Implementation of a PC-based Level 0 Trigger Processor for the NA62 Experiment

M Pivanti[1], S F Schifano[2], P Dalpiaz[1], E Gamberini[1], A Gianoli[1], M Sozzi[3]

[1] Physics Dept and INFN, Ferrara University, V. Saragat 1, 44122 Ferrara, Italy
[2] Mathematics and Informatics Dept and INFN, Ferrara University, V. Saragat 1, 44122 Ferrara, Italy
[3] Physics Dept and INFN, Pisa University, Largo Pontecorvo 3, 56127 Pisa, Italy

E-mail: gianoli@fe.infn.it

Abstract.
Lowest level (sometimes called Level 0, L0) triggers are fundamental components in high energy physics experiments, and yet they are quite often custom-made. Even when using FPGAs to achieve better flexibility in modifying and maintaining, small changes require hardware re-configuration and changes to the algorithm logic could be constrained by the hardware. For these reasons we are developing for the NA62 experiment at CERN a L0-trigger based on the use of a PC and commodity FPGA development board.

1. Introduction

The performance of "level 0" (L0) triggers is crucial to reduce and appropriately select the large amount of data produced by detectors in high energy physics experiments. This selection must be accomplished as fast as possible since data staging is a critical issue and the memory for data staging is a critical resource. For example in the NA62 experiment at CERN the event rate is estimated at about 10 MHz and the L0-trigger should reduce it by a factor of 10 within a time budget limit of 1 ms. So far, the most common approach to the development of a L0 trigger system was based on custom hardware processors, so event filtering is performed by algorithms implemented in hardware. More recently, the implementation of custom processors has been based on FPGA devices, whose hardware functionalities can be configured using specific programming languages. The use of FPGAs offers greater flexibility in maintaining, modifying, improving filter algorithms, however even small changes require a hardware re-configuration of the systems, and changes to the algorithm logic can be limited by hardware constraints that have not been foreseen at development time. So, even if this approach guarantees fast processing, strong limitations still remain in the available flexibility when changing filtering algorithms on the fly or testing more filtering conditions at the same time, as required during the data-taking phase of the experiment.

In this paper we present an innovative approach in the implementation of a L0-trigger system based on the use of a commodity PC, describing the architecture we are developing for the NA62 experiment at CERN. Data streams coming from the detectors are collected by an FPGA installed on a PCI-Express board plugged on a commodity PC. The FPGA receives data from detectors via Gigabyte Ethernet (GbE) data links, and stores them into the main memory of the PC. The PC then performs the filter algorithms on data available on its own memory, and writes back results to the FPGA for routing to the appropriate destination.

2. NA62 requirements and DAQ architecture

The goal of the NA62 experiment at the CERN SPS[1] is to measure the branching ratio of the ultra-rare decay of the charged kaon $K^+ \to \pi^+\nu\overline{\nu}$. This decay, together with its companion $K^0_L \to \pi^0\nu\overline{\nu}$, can be predicted to unusually high accuracy in the Standard Model, making them powerful probes for new physics beyond the Standard Model. The tiny expected branching ratios, in the 10^{-10} range, are the price to pay for their extraordinary sensitivity and pose great challenges to their experimental determination.

We refer the reader to [2] for a detailed description of the experiment and we concentrate here on the DAQ requirements. Since the experiment is a rare-decay one, the trigger and data acquisition systems need to be high performance ones.

NA62 is a fixed target experiment. Compared to collider ones our requirements for radiation hard electronics are less stringent. The space available is also less of a problem, giving us more options on where to put the electronics. Finally looking at the beam structure, our beam is not bunched in time but follows a low duty-cycle spill structure (typically 9 s of beam every 40 s). The rate of events in the decay region is totally dominated by background and according to simulations the rate on the main detectors is of the order of 10MHz. Since we do not need a fine time synchronization for the events, we can use any frequency above this for the free running master clock. We chose the same frequency as LHC experiments, i.e. 40,07897 MHz [4].

Software triggering would present great advantages in terms of flexibility, but a completely triggerless approach (full continuous readout) was deemed not a realistic option because of the tens of thousands channels involved, the aforementioned expected rates and an expected event size of the order of 30 kB. A scheme with a single hardware trigger level (L0) was thus adopted, while following second and third stages (L1 and L2) will be completely software based exploiting PC-farms with large input bandwitdh. This leads us to a unified Trigger and Data AcQusition (TDAQ) system [2], which assembles trigger information from readout-ready digitized data in a simple and cost-effective way. For most of the detectors involved, the building block is the TEL62 Trigger and Data Acquisition board[3].

