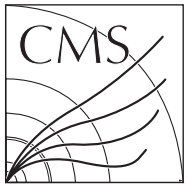
Available on CMS information server

CMS CR -2014/360



The Compact Muon Solenoid Experiment

Conference Report

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland

29 October 2014 (v2, 01 November 2014)

A simulation framework for the CMS Track Trigger electronics

Christian Amstutz, Guido Magazzù, Marc Weber, Fabrizio Palla

Abstract

A simulation framework has been developed to test and characterize algorithms, architectures and hardware implementations of the vastly complex CMS Track Trigger for the high luminosity upgrade of the CMS experiment at the Large Hadron Collider in Geneva. High-level SystemC models of all system components have been developed to simulate a portion of the track trigger. The simulation of the system components together with input data from physics simulations allows evaluating figures of merit, like delays or bandwidths, under realistic conditions. The use of SystemC for high-level modelling allows co-simulation with models developed in Hardware Description Languages, e.g. VHDL or Verilog. Therefore, the simulation framework can also be used as a test bench for digital modules developed for the final system.

Presented at *TWEPP 2014 TWEPP 2014 - Topical Workshop on Electronics for Particle Physics*

Preprint typeset in JINST style - HYPER VERSION

A simulation framework for the CMS Track Trigger electronics

C. Amstutz[a]*, G. Magazzù[b], M. Weber[a] and F. Palla[b]

[a] *IPE – Karlsruhe Institute of Technology,*
Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany
[b] *INFN - Sezione di Pisa,*
Largo B. Pontecorvo 3, 56127 Pisa, Italy
E-mail: christian.amstutz@kit.edu

ABSTRACT: A simulation framework has been developed to test and characterize algorithms, architectures and hardware implementations of the vastly complex CMS Track Trigger for the high luminosity upgrade of the CMS experiment at the Large Hadron Collider in Geneva. High-level SystemC models of all system components have been developed to simulate a portion of the track trigger. The simulation of the system components together with input data from physics simulations allows evaluating figures of merit, like delays or bandwidths, under realistic conditions. The use of SystemC for high-level modelling allows co-simulation with models developed in Hardware Description Languages, e.g. VHDL or Verilog. Therefore, the simulation framework can also be used as a test bench for digital modules developed for the final system.

KEYWORDS: Particle tracking detectors; Data acquisition concepts; Trigger concepts and systems (hardware and software); Simulation methods and programs.

*Corresponding author.

Contents

1. Introduction

The increased luminosity of the Large Hadron Collider (LHC) for the Phase-II Upgrade will lead to an increase in simultaneous collisions of protons at the interaction point of the Compact Muon Solenoid experiment (CMS) [1]. The current trigger system of CMS is unable to cope with the anticipated collision rate. Hence, tracks of charged particles in the silicon tracker must be considered by the first level trigger to take reliable trigger decisions [2].

Figure 1 shows the CMS Track Trigger within the context of the global CMS trigger system. The CMS Track Trigger receives its data from the outer tracker which consists of silicon sensor modules. These modules are arranged in six layers in the barrel section which is complemented by two endcaps. The barrel and the endcaps are built of the same types of modules [3, 4]. The modules consist of two closely silicon sensors which enable the determination of the transversal momentum of charged particles as is illustrated in figure 2. The modules in the three outer layers of the tracker are built of two silicon strip sensors and are called *2S Modules*. To achieve a higher spatial resolution, in the inner three layers the *PS Modules* are used instead where one strip sensor is replaced by a macro-pixel sensor.

The CMS Track Trigger searches for tracks of high transversal momentum within the silicon detector data and forwards this information to the first level trigger. The first level trigger takes into account the data from other CMS subdetectors and triggers the read-out of the data buffers of all CMS subdetectors if an interesting event has been found.

