A programmable 10 Gigabit injector for the LHCb DAQ and its upgrade

V. Delord[a,b], J. Garnier[a], N. Neufeld[a]

[a] CERN, 1211 Geneva 23, Switzerland

[b] ISIMA, 63173 Aubière, France

vincent.delord@cern.ch, jean-christophe.garnier@cern.ch, niko.neufeld@cern.ch

Abstract

The LHCb High Level Trigger and Data Acquisition system selects about 2 kHz of events out of the 1 MHz of events, which have been selected previously by the first-level hardware trigger. The selected events are consolidated into files and then sent to permanent storage for subsequent analysis on the Grid. The goal of the upgrade of the LHCb readout is to lift the limitation to 1 MHz. This means speeding up the DAQ to 40 MHz. Such a DAQ system will certainly employ 10 Gigabit or technologies and might also need new networking protocols: a customized TCP or proprietary solutions. A test module is being presented, which integrates in the existing LHCb infrastructure. It is a 10-Gigabit traffic generator, flexible enough to generate LHCb's raw data packets using dummy data or simulated data. These data are seen as real data coming from sub-detectors by the DAQ. The implementation is based on an FPGA using 10 Gigabit Ethernet interface. This module is integrated in the experiment control system. The architecture, implementation, and performance results of the solution will be presented.

I. Introduction

The LHCb experiment [1] is currently using a partition dedicated for tests, using a data-flow generator [2] [3]. It gets simulated data from an on-site storage, formats them to the Online protocol and sends them to the High Level Trigger (HLT) [4] farm. The entire Online and Offline systems can be tested this way during LHC shutdown periods, and even in parallel of normal activities.

The project presented in this paper is related to the LHCb upgrade project, and comes mainly from two requirements. The data acquisition (DAQ) [5] system relies currently on Gigabit Ethernet. Its rate is about 35 $\mathrm{GB/s}$. The average size of an event is 35 kB, the event rate is 1 MHz. The upgraded detector aim to reach the full readout speed at 40 MHz. The upgraded DAQ will likely use 10 Gigabit Ethernet (GBE) or Infiniband. The HLT farm processes these events and produces an output rate of 2 kHz.

The idea is to provide a new solution which would be integrated into the system like a real readout board. It would behave like a readout board, except that it would get simulated data from a storage system instead of the physics data from the detector. It is however a long term R&D project and it would be interesting to include this test device in the current DAQ configuration.

A first design is presented in this paper. Sec. II. presents the study and the specifications of the project. Sec. III. presents the main ideas and technologies which manages each part of the system. Sec. IV. discusses about the current limits and the next steps in the design.

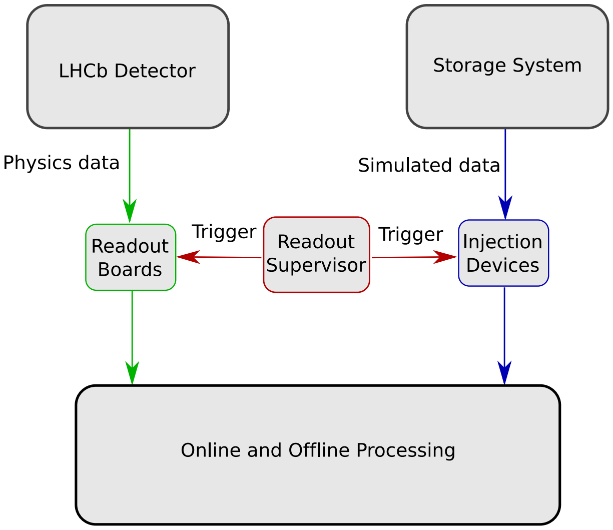

Figure 1: The LHCb data acquisition system and its main data-flows. Both are triggered and controlled the same way. The only difference is the source of the physics events.

II. Specifications

A. Aims

The aims of the test device "injector" are:

- To provide a data-flow identical to the normal data-flow coming from the detector and the Readout Boards [6]. It means that it has to send network frames as if they were coming from the Readout Board layer, faking the IP addresses [7] and other informations.
- This data-flow has to be complex enough in order to be used for trigger and Offline tests. The simulated data-flow is usually represented by several files of ten million events. The average size of an event is 35 kB.
- To be integrated into the DAQ as a Readout Board. It means to be the connected to the Readout Supervisor (Timing and Fast Control, TFC) [8] and to be triggered by it.

