FPGA based data-flow injection module at 10 Gbit/s reading data from network exported storage and using standard protocols

This article has been downloaded from IOPscience. Please scroll down to see the full text article.

2011 JINST 6 C02003

(<http://iopscience.iop.org/1748-0221/6/02/C02003>)

View [the table of contents for this issue](#), or go to the [journal homepage](#) for more

Download details:

IP Address: 137.138.124.233

The article was downloaded on 27/07/2012 at 08:34

Please note that [terms and conditions apply](#).

RECEIVED: October 13, 2010

REVISED: January 10, 2011

ACCEPTED: January 19, 2011

PUBLISHED: February 7, 2011

TOPICAL WORKSHOP ON ELECTRONICS FOR PARTICLE PHYSICS 2010,
20–24 SEPTEMBER 2010,
AACHEN, GERMANY

FPGA based data-flow injection module at 10 Gbit/s reading data from network exported storage and using standard protocols

B. Lemouzy^{a,b}, J.-C. Garnier^{a,1} and N. Neufeld^a

^aCERN,
1211 Geneva 23, Switzerland

^bISIMA,
63173 Aubière, France

E-mail: benjamin.lemouzy@cern.ch

ABSTRACT: The goal of the LHCb readout upgrade is to accelerate the DAQ to 40 MHz. Such a DAQ system will certainly employ 10 Gigabit or similar technologies and might also need new networking protocols such as a customized, light-weight TCP or more specialized protocols. A test module is being implemented to be integrated in the existing LHCb infrastructure. It is a multiple 10-Gigabit traffic generator, driven by a Stratix IV FPGA, and flexible enough to generate LHCb's raw data packets. Traffic data are either internally generated or read from external storage via the network. We have implemented a light-weight industry standard protocol ATA over Ethernet (AoE) and we present an outlook of using a file-system on these network-exported disk-drivers.

KEYWORDS: Data processing methods; Simulation methods and programs; Computing (architecture, farms, GRID for recording, storage, archiving, and distribution of data); Data acquisition concepts

¹Corresponding author.

Contents

1	Introduction	1
2	Specifications	2
3	Implementation	3
4	Conclusion, status of the project and further developments	4

1 Introduction

LHCb [1] is one of the four large experiments at CERN which takes data from proton-proton collisions at the LHC. Data flows from hardware readout boards (TELL1 [2]) through a large Gigabit Ethernet network to an Event-Filter Farm (EFF) of 2000 nodes running the selection of the physics events [3]. The total event size, determined by the average number of electronics channels activated by a collision, is quite small: 60 kB according to what was observed during the first year. The data acquisition and data handling in LHCb are hence faced with a very high rate of rather small events. The current readout rate is 1 MHz and is planned to be upgraded to the collision rate of 40 MHz by 2016. This means that there would be no more rate reduction before the EFF. Because not all LHC beam-crossings contain particles, the event rate in the Event-Builder will be 30 MHz. Each event out of the 30 MHz will be about 100 KiB, taking into account the non-Gaussian distribution of fragment sizes from the readout-boards. An “event” is the aggregation of fragments. We expect around 1000 sources (corresponding to the readout-boards in the current system). An average fragment size would then be 100 bytes per source. We intend to use fragment coalescing to reduce the message rate using a protocol called Multi Event-fragment Protocol (MEP) [4]. Most of these figures are assumptions and may change until the year 2016. The upgraded data acquisition network will likely rely on 10 Gigabit Ethernet (10 GbE) or InfiniBand.

It is important to be able to test the data acquisition consisting of readout-network and event-filter-farm independent of the readout-boards. This decouples testing and commissioning from the readiness of custom-electronics boards and the availability of the detector hardware for central tests.

The current LHCb DAQ system has therefore been (also) commissioned using a software simulator [5, 6], based on a powerful PC server. Based on the experience with this important tool, we realize that this architecture will not easily scale with the upgrade of the overall DAQ system. We have therefore started to produce a completely new version of the dataflow simulator.