The total incoming data rate of the CMS Track Trigger is in the order of 50 Tbit/s, the processing of all the data must take less than 12.5 µs. These numbers imply that the future system is

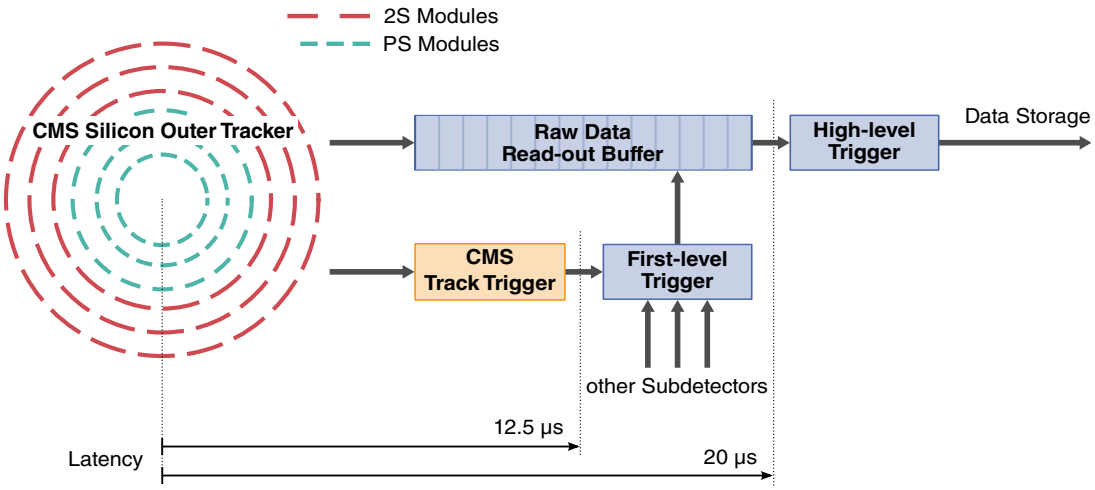


Figure 1: Block diagram of the CMS trigger system.

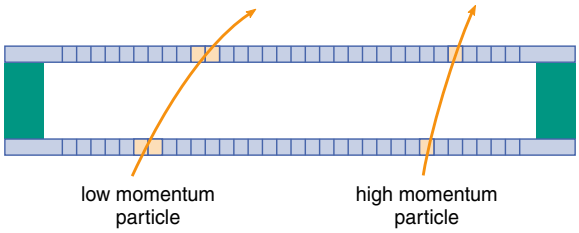


Figure 2: Two particles crossing the two silicon layers of an outer tracker module.

the most challenging digital processing system ever conceived and will be cutting-edge in all of its features.

While it is obvious that general purpose processors can not be used and that Field-Programmable Gate Arrays (FPGA) will be the main computational devices, as of today it is uncertain how such a system should best be built. One suggested system architecture is based on the partitioning of the silicon tracker data into sectors and processing the data of each sector in a separate processing unit. The processors would be time-multiplexed to decrease the latency [5]. Whereas, another proposed architecture is in its entirety time-multiplexed and the data of one event from the whole detector is sent to a single processing unit. Multiple processing units process the data from different events [6].

How should an efficient system that fulfils the requirements be built? Constructing a demonstration system is time-consuming and expensive. This is where the simulation framework described in this document comes into play, as it is a tool that allows us to simulate an essential portion of any proposed CMS Track Trigger architecture. The goals of the simulation framework are the quick evaluation of different system architectures and the evaluation of its performance. Critical figures of merit are for example latencies of the whole or a part of the system, required sizes of data buffers and data link bandwidths. In preceding and complementing a demonstration system, the simulation framework provides the direction system development should take to alleviate possible bottlenecks identified by the hardware approach.

2. Concept of the simulation framework

To allow a large portion of the CMS Track Trigger to be simulated, we chose to model the simulation framework on a high abstraction level. The model includes just enough details to obtain accurate results to take decisions on the system architecture of the CMS Track Trigger, but as few details as possible to keep the simulation time short. This is particularly important because the CMS Track Trigger is a large system. It is indeed not always necessary to simulate the complete system, but as the input data rates from different detector regions vary, it is often essential to simulate a substantial part of a proposed trigger system. The input data itself comes from Monte Carlo simulations of the detector, i.e. the varying input data rates from different modules represent realistic operating conditions.

