events/s. The proposed Data Acquisition System (DAQ) [1], which will handle this stream of information comprises a number of Readout Units (RU), Filter Units (FU) and an Event Builder (fig.1). The FUs, which shall be in the form of a computer farm will allow for a two-step (or more) event building and event selection.

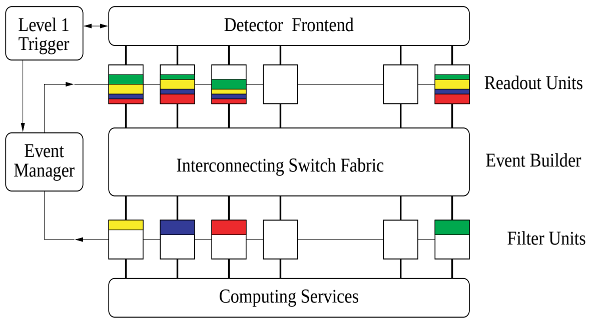

Fig. 1. CMS DAQ basic structure, colour depicts event building process.

In the current design the proposed number of RUs and FUs is of the order of 500. Data corresponding to one event is distributed among all the RUs and it is the main goal of the DAQ system to assemble it in one of the FUs for further processing and possible storage. It is assumed that with today’s advancements in information technology one should be able to find off-the-shelf solutions for the interconnecting network of the DAQ system. There exist a number of different interconnecting technologies providing switching capabilities and CMS is investigating their use. This should be done by integrating commercial network products with the custom designed electronics. Those developments include laboratory test setups, but their results are limited to small cases with event builder size of the order of 2x2 [2]. Tests of bigger systems are foreseen in the near future, yet a complete setup with hundreds of RUs and FUs is certainly not going to be affordable. The study of the behavior of the full system is therefore only possible with simulations. We have designed a simulation package, where each of the main components of the DAQ system is independently developed with the necessary degree of detail. Several implementations of the same module are possible so as to allow for the study of a complete system equipped with different hardware components. Using the basic simulation blocks, a system of any size can be simulated. In the context of evaluating various designs, hardware and software protocols, the simulation package is a basic tool that allows testing of the linearity of the Event Builder. Where possible the parameters of the simulation modules were tuned with data obtained from prototype measurements.

Simulator Design

The flow of information in the CMS DAQ system can best be simulated by sending messages corresponding to data or control signals exchanged between various modules. This observation has led to the design of the simulation package as a discrete event simulator. Furthermore, it is possible to use the event-driven approach as opposed to time-slicing simulations [3], thus allowing for high precision of discrete time clock (nanoseconds). The precedence of events occurring very close to each other can therefore be preserved. Computationally this high resolution is not expensive, as reactions to incoming messages (e.g. transitions between states) occur only when needed, with intervals corresponding to time delays of events happening in the real system that is being modeled - usually of the order of microseconds.
The real system (fig.1) was modeled by four separate modules, which are designed to be and act during the simulation as four separate entities (fig.2). Each module provides an interface for sending/receiving messages to/from other modules and itself. Internally a RU/FU/Switch module may contain arbitrary number of Readout Units, Filter Units or Switch boxes respectively. The dispatch of messages to the right object instance is transparent to modules, which just know the interface protocol. It can be said that messages between modules are self-delivering as the modules need not know who is the handler of the message sent.

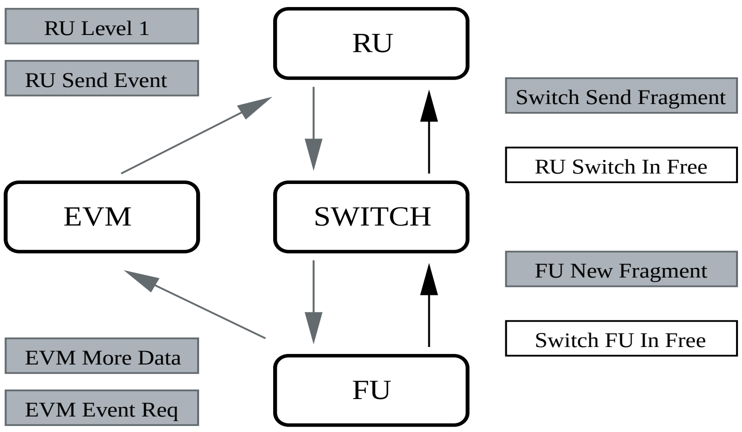

Fig. 2. DAQ simulation module components and inter module messages shown as arrows with colour matching label frames.