The project presented in this paper is motivated mainly by the following two requirements:

- We want to study implementations of different protocols over 10 Gigabit Ethernet.
- We want to test the high level trigger with a realistic (i.e. high) input rate, independent from the readout boards.

The idea is to provide a device which would be integrated into the system like a real readout board. It would behave like the readout boards, except that it would get simulated data from a storage system, for example the central LHCb Online Storage.

Such an improved simulator will also be very useful in the current LHCb system.

Section 2 presents the requirements and the resulting specifications of the project. Section 3 presents the main ideas and technologies employed in each part of the system. Section 4 discusses about the current limitations and further developments.

2 Specifications

The requirements of a dataflow simulator (often called "injector" in LHCb) are:

- To provide a data-flow identical to the normal data-flow coming from the detector and readout boards. This means that it has to send network packets as if they were coming from *all* the readout boards simultaneously. Since it has only a single link into the network it needs to dynamically change its IP address and other information. Data must be read from external storage (since in the case of an FPGA the internal storage cannot be assumed to be big enough to hold all relevant data. The simulated data-flow is usually represented by several files of ten million events.).
- The generated data-flow has to be complex enough in order to be used for trigger and Offline tests.
- It must be integrated into the DAQ as a readout board, in particular it needs to be driven by the central trigger distribution system of LHCb, the TFC system [7]) and to be triggered by it.
- To be integrated into the Experiment Control System (ECS [8]).

At the same time, it would be interesting to test protocols for the DAQ upgrade, like a simplified Transport Control Protocol (TCP) [9].

In order to inject at high data rate, a device with a 10 GbE interface is studied. Using a single 10 GbE interface would allow getting a 35 kHz rate according to the current design figures.

From a functional point of view the main task is to read simulated data and to format them in the DAQ networking protocol, according to the trigger information coming from the Readout Supervisor. It is required to be always synchronized with the Readout Supervisor and with peer injectors. This means that delays caused by reading simulated events, or accessing the network interface, have to be avoided.

We decided to implement the injector in hardware, based on an FPGA. This promises high performance for data processing and the possibility to drive 10 GbE interface at wire-speed.

A PCI development board, based on the Stratix IV GX FPGA, was chosen for a preliminary implementation [10]. It features two Small Form Factor Pluggable Transceiver (SFP+) connectors which allow us to implement 10 GbE interfaces. Two Gigabit Ethernet interfaces can be used for control and monitoring. A mezzanine connector permits to integrate the synchronisation with the Readout Supervisor via an extension board.

3 Implementation

Storage access. The injected data are read by the FPGA from storage, which is accessible via the network. Modern operating systems do not store files in raw form on the disk-blocks but use a *file system*. We need a file system which is understandable by the FPGA. Then we need to define the network protocol used to access the data.

Several file systems were studied. We can store data as raw data, i.e. no file system is used to format the partition. It is easy to read data because there is no fragmentation and no File Allocation table (FAT). On the other hand, this is not user friendly and it is very difficult to manage data in this format.

Another solution is to use an existing file system, like FAT32 [11]. The FAT has to be interpreted in order to read data, but it ensures good memory usage and users can easily manage files. Some performance issues can be perceived when the fragmentation gets too high. The FAT32 file system was chosen in the end, because it has the minimum set of features required to manage comfortably the data and at the same time matches our constraints and requirements. Moreover due to its simplicity the implementation is relatively straight forward.

Standard solutions for storage access are nowadays iSCSI (Internet Small Computer System Interface) [12] and Fibre Channel [13]. Fibre Channel was immediately discarded because of its cost, its complex implementation and its difficult integration in our system. iSCSI is based on TCP. The TCP state machine implementation on an FPGA is (still) a challenge. It brings a challenge in memory management, because of the TCP window: the TCP module needs to synchronize packet sequence numbers and to check the acknowledgement numbers to be sure that data are not lost or duplicated. A lot of memory can be spent to perform this, to keep the sent packets until they are all acknowledged, and like in the IP core, to manage the packet fragmentation. The statefulness of the protocol and the many features (the Linux implementation of TCP is several 10000 lines of C-code!) make iSCSI unsuitable for the scope of this project.