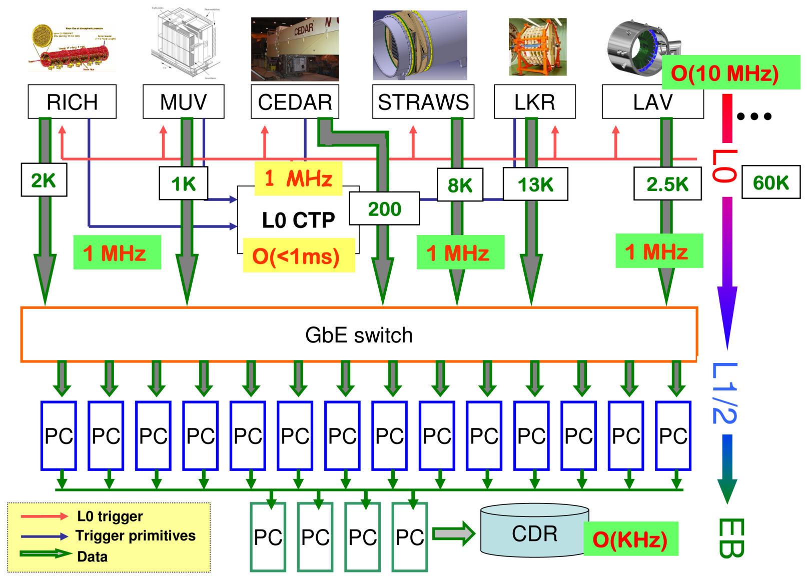

Figure 1. Schematic drawing of the NA62 trigger and DAQ system. All detectors' connections are gigabit ethernet links.

The requirement for the L0 are to reduce the total trigger rate to $\sim$ 1 MHz with a maximum latency of 1 ms using information coming from the charged hodoscope (CHOD), Cherenkov (RICH), Photon Veto (LAV), Electro-magnetic calorimeter (LKr) and Muon Veto (MUV) detectors. The trigger primitives from each sub-detector involved in the trigger decision are generated in the TEL62 boards and sent out asynchronously via multiple standard gigabit ethernet connections with accompanying high-resolution (few ns) time-stamp data. The default (primary trigger) algorithm does not require to reconstruct any track o cherenkov ring, but has to re-align in time the sub-detector primitives. Data from all sub-detectors will be stored in front-end buffers (such as those present on the TEL62) during L0 evaluation. In case of a positive L0 trigger decision, the L0 will be sent to the TEL62 boards by the Trigger and Timing Control (TTC) system[4] after a fixed latency. The L0 latency is constant and the sub-detectors will send their data to a dedicated PC farm for L1 and L2 selection.

3. L0 Trigger Processor

Normally a critical component like the L0 Trigger Processor (L0TP) is a full custom device. As such it requires a certain amount of specific knowledges both from the developers for its implementation and from end-users to maintain its application logic. The collaboration decided to investigate the possibility of relying only on commodity components. We are interested not only in reducing costs and development time, but also in simplifying future upgrades during the experiment lifecycle. We are especially interested in the capability to modify/refine trigger evaluation "on the fly", without hardware intervention. Our goal is to allow "not-hardware-aware" users to take care of primitives analysis using a high-level language.

To achieve this result we investigated the feasibility to develop the L0TP using a common PC running the Linux operating system. Due to the need to interface the PC with 6 detectors via Gbit Ethernet (GbE) links, managing with the most possible efficiency the primitives receiving and decoding, we decided to use a commercial FPGA development board board that implements 8 Gbit Ethernet interfaces and implements a PCI Express link to deliver data directly to the main-memory of the PC.

The PC used hosts an Intel core-i7 3930K CPU and 16GB of RAM DDR3@1333, while for the FPGA part we used a Terasic DE4 developement kit that features an Altera Stratix-IV GX230 FPGA, 4 integrated GbE links, with the possibility to add 4 more GbE links using two expansion board, and one 8x PCI Express Gen. 2 link to interface with the PC. The operating system we use for the PC is Scientific Linux 6.4 for which we developed a kernel module (Drv) for low-level access to the device (development board), a lightweight library (Lib) built on top of Drv to export device-features to users, and on top of Lib we developed an user application (App) to test all the functionalities of the device.

4. Implementation

In order to understand if a PC-based L0TP could fit the timing constraints, the first step is to evaluate if the Round-Trip Time (RTT) in the path FPGA to CPU and vice-versa could be kept significantly lower than the given time-budget of 1 ms in order to i) receive primitives, ii) evaluate trigger conditions, iii) send back a trigger.