The implementation was done in C++, which allowed great flexibility mainly in the level of detail of various models, e.g. when testing RU, the Event Builder switch was reduced to a simple delay line. It was decided not to completely develop the simulation framework but rather to built on top of existing simulation software packages. The Communication Networks Class Library (CNCL) [4] was chosen, as it provides a good core engine for event-driven simulations. Events in the simulation are scheduled through CNCL's heap scheduling mechanism. From the implementation point of view each of the four modules is a sub-class of CNCL's Event Handler abstract base class. A number of other features provided by the library was used: random number generators, lists, FIFOs etc. We have investigated introducing distributed simulation mechanisms to the library. Due to the presence of a closed loop in the DAQ system, timing problems were encountered and as of today we do not intend to continue distributed simulation developments. The final package is a set of libraries. The end-user determines which components are needed and links them to his main program. The size of the system, i.e. the number of RUs and FUs, is set in an initialization file together with other parameters which are briefly presented for each module in the following section.
The Histoscope package [5] was chosen for collecting and displaying data in real time in the form of histograms. The monitoring features of histoscope are valuable for understanding the transient behavior of the simulated systems. Furthermore, the output of histoscope can easily be read by other, more sophisticated data analysis tools (eg.PAW). For further simulation runs it is foreseen to introduce a global n-tuple hold-

ing all the data from a given run which will be useful for finding correlations between the various parameters in the system.

Module Description

Readout Unit

The current simulator uses a simple model providing the basic RU functionality [6], [7]. As the simulation package does not include Front End Driver (FED) or in fact any hardware close to the frontend electronics of the detector, the RU is also responsible for 'data generation', which in fact means drawing a random number to determine the event fragment size for a given trigger in each memory unit. The configuration parameters of this module therefore include the event fragment size distribution.

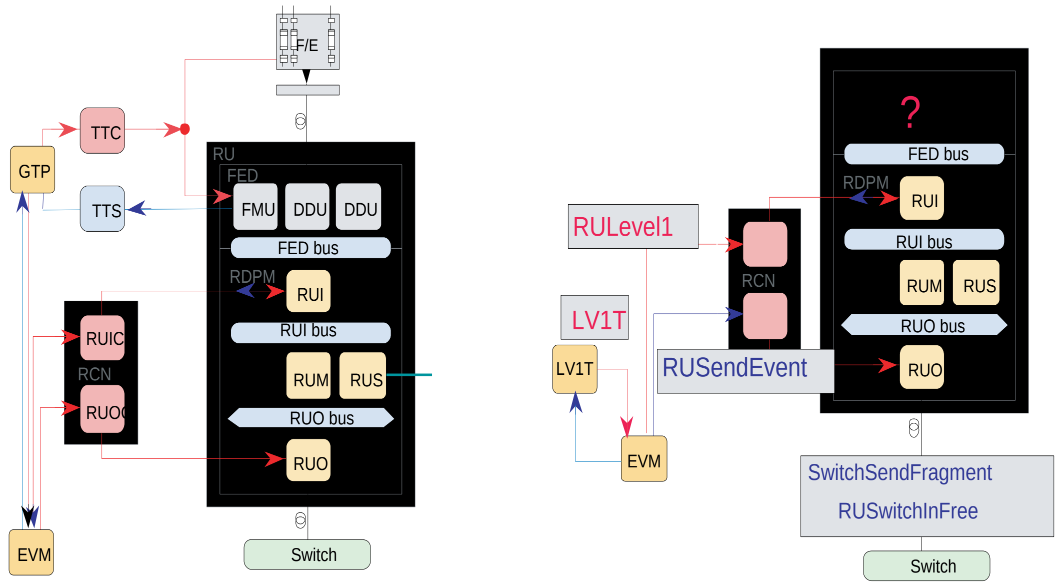

Fig. 3. Readout Unit. Left: actual device, right: simulation model with messages.