The CMS Track Trigger is modelled by splitting it into functional blocks. Each of the blocks represents a logical part of the CMS Track Trigger's functionality, but does not necessarily correspond to a single hardware unit. Especially in an FPGA, different functional blocks can be implemented within the same device. All functional blocks are modelled as generically as possible to allow us to change block parameters easily, e.g. the lengths of FIFOs. How detailed a certain block is implemented depends on the effect of this block on the simulation, i.e. if a part of a block has no effect on the figures of merit to be evaluated then the part is left out. For example, JTAG debugging of an integrated circuit is not implemented within the framework. The same applies to the connections between blocks. Only the signals necessary for the functionality itself are modelled within the simulation framework, i.e. signals such as a reset are not included if they have no effect on the figures of merits of interest. For signals of which the concrete implementation is known, a block is added to model the properties of the physical link. For example, the latency of the serial links between the detector modules and the track trigger is known, and a delay is added in the simulation framework.

2.1 SystemC

The main modelling language of the simulation framework is SystemC which is a C++ library extending the features of C++ with the possibility of hardware simulation [7, 8]. SystemC is mainly used for simulations on system level whereas the traditional Hardware Description Languages (HDLs) such as VHDL and Verilog are mostly used for the description of single hardware modules. SystemC is used for the simulation framework as it allows higher simulation speeds compared to HDLs. There are two reasons for that. Firstly, the model can be built on a higher abstraction level and therefore fewer details must be simulated. Secondly, the SystemC code is compiled into an executable which runs directly on the processor and runs not within a simulator. As SystemC is based on C++, code from a huge number of libraries can be used within the model. For example, a FIFO buffer could be modelled using the data structure *queue* of the C++ standard library which consists of tested code of high performance. In contrast to the HDLs, SystemC is not readily synthesizable and it needs to be translated before it is used as firmware.

The blocks of the simulation framework could either be modelled in SystemC or an HDL such as VHDL or Verilog. ModelSim® is capable of simulating blocks written in SystemC and HDL along each other, which is called co-simulation [9]. Co-simulation enables the utilization of the

simulation framework as a test environment for algorithms or as a test bench for HDL code. These two functions are described in the following sections.

2.2 Algorithm evaluation within the framework

When the idea for a new algorithm for one of the functional blocks in the simulation framework arises, this algorithm can be directly evaluated within the simulation framework. Firstly, this allows to check if the complete system still behaves correctly with the new algorithm. Secondly, the influence of an algorithm on system properties, such as latencies and transmission rates, can be studied within the simulated system. For example, a technique proposed to be used within the CMS Track Trigger is time-multiplexing of functional blocks to increase the throughput, which means that data from one time slot is assigned to one block and data from the next slot to another block. Different time-multiplexing policies can be tested by simply exchanging the algorithm.

As the simulation framework is based on SystemC, algorithms written in C++ are easily included. However, it is also possible to evaluate algorithms written in other programming languages if a library exists to include code of that language within C++. For example, Python code is easily called by the library (python.h) that comes together with Python environment [10].

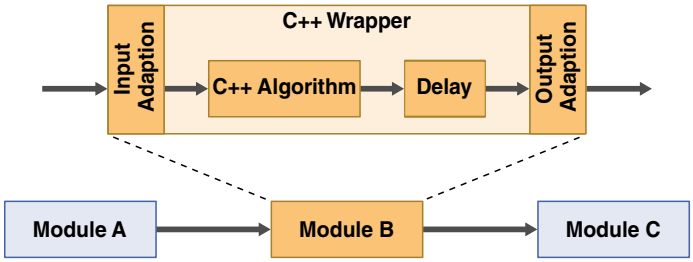
For cases in which the algorithm replaces a complete functional block in the simulation framework, a wrapper is required. This is shown in figure 3 (a), where the functionality of *Module B* written in SystemC is replaced by a C++ algorithm. A wrapper is a block written in SystemC which applies the data coming from the input ports of the block to the algorithm and writes the results of the algorithm to the output ports. On top of this, the wrapper emulates those properties the block will have after its implementation in hardware. For example, the output signal can be delayed by a number of clock cycles to model the expected latency of the block.