- To be integrated into the Experiment Control System (ECS) [9] .
- To be used in parallel with every other LHCb activity.

In the end, the architecture shown in Figure 1 would provide two identical data-flows. One will be dedicated to physics analysis, while the other one will be used for large scale tests.

As the project is in its very first stages, and related to the parallel on-going LHCb upgrade, it has some specific aims. For the design period of the upgraded DAQ architecture, it would be interesting to use this injection device as a pattern generator. Since the protocol that an upgraded DAQ will use is not define yet, it is interesting to have a modular architecture for the injector so we could perform tests using the current Multi-Event Packet (MEP)[10] protocol, or using the Transport Control Protocol (TCP) [11].

B. Analysis

In order to get a high data rate injection, this device will be first studied with a 10 GbE interface. Using a single 10 GbE Injector would allow to get a 35 kHz rate. Our aim is to provide an input rate high enough for the HLT farm to perform event selection, i.e. greater than 2 kHz. This is therefore already much faster, and it would be possible to use several injection devices to increase this rate to reach the real one. Driving a $10\ GbE$ network interface is quite limited using commodity hardware [12]. Indeed reaching the line rate requires at least one CPU entirely dedicated to drive the interface. Processing the events is also quite heavy.

The main task is to read simulated data, to process them lightly before to format them to the networking protocol, according to the trigger information coming from the readout supervisor. It can be achieved using a pipelined architecture, with different stages for each part of the processing: reading, formatting, encapsulating, sending (as shown in Figure 2).

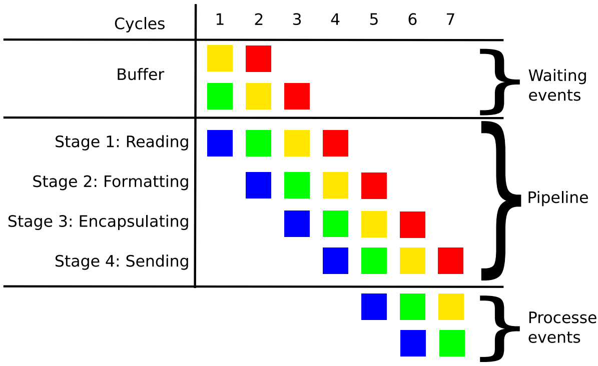

Figure 2: 4-stage pipeline processing events independent from each other.

The Readout Supervisor uses the Time, Trigger and Control (TTC) [13] interface to distribute information over all Readout Boards and over Injection Devices. It is required to process these information in Real-Time and to be always synchronised, with the supervisor and with peer injectors. This means that we cannot suffer from a delay caused by reading the simulated event or from the access to the network interface.

It has been decided, in order to meet all the requirements the best as possible, to implement the injector on a hardware setup, based on a Field-Programmable Gate Array (FPGA). An hardware development is indeed the best solution to process the Readout Supervisor triggers. This promises better performances processing data, and driving the 10 GbE interface.

An Altera PCI development board, based on the Arria GX FPGA, was chosen for a preliminary implementation. It is featuring an High-Speed Mezzanine Connector (HSMC) which allows us to interface various types of connectors for the 10 GbE and the TTC interfaces. This board will not reach the 10 GbE line rate. It is used for proofs of concept, for preliminary implementation and tests. The next version will very likely use an Altera Stratix family FPGA, in order to drive as efficiently as possible a Small Form Factor Pluggable Transceiver (SFP+).

According to all these choices, Figure 3 presents schematically the architecture of the hardware data injector. The design has to be modular, so we could easily replace a core by another one. This would be used mainly on the layer 4 networking core, to address the specifications, and for the storage access, as this part is still under study and it would be interesting to compare several solutions.