The latencies and speeds of the DMA engines in the Readout Unit Input (RUI) and Readout Unit Output (RUO) can be set, as well as the Readout Unit Memory (RUM) page size. Memory usage as well as the length of time an event fragment stays in memory is histogrammed on-line. A more detailed (low level) model of the Readout Dual Port Memory (RDPM) is under development.

Switch

The switch simulation is specific to a particular networking technology. So far, two separate modules have been developed for ATM [8] and Fibre Channel [9]. In each case the corresponding standard describing the routing of packets as well as their size has been followed. Bandwidth, latency and other vendor-specific parameters can be set at initialization; default values corresponding to test bed setups are provided. The switch simulation module is logically interfaced to the rest of the modules at the DMA input/output level. Therefore the Network Interface Cards (NIC) are also modeled within this module. Such a solution allows greater flexibility in switching between various technologies, as every technology-dependent issue is encapsulated in this module and is therefore hidden from the rest of the simulator. The results presented in the next section were obtained using a Fibre Channel (FC) switch module. It simulates the behavior of FC switches operating in three classes of service:
- Class 1 - connection oriented;
- Class 2 - connectionless traffic with acknowledgment of delivery;

- Class 3 - connectionless traffic with no acknowledgments.

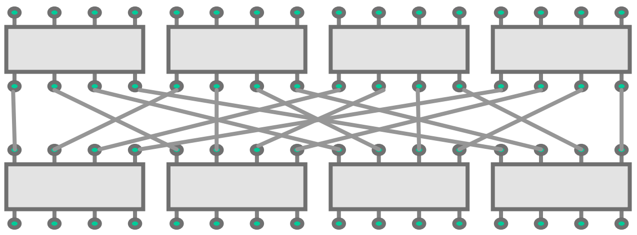
Fig. 4. Example of possible Clos network implementation

The internal design of the simulator allows to study a generic device following the FC standard, but it can also precisely reproduce the architecture of the Ancor MKII 8x8 unit [10]. A switch of any size can be instantiated either as a full crossbar or with an internal bus. Certainly, for large configurations this would have been unrealistic and therefore multi-stage switch fabrics can be constructed using Clos interconnecting networks (fig.4). The list of parameters includes the class of service and acknowledgment mode, the fabric architecture, the NIC memory size, the link bandwidth and a detailed set of latencies for different frame(packet) types.

Filter Unit

The data rates that need to be sustained by the FU are far less demanding than those of RU. Many factors are therefore not as important and the simulated model is less precise in terms of technical details. However, many parameters can be varied to see the influence of various approaches to processing units on the overall performance. The number of Filter Nodes (FN) per FU and the number of CPUs per FN determine the farm architecture. The number of buffers in the Filter Unit Memory (FUM) to be used for event assembly helps study the necessary memory resources that must be provided in the real system. For the moment, the simulator distinguishes between level-2 (LV2) and level-3 (LV3) trigger tasks. They are characterized by functions defining LV2 and LV3 processing times as well as the corresponding rejection factors.

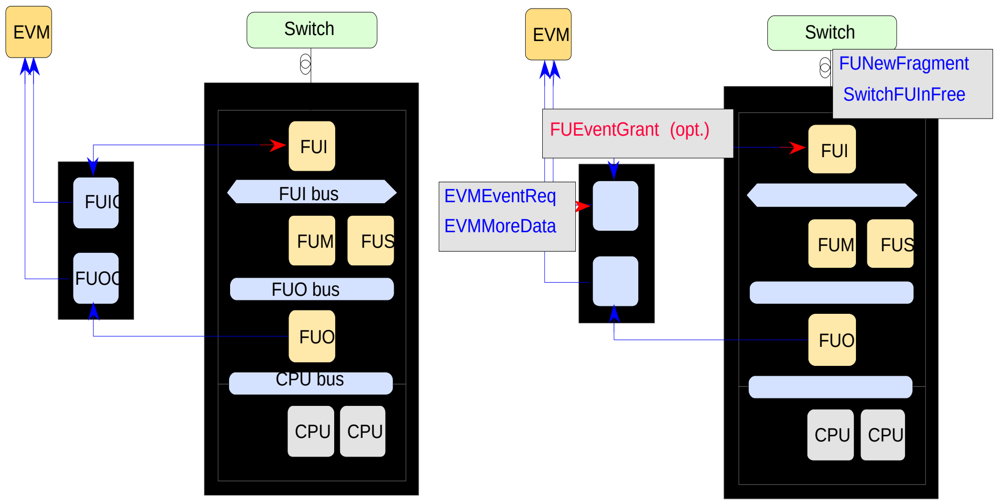

Fig. 5. Filter Unit. Left: actual device, right: simulation model with messages.