2.3 Use of the simulation framework as a testbench

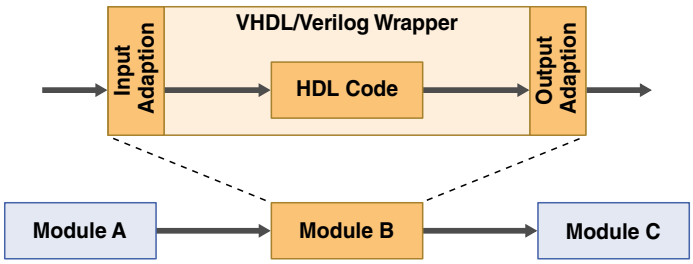
The possibility to run co-simulations of SystemC and HDLs in Modelsim® allows the simulation framework to be used as a test bench for FPGA firmware. In this case, the behaviour of the firmware is tested within an environment that models the complete CMS Track Trigger system. The mechanism is shown in figure 3 (b) and is basically the same as described in the previous section (Section 2.2) on algorithm evaluation. A SystemC wrapper is programmed around the HDL code. This wrapper is necessary to adapt the data types of the inputs and outputs of the firmware to the data types provided by SystemC. Additionally, the wrapper provides signals that do not exist within the framework, e.g. a reset signal is not necessary in a functional simulation but exists in the final firmware block.

3. Implemented system structure

The implemented structure of the CMS Track Trigger is based on the splitting of the detector into sectors as proposed in [5]. Because one sector is inherently a very large system, a simplified sector called *slice* is implemented within the simulation framework. A slice consists of a stack of six detector modules, one for each layer of the detector in the barrel region, and its data processing. The structure of one slice is shown in figure 4. The reduced system size speeds up the development and simulations of the simulation framework. Nevertheless, the implemented slice allows to test



(a)



(b)

Figure 3: *(a)* Evaluation of a C++ algorithm by exchanging Module B with a wrapper which includes the algorithm and models some delay. *(b)* The use of the simulation framework as a test bench for VHDL or Verilog code. Module B is exchanged by a wrapper containing a HDL module.

the functionality and communication of all system components and to test the interaction between slices, an array of 2x2 slices is implemented.

The model is split into a front-end and a back-end. The front-end contains the modules that are part of the silicon tracker itself. The functional blocks which represent the CMS Track Trigger processor are combined in the back-end and will be installed in cabinets in the experiment's cavern. One slice is made by six front-end modules, one for each layer of the silicon tracker, and one back-end module, that collects hits from the front-end and routes them to the track finding processors based on Associative Memories (AM) housed on so-called AM boards. The back-end also collects the recognized tracks (roads) and stores them in output files.

3.1 The front-end

The front-end consists of modules which model the behaviour of the detector modules in the CMS outer tracker. A separate model for both, the 2S Modules and the PS Modules exists within the simulation framework. One slice in the simulation framework contains six sensor modules, three of each type. In general, the functional composition of the two module types is identical. They differ only in the components that are used for the module. Each module contains eight read-out chips. This is half the size of the modules used in the detector which consists of sixteen chips. The hit data is generated

On the PS Modules, the macro-pixel sensors are read out by the Macro Pixel ASICs (MPA), which also merges the macro-pixel data with the data from the strip sensor. On the 2S Modules, the sensors are read out by the CMS Binary Chips (CBC). In the simulation framework, the model