We decided to use a simpler protocol called ATA over Ethernet (AoE) [14]. As it relies only on Ethernet, it requires to be on the same network segment, but of course it misses the advantages of a routed protocol. In the LHCb Online environment these restrictions can be accepted. The injector needs to be connected to the storage system in the same subnet (also called virtual LAN). It is however possible to connect the injector to a commodity server via AoE.

Network implementation. In order to perform tests and to ensure that the injector is fully operational, several network layers have to be implemented, as shown in figure 1. For some layers, we decided to implement only the minimum required features. The basis of the network communication model, Ethernet, IP and ARP [15], are mandatory, for any of our application activity. Then other layers, more specific to this project, were added. The ATA over Ethernet layer is used to access the storage partition which stores data. MEP is used to encapsulate physics data and to send them through the network. The Internet Control Message Protocol (ICMP [16]) is used by the injector to respond to ping requests.

A round robin scheduler is used to choose which network module can use the Ethernet core. Each network module has the *EthernetOk* input, and the *EthernetRequest* and *EthernetFinish* outputs: when the module needs to send data through the Ethernet core, it sends a request. The sched-

Figure 1. Representation of network layers needed by the injector. Data that are coming from the ATA over Ethernet module are encapsulated in MEPs and sent through the network using IP and Ethernet layers.

Figure 2. Global view of injector

uler saves all requests and tests in circular order if a module needs the Ethernet core and, when the scheduler gives authorization to a module, it is waiting for the *EthernetFinish* signal to continue.

4 Conclusion, status of the project and further developments

All network layers are now implemented and the injector works as shown in figure 2. However, the injector still needs to be tested in the production environment to tune several parameters, like the buffering or priorities in the scheduler. A better implementation of the scheduler, managing priority queues, could be useful.

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] LHCb collaboration, A.A. Alves et al., *The LHCb Detector at the LHC*, 2008 *JINST* **3** S08005.
- [2] G. Haefeli et al., *The LHCb DAQ interface board TELL1*, *Nucl. Instrum. Meth. A* **560** (2006) 494.
- [3] LHCb HLT homepage, <http://lhcb-trig.web.cern.ch/lhcb-trig/HLT>.
- [4] B. Jost and N. Neufeld, *Raw-data Transport Format*, *EDMS* 499933.
- [5] M. Frank et al., *Online testbench for LHCb high level trigger validation*, *J. Phys. Conf. Ser.* **219** (2010) 022008.
- [6] O. Callot et al., *High-speed data-injection for data-flow verification at LHCb*, *IEEE Trans. Nucl. Sci.* **57** (2010) 497.
- [7] LHCb TFC homepage, <http://cern.ch/lhcb-online/TFC>.
- [8] LHCb ECS homepage, <http://cern.ch/lhcb-online/ecs>.
- [9] RFC: Transmission Control Protocol, <http://tools.ietf.org/html/rfc793>.
- [10] Hitechglobal development board, <http://www.hitechglobal.com/Boards/Stratix4GX.htm>.
- [11] Microsoft FAT32 File System Specification, <http://www.microsoft.com/whdc/system/platform/firmware/fatgen.mspx>.
- [12] RFC: Internet Small Computer Systems Interface, <http://tools.ietf.org/html/rfc3720>.
- [13] RFC: Fibre Channel (FC) Frame Encapsulation, <http://tools.ietf.org/html/rfc3643>.
- [14] ATA over Ethernet protocol specifications, <http://support.coraid.com/documents/AoEr11.pdf>.
- [15] RFC: Address Resolution Protocol, <http://tools.ietf.org/html/rfc826>.
- [16] RFC: Internet Control Message Protocol, <http://tools.ietf.org/html/rfc792>.