Event Manager

The Event Manager (EVM) module includes the functionality of level-1 (LV1) trigger generator. This design was chosen to minimize the amount of inter module messages. Currently, the EVM supports only the basic functionality required by the DAQ architecture. It is a passive device broadcasting requests for data coming from different FUs to the RUs. This module collects statistics of the system throughput, histogramming such information as the sustained LV1 trigger rate, the LV2 and LV3 processing times etc.

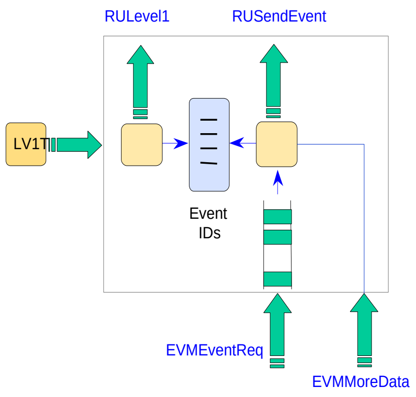

Fig. 6. Event Manager model.

First Results

First runs were done at marginal cases to easily verify the simulation output, which can be predicted by simple analytical calculations. Single step event building was used i.e. LV3 processing was suppressed, while LV2 processing time was set to a gaussian distribution with 2ms mean and a variance of 1ms. Random traffic shaping was used in the RUs. Fibre Channel class-2 of service was used, as it seems most appropriate for event building. Class-3 with no confirmation of delivery would require additional verification algorithms and class-1 adds open-connection overhead. The event fragment size was set to a value of 2kB.
A simple 8x8 system was studied first. This helped set up the necessary histograms and provided valuable insight into the behavior of the simulator. The rate of accepted LV1 triggers and the time LV2 events spent in the system (fig.7) was primarily measured. These results were then crosschecked with histograms of the RU memory usage and the packet travel time data in the switch.

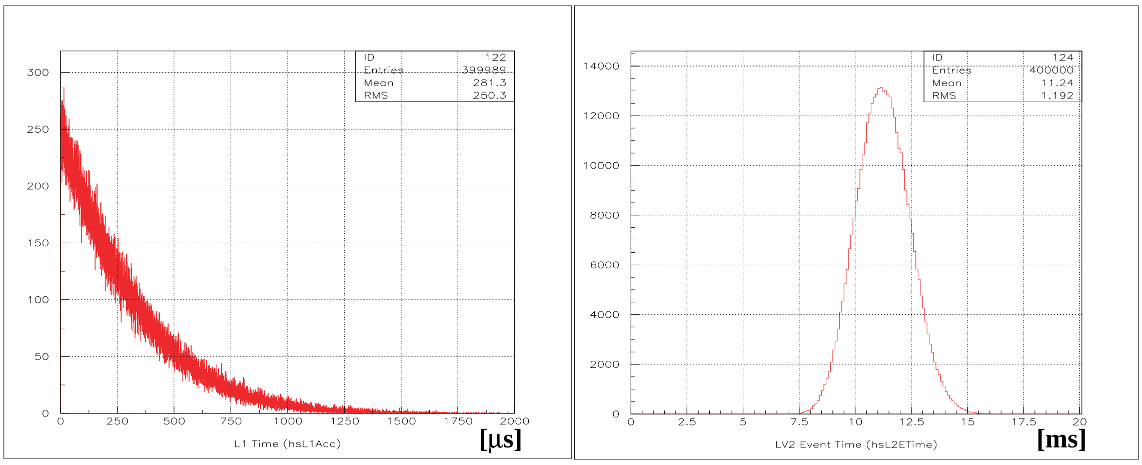

Fig. 7. Left: distribution of intervals between accepted LV1-triggers. Right: distribution of time LV2- events spent in the system, measured from arrival in the RUs to disposal.