To evaluate the above mentioned RTT in a reliable manner, we developed a firmware/software system that generates fake primitives inside the FPGA and send them to the main memory of the PC. The CPU analyzes the primitives and send-back to the FPGA the timestamp of each primitive, that are then compared with the current timestamp inside the FPGA, the difference is the RTT. Figure 2 shows the setup.

Going deep into the design, the firmware implements two generators: one for timestamps needed to calculate the RTT (TsGen); the other for data needed to evaluate the trigger condition (PrGen). The TsGen is simply a counter that increments at each FPGA clock cycle (4 ns), while

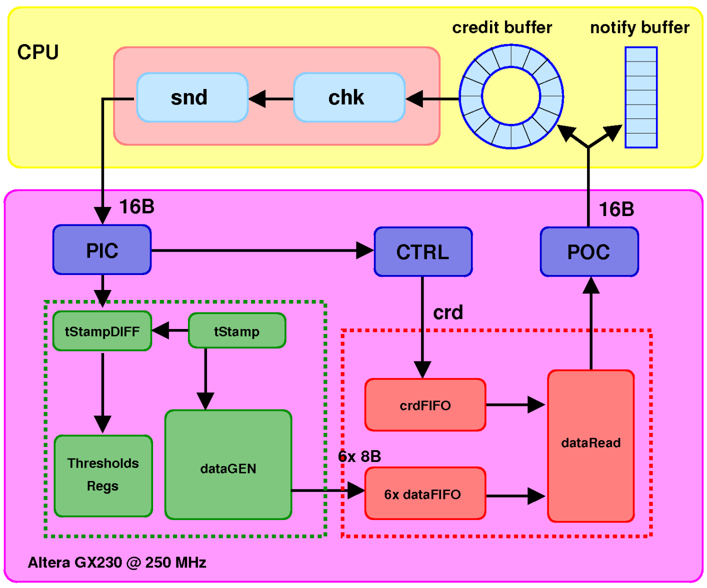

Figure 2. Schematic diagram of the firmware/software: the FPGA firmware is made of a "generator" module (containing timestamp and primitive generators) and a PCIe interface module; the device driver on the PC prepares and manages the memory to be used by the FPGA.

the PrGen generates 6 streams of primitives (simulating 6 sub-detectors participating to the L0 trigger) that are queued into separated FIFOs, waiting to be sent.

The so-called Primitives-Consumer (PrCon), must interact with the PCI Express Interface (PCIe) to deliver primitives to main PC memory, requiring the physical address of the buffers allocated for the purpose. These addresses are provided by the PC and stored into the Credit-FIFO (CrdFifo). One credit (Crd) contains i) the physical address at where the buffer starts in main-memory ii) the buffer size iii) and the Notify-index.

As primitives begin to be generated, PrCon extracts one credit and uses its informations to produce packets to be sent over the PCIe, composed by 6x8 Bytes primitives, 1x8 Bytes timestamp, 1x8 Bytes additional informations.

The size of 64Bytes for PCIe packets is equal to the "cache line size" of the CPU, allowing the primitives-transactions-management to be the most efficient as possible.

The PCIe module on this design is an Altera Intellectual Property that exports a simplified streaming interface to the users, called Avalon-Streaming[5].

The buffers in main-memory are allocated in kernel-space by Drv, granting "continuos areas" in physical memory made directly accessible to user-applications via "mmap()" system-call. In this way the application can have a direct access to them, and avoid the overhead given by kernel-calls each time the buffers must be accessed.

Once the primitives are stored into main-memory, the test-application accesses them to evaluate a trigger condition and send-back to the FPGA the timestamp included into the packet. The FPGA compares such timestamp with the actual one to compute the RTT. The RTT value is then compared to a set of thresholds, and the corresponding counter is increased by one: at the end of a simulation run the counters provide us the RTT distribution.

5. Results

To measure the RTT we prepared a simplified set of trigger primitives and selection algorithm. The simplification lies in the trigger primitives being "time aligned" and the selection algorithm checking just one trigger condition, skipping the event as soon as a negative condition is found (e.g. there is a signal in one sub-detector where the absence of any signal is required). Figure 3 shows the distribution of RTT values for a single 120 s long run. This measurement was repeated several times to check the stability of the average value and the result is shown in figure 4.