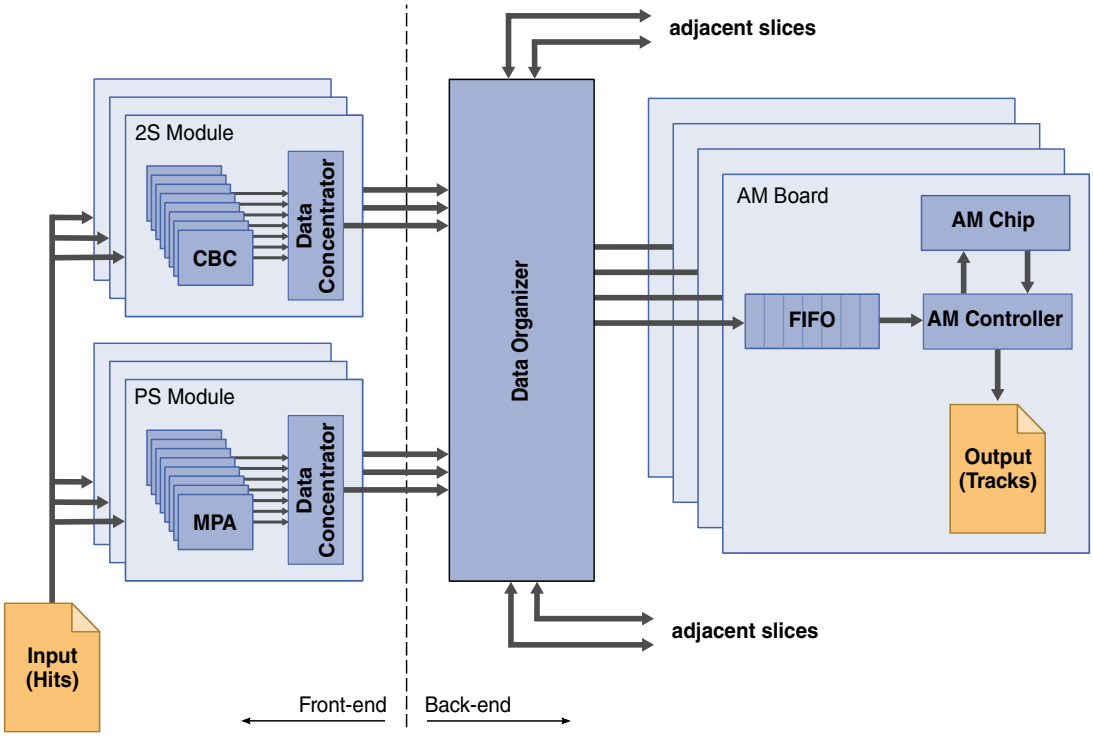


Figure 4: One slice of the CMS Track Trigger as implemented in the simulation framework.

of these two chips act as the entry point for the data. They read the hits from the file at the corresponding simulation time and output the data in the correct format for the simulation. The MPA chips transmit up to four acquired hits in two consecutive events, and the CBC chips transmit up to three hits each event. If there are more hits than the ones that could be transmitted they are discarded.

Each front-end module also contains one Data Concentrator chip which collects data from eight read-out chips and reconstructs additionally the correct timing of the incoming hits for the PS Modules. The hits of eight consecutive clock cycles are buffered in the Data Concentrator and then up to ten/twelve hits of them are transmitted to the back-end. The transmission will be realized by high-speed serial links. Since the details of the serial links do not influence the functionality of the CMS Track Trigger, they are treated as black-boxes in the simulation framework and only the latency which the links add to the signal is simulated.

3.2 The back-end

In the back-end the Data Organizer receives data from the front-end and it reconstructs the timing of the received hits, up to ten in one clock cycle for PS Modules and up to twelve in one clock cycle in 2S modules. To detect tracks that cross the border of a detector slice, the Data Organizer also exchanges data with the adjacent sectors. When all the hits belonging to one sector are gathered, the Data Organizer formats the data for the AM boards. Finally, the hits are written to the AM board to separate FIFOs for each layer.

The next step in the track generation is the pattern recognition by AM Boards. To increase the throughput, four AM Boards exist which are time-multiplexed by the Data Organizer. The implemented time-multiplexing algorithm is round-robin, i.e. in every clock cycle the hits are transmitted to another board and after the last board the hits go to the first board again.