A system scaling study was done with simulations of systems consisting of 4x4, 16x16, 32x32, 64x64 and 512x512 event builder size (fig.8). The low performance of small scale systems comes from the fact, that a few FU operating in parallel cannot change the order of magnitude of LV2 processing time. The saturation at approximately 16kHz for large systems is the result of the relatively high latency of today’s data-link components that were modeled. The overheads encountered in FC products, with a nominal data bandwidth of 103MB/s in reality triple the effective transfer time of small data frames i.e. it takes approximately 65µs to transfer 2KB of data as opposed to 20µs that one would expect given the bandwidth parameter.

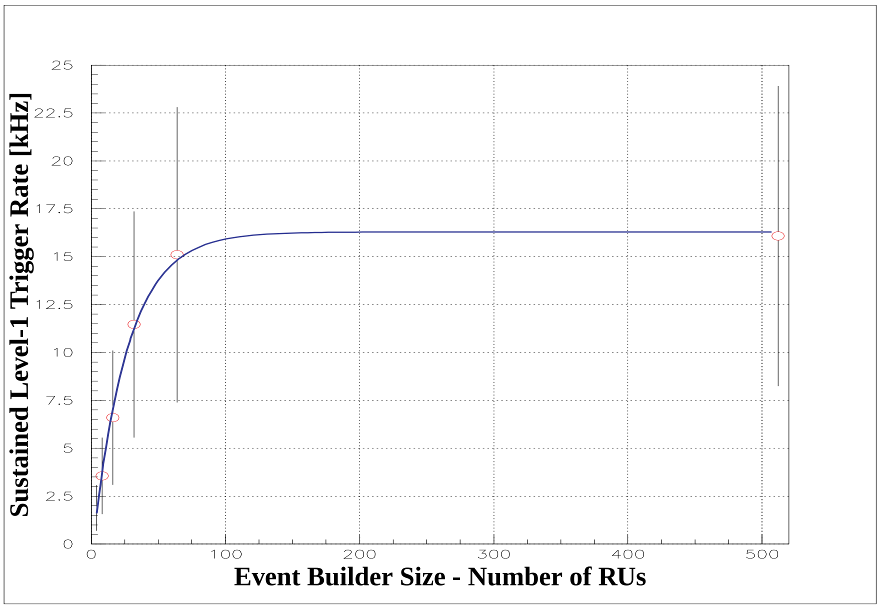

Fig. 8. Event Builder scaling plot, RMS error bars. Nonlinear behaviour and saturation due to high latency of FC network components.

Conclusions and Objectives

A modular simulation system of the CMS DAQ has been developed. First results have been obtained and presented. They show, that we have designed a suitable tool to study DAQ system issues, both in terms of design parameters as well as scaling. Verification with basic speed measurements matches expectations but further tests are planned within the CMS DAQ group to reproduce the exact behavior of a real 4x4 event builder demonstrator. Despite our efforts to minimize the inter-module messages large scale system simulations can take a considerable amount of time - depending on system size 1 second of DAQ time takes from 5 to 56 minutes CPU time (SUN Ultra Sparc 300MHz). Nevertheless, we believe that the basic simulation framework was well designed and the tool will play an important role in the final design of CMS DAQ system. It is planned to add a more detailed version of RU module as well as to expand the EVM in order to test various event building protocols. Additionally, as new networking technologies are being taken into consideration, new switch modules will be added to the library.

References

[1] C*MS Technical Proposal*. CERN/LHCC 94-38, December 1994

[2] T.Ladzinski *et al*. *CMS Data Links and Event Builder Studies*. 3rd Workshop On Electronics for LHC Experiments, London, September 1997.

[3] P.Fishwick *Simulation Model Design and Execution*. Prentice-Hall 1995

[4] http://www.comnets.rwth-aachen.de/doc/cncl.html

[5] http://www-pat.fnal.gov/histo_doc/histo_ug.html

[6] S.Cittolin *et al*. *Dual Port memories in LHC Experiments.* CMS-RD12 TN 95-04, March 1995

[7] P.Sphicas *et al.* http://cmsdoc.cern.ch/ftp/distribution/tridas/4.Talks/RUnextstep.pdf

[8] ATM Forum Specifications, http://www.atmforum.com

[9] Fibre Channel Standard, ANSI X3T11, ANSI X3.230(FC-PH Rev.4.3) http://www.fibrechannel.com

[10] http://www.ancor.com