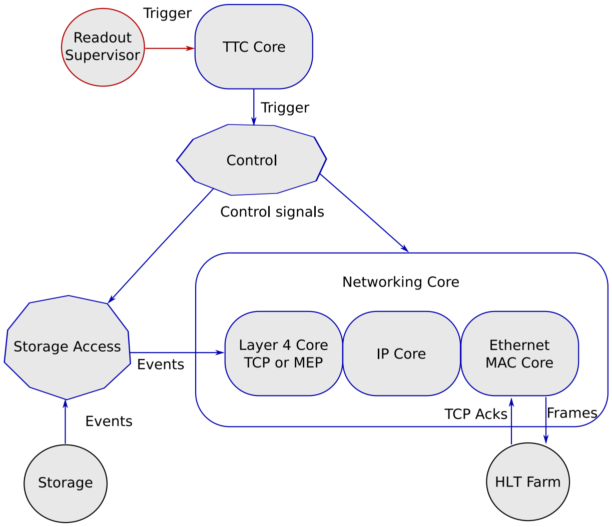

Figure 3: Architecture of the FPGA.

III. Implementation

The architecture of every core follows a generic scheme shown in Figure 4. It consists of a Control Unit, which is a Finite State Machine, and a Processing Unit. The Control Unit generates signals to trigger actions in the Processing Unit. The processing unit implements memories, registers and computing units in order to process the data-flow.

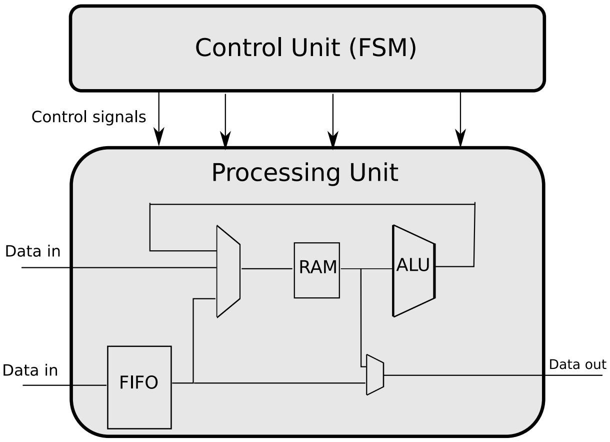

Figure 4: Generic model of a core.

This section presents the implementation of the networking layer, and the investigations for the storage access layer and the integration into the experiment control system.

A. *Networking Implementation*

The current network stack in LHCb is MEP over IP over Ethernet. MEP is a kind of User Datagram Protocol (UDP) [14] which is limited in features. With this device, we would therefore like to test other protocols over IP. We are considering to use the TCP protocol for the upgraded DAQ. It would provide flow control and would ensure that no data are lost over the network. IP and Ethernet cores will be always used.

The idea is to implement one core per protocol, and to connect them in a pipeline. All modules are therefore working in parallel and producing a stream of packets on the network interface.

A licensed Intellectual Property manages the 10 GbE Ethernet Media Access Control (MAC). On top of it, the IP and MEP cores were developed. A particularity of our design is that the IP core is custom. It does not include the IP fragmentation process, and it is only performing data sending. We can afford this only in the case of the MEP protocol, as we need only to send data, not to receive them. The fragmentation is performed in the output of the MEP module. These non-respect of the standard allows the minimization of the resources used by the system in the case of the MEP transport protocol, as it requires less memory usage. The complete frame (header and physics data) is indeed cut while it flows out from the MEP core, as shown in Figure 5, so the IP core input frame length is always lower than the maximum size. The IP core requires however a few more signals to manage the fact that the incoming frame IP headers need to have consistent information.

In the case of the incoming TCP integration, we will manage the IP fragmentation in the output of the TCP core. Then we will need one more IP module, dedicated to the reception of data. These data will mainly consist in TCP acknowledgement packets. Indeed receiving data, even small packets, requires to implement the IP reassembly.

Our network architecture will use, for each protocol, one core dedicated to sending data and one core dedicated to receiving data.

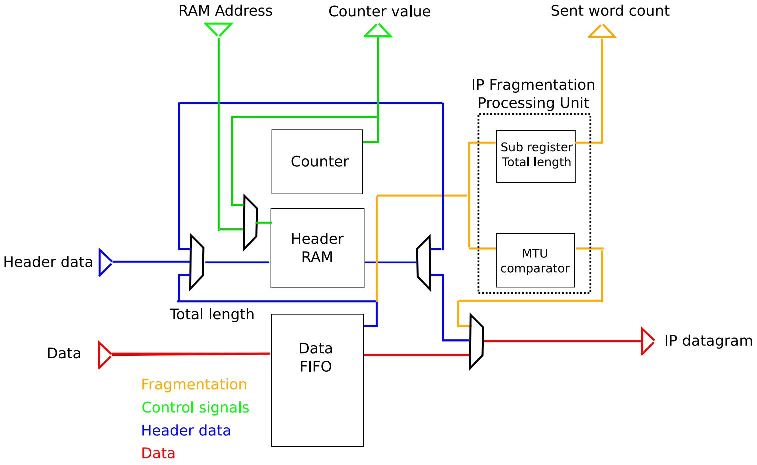

Figure 5: Processing Unit of the MEP core, with the fragmentation module.