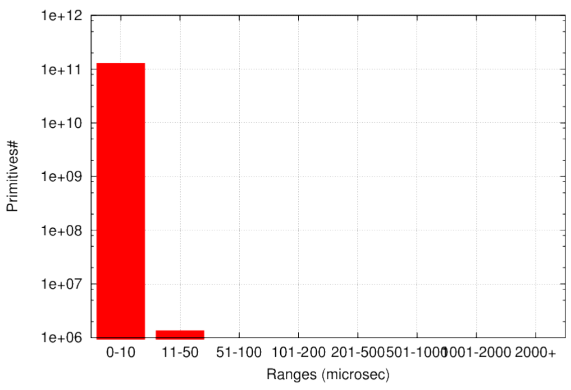

Figure 3. Distribution of RTT values for a single run, one trigger condition, primitives time aligned

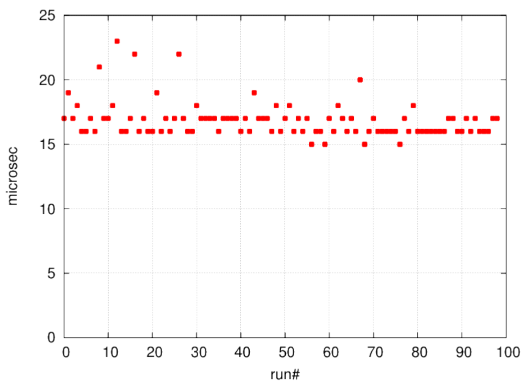

Figure 4. Distribution of average RTT values for a series of runs, one trigger condition, primitives time aligned

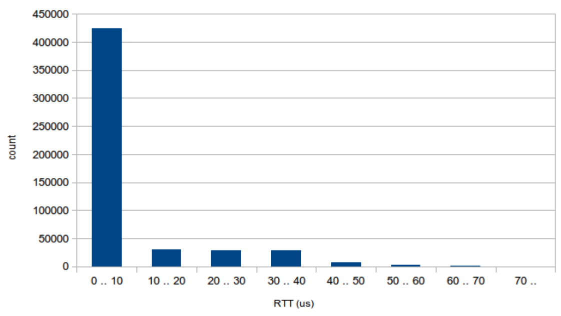

Figure 5. Single run RTT distribution using polling synchronization, eight trigger conditions, primitives not time aligned

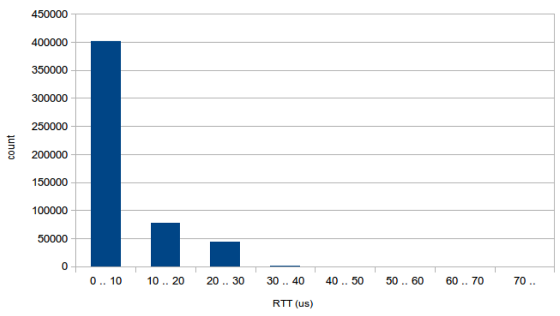

Figure 6. Single run RTT distribution using credit-notify synchronization, eight trigger conditions, primitives not time aligned

A more detailed test has been made, using a realistic set of trigger primitive (equivalent to about 50 ms of data) and a more sophisticated selection algorithm. This time the primitives are not "time aligned", meaning that the selection algorithm has to align them. Moreover the algorithm was modified to look for more than one trigger. As a further matter, we also investigated if the credit-notify method to synchronize FPGA and CPU is indeed efficient, replacing it with a polling method. The results are shown in figures 5 and 6. Even if the polling method has a higher first bin, it presents a longer tail with RTT of the order of 70 μs for the worst case, compared to the credit-notify method which has a worst case RTT of the order of 40 μs.

6. Conclusions

L0 trigger systems are normally custom-made hardware systems. We investigated the possibility to implement the L0 Trigger Processor for the NA62 experiment at CERN based on commodity, off the shelf hardware and proved the feasibility of the project. The design and the implementation of it was briefly described, together with the results of the tests on the system latency.

Acknowledgments
This work has been done in the framework of the COKA project of INFN.

References
[1] The NA62 Collaboration *Proposal to measure the Rare Decay $K^+ \to \pi^+ \nu\bar{\nu}$ at the CERN SPS* CERN SPSC-2005-013
[2] The NA62 Collaboration 2010 *NA62 Technical Design Document* (Geneva: Cern)
[3] Angelucci B, Pedreschi E, Sozzi M, Spinella F 2012 *JINST* **7** C02046
[4] Taylor B G (CERN RD12 collaboration) *Timing distribution at the LHC*, in: Eighth Workshop on Electronics fo LHC Experiments, Colmar, 9-13 September 2002.
[5] *Avalon Interface Specifications* Altera MNL-AVABUSREF-2.1, May 2013