The central part of the AM board is the AM controller. It fetches the hits belonging to an event from each FIFO and transmits them to the AM chip. The AM chip model contains patterns which represent the high transversal momentum tracks of interest. The hits fed to the AM chip are scanned for the existence of these patterns. When a pattern is found, its number is returned to the AM controller. The AM controller stores the pattern number together with the time stamp in an output text file. When the processing has finished, the AM controller fetches the next set of hits belonging to the next unprocessed event.

3.3 Tuning parameters of the simulation framework

The generic implementation of the blocks in the simulation framework allows the easy change of their parameters and therefore the evaluation of the influence of these parameters to the system performance. Already within the relatively small simulation setup of a single slice as presented in this document, several parameters can be modified. These block parameters are:

- Read-out chip output bandwidth, i.e. maximum number of hits transmitted by each read-out chip: currently 3 per event from the CBCs, 4 hits in 2 consecutive events from MPAs;
- Data Concentrator output bandwidth, i.e. maximum number of hits transmitted by each Data Concentrator: currently 10 hits in 8 consecutive clock cycles from PS Modules, 12 hits in 8 consecutive events from the 2S Modules;
- Number of AM boards attached to each Data Organizer, currently 4;
- Size of the FIFOs on the AM boards, currently 1024 hits;
- Number of patterns stored in each AM chip, currently 1024.

With the modelling of larger systems out of the simulation framework's functional blocks, more parameters will be available to configure the architecture. Beside the general structure of the architecture, the parameters will be the most important results that will be obtained from the simulation framework.

4. Conclusion

A simulation framework for the CMS Track Trigger has been developed in order to evaluate proposed system architectures. On the basis of the current status, modules have been programmed in SystemC and VHDL which models the functionality of the CMS Track Trigger. A test system of an array of 2x2 slices has been programmed. To use input data from physics simulations, an exact replication of the detector would be necessary. Therefore, the tests of the simulation framework are performed with input files containing artificial tracks. The expected results of these tracks are known and are compared with the tracks found and written to the output files by the simulation

framework. The simulations that have been run verified the correct functionality of the simulation framework's building blocks.

The blocks of the simulation framework are now ready to be used as building blocks for modelling larger parts of the CMS Track Trigger and evaluate them. The obtained values will be arguments in the discussion about the right system architecture for the CMS Track Trigger. Hence, we showed that it is feasible to create a high-level model of detector read-out electronics. A similar framework will be a valuable tool to consider for future experiments, when decisions on the system architecture need to be taken in an early stage of the planning.

References

[1] The CMS Collaboration, *The CMS experiment at the CERN LHC*, 2008 *JINST* **3** S08004.

[2] D. Abbaneo, *Upgrade of the CMS Tracker with tracking trigger*, 2011 *JINST* **6** C12065.

[3] N. Pozzobon, *Development of a Level 1 Track Trigger for the CMS experiment at the high-luminosity LHC*, *Nuclear Instruments and Methods in Physics Research Section A* **732** (2013).

[4] D. Abbaneo et al., *A hybrid module architecture for a prompt momentum discriminating tracker at HL-LHC*, 2012 *JINST* **7** C09001.

[5] J. Olsen et al., *A full mesh ATCA-based general purpose data processing board*, 2014 *JINST* **9** C01041.

[6] G. Hall et al., *A Time-Multiplexed Track-Trigger architecture for CMS*, Proceedings of *Workshop on Intelligent Trackers 2014*, (in print).

[7] W. Müller et al., *System C - Methodologies and Applications*, Springer, Dordrecht 2003.

[8] D. C. Black et al., *SystemC: From the Ground Up - Second Edition*, Springer, New York 2010.

[9] *ModelSim SE User's Manual - Software Version 10.1b*, p.632-662 Mentor Graphics Corporation, Wilsonville, 2012.

[10] Python/C API Reference Manual, https://docs.python.org/2.7/c-api/index.html (accessed Oct 20, 2014).