B. *Storage access*

The FPGA injector device cannot store a large amount of physics data. In order to address our requirements, it has to read data from an external storage system. Mainly two options were studied:

- Access to a hard drive disk via the PCI interface.
- Access to a remote storage system via the protocol iSCSI [15].

The most scalable, interesting and challenging solution is the iSCSI implementation. Though it is currently provided by many industrial company for FPGA-based storage acceleration solutions, open source IP cores are not available yet. Its implementation calls for a quite a few time resources.

Here we would use it to access to a raw partition of our storage system, which would contain simulated physics events in a raw format. This partition would not be interpreted by a filesystem but would store directly the data.

C. *Trigger and Control System*

The hardware injector is triggered the same way as a normal Readout Board. It is receiving this trigger and all associated information via a TTC optic signal. This signal is encoded on a double channel, one is the proper trigger telling if the event is accepted. The other one is used to distribute information relative to the LHCb DAQ, as for example the destination HLT farm node, and information about the trigger. These information are required to write the IP and the MEP headers.

There are basically two ways to implement the reception of this signal. The first one is to interface directly a PIN diode. The other one is to use a TTCRx board [16].

This part is very important for the integration of the injection device in the control system, whereas it is not for preliminary tests. We can simulate the trigger information. This part relies on emulation, before to be implemented.

Nevertheless the selected solution is currently to interface the TTCRx board. It requires the design of a routing daughter board which would convert the TTCRx interface with the HSMC of the development board.

IV. Conclusion

This project is still very young. It is integrated into the upgrade of the LHCb detector, more particularly in the upgrade of the Online Data Acquisition system.

For the first few months of development, we focused on the implementation of the networking layer. So far we have the network architecture for data transmission in the MEP protocol. Though simulation is correct, it is required to carry out real performance tests in order to validate this design. The integration in the control system and the storage access layer implementation will follow shortly after.

Acknowledgments

This research project has been supported by a Marie Curie Initial Training Network Fellowship of the European Community's Seventh Framework Programme under contract number (PITN-GA-2008-211801-ACEOLE)

References

[1] The LHCb Collaboration, A Augusto Alves Jr *et al.*, The LHCb Detector at the LHC, JINST **3** S08005 (2008).

[2] J. Garnier *et al.*, High-Speed Data-Injection for Data-Flow Verification in LHCb, *16th IEEE Real Time* (2009).

[3] M. Cattaneo, LHCb Full Experiment System Test, *CHEP* (2009).

[4] LHCb HLT homepage, http://lhcb-trig.web.cern.ch/lhcb-trig/HLT.

[5] P. R. Barbosa-Marinho *et al.*, LHCb Technical Design Report, CERN/LHCC/2001-040 (2001).

[6] A. Bay *et al.*, The LHCb DAQ interface board TELL1, *Nucl. Instrum. and Methods* **A560** (2006) 494.

[7] Information Sciences Institute, University of Southern California, RFC791 - Internet Protocol.

[8] LHCb TFC homepage, http://cern.ch/lhcb-online/TFC.

[9] LHCb ECS homepage, http://cern.ch/lhcb-online/ecs.

[10] B. Jost, N. Neufeld, Raw-data Transport Format, EDMS 499933.

[11] Information Sciences Institute, University of Southern California, RFC793 - Transmission Control Protocol.

[12] Domenico Galli *et al.*, Performance of 10 Gigabit Ethernet Using Commodity Hardware, *16th IEEE Real Time* (2009).

[13] B.Taylor, Timing Distribution at the LHC, *8th Workshop on Electronics for LHC Experiments* (2002).

[14] J. Postel, RFC768 - User Datagram Protocol.

[15] J. Satran *et al.*, Internet Small Computer Systems Interface (iSCSI) (2004).

[16] J. Christiansen *et al.*, TTCrx Reference Manual (2004).