# Demonstration of track reconstruction with FPGAs on live data at LHCb

*Federico* Lazzari[1,2,*], *Wander* Baldini[3,4,], *Giovanni* Bassi[5,2,], *Andrea* Contu[6,], *Riccardo* Fantechi[2,], *Sofia* Kotriakhova[3,4,], *Michael Joseph* Morello[5,], *Fabio* Novissimo[1,2,], *Giovanni* Punzi[1,2,], and *Giulia* Tuci[7,]

[1]Università di Pisa, Pisa, Italy
[2]INFN Sezione di Pisa, Pisa, Italy
[3]Università degli Studi di Ferrara, Ferrara, Italy
[4]INFN Sezione di Ferrara, Ferrara, Italy
[5]Scuola Normale Superiore, Pisa, Italy
[6]INFN Sezione di Cagliari, Cagliari, Italy
[7]Ruprecht-Karls-Universität Heidelberg, Heidelberg, Germany. Formerly with University of Chinese Academy of Sciences, Beijing, P.R.China

**Abstract.** The LHCb experiment is currently taking data with a completely renewed DAQ system, capable for the first time of performing a full real-time reconstruction of all collision events occurring at LHC point 8. The Collaboration is now pursuing a further upgrade ("LHCb Upgrade-II"), to enable the experiment to retain the same capability at luminosities an order of magnitude larger than the maximum planned for the current Run3. To this purpose, a vigorous R&D program is ongoing to boost the real-time processing capability of LHCb, needed to cope both with the luminosity increase and the adoption of correspondingly more granular and complex detectors. New heterogeneous computing solutions are being explored, with the aim of moving reconstruction and data reduction to the earliest possible stages of processing. In this talk, we describe the results obtained from a realistic demonstrator for a high-throughput reconstruction of tracking detectors, operating parasitically on real LHCb data from Run3 in a purposely-built testbed facility. This demonstrator is based on a extremely parallel, "Artificial Retina" architecture, implemented in commercial, PCIe-hosted FPGA cards interconnected by fast optical links, and encompasses a sizeable fraction of the LHCb VELO pixel detector. The implications of the results in view of potential applications in HEP are discussed.

## 1 Introduction

With the slowing down of Moore's law, HEP experiments are looking at heterogeneous computing solutions as a way to manage ever-increasing data flows and complexity. LHCb is at the forefront of these developments due to its specific physics needs, calling for the full software reconstruction of events in real-time at the LHC average rate of 30 MHz. LHCb has adopted a GPU-based solution for the first stage of the High Level Trigger (HLT1) for Run3 [1]. Additional computing enhancement are being sought for Upgrade-II, where the

[*]e-mail: federico.lazzari@cern.ch

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

luminosity will be increased by a further factor of 5 to 10 with respect to Run 3. To this purpose, a coprocessor testbed has been established, to allow parasitical testing of new processing solutions in realistic DAQ conditions during the Run 3.

One such solution under development is a highly-parallelized custom tracking processor based on the "Artificial Retina" architecture. The "Artificial Retina" architecture [2] takes advantage of FPGA parallel computational capabilities, by distributing the processing of each event over an array of FPGA cards, interconnected by a high-bandwidth (~15 Tb/s) optical network. This is expected to allow operation in real-time at the full LHC collision rate, with no need for time-multiplexing or extra buffering. Several lab prototypes have already been built and tested during the past years of R&D, allowing to separately test the various functionalities of the system on a reduced scale [3–5]. In this work, we describe the results obtained with a more complete and realistic demonstrator of the technology, covering a significant chunk of an existing LHCb detector with all the functionality expected from a final device.

## 2 The "Artificial Retina"

The "retina architecture" is an arrangement of parallel computing units (cells) fed by a custom distribution network, programmed to perform a computation resembling the "Hough transform" [6], a mathematical approach to finding lines in image processing tasks. A detailed description of the principles of this architecture and of the cell implementation can be found in [3, 7, 8]. We summarise here the most relevant points.

As a preliminary step, the track space to be covered is discretised and represented as a matrix of cells (Fig. 1 (left)). The center of each cell defines the parameters of a reference track. The reference track intersects the detector's layers in specific spatial points called receptors. Each cell is mapped to a custom computational engine, that is implemented in a modest amount of logic circuitry inside the FPGA, physically independent from all others. The task of the engine is to receive hits in input and check whether they are compatible with the assigned reference track.

To this purpose, each computational engine assigns a weight to every received hit, that is determined by its distance from the receptor on that layer. The weight can in principle take an arbitrary form, but a Gaussian function works well [9]. The weight function is conveniently truncated to zero at some distance from the receptor ("search distance"). The weights of all received hits are accumulated into a single value, whose final value ad the end of the event represents the 'excitation level' of the corresponding cell. A high value indicates good compatibility of the set of received hits to the reference track (Fig. 1 (centre)).

After all cells have received all relevant hits from the event at hand, the next step of cluster finding begins. This is the determination of local maxima of the excitation function within the matrix of all cells. Local maxima over a threshold value are taken as track candidates, with the position of their centroid within the matrix taken as estimate of the candidate track's parameters (Fig. 1 (right)).

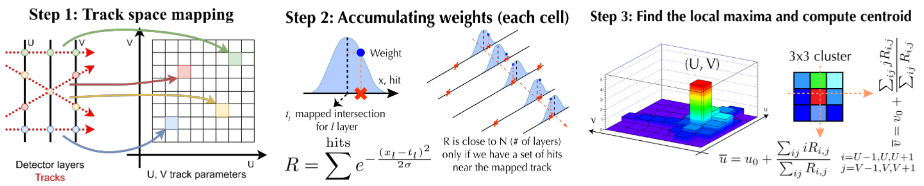
Step 1: Track space mapping
U
V
V
U
Detector layers
Tracks
U, V track parameters
Step 2: Accumulating weights (each cell)
Weight
x, hit
$t_l$ mapped intersection for $l$ layer
$R = \sum^{\text{hits}} e^{-\frac{(x_l - t_l)^2}{2\sigma}}$
R is close to N (# of layers) only if we have a set of hits near the mapped track
Step 3: Find the local maxima and compute centroid
(U, V)
3x3 cluster
$\overline{u} = u_0 + \frac{\sum_{ij} iR_{i,j}}{\sum_{ij} R_{i,j}}$
$\overline{v} = v_0 + \frac{\sum_{ij} jR_{i,j}}{\sum_{ij} R_{i,j}}$
i=U−1,U,U+1
j=V−1,V,V+1

**Figure 1.** Track reconstruction steps with the "Artificial Retina" architecture.

The steps outlined above are executed in a pipeline, so that all the different parts of the device are always active. This ensures optimal exploitation of the hardware for maximum throughput. Each step is executed using the output of the previous one as input, but the two can be performed at the same time on different parts of the data, possibly on different events. The same paradigm is also adopted at a more detailed level, within the steps previously described, according to a full dataflow protocol.

A special word, called EndEvent (EE) is used on each data channel to separate the data belonging to different events; this allows each part of the system to distinguish them without the need for a global event synchronization. Every processing block works in parallel with all the others at the same horizontal level, thus ensuring a very high degree of parallelization of the processing. The overall flow of data is regulated by a back-pressure mechanism to avoid data-loss when the processing time of a step is not deterministic. When an entity is not ready to receive new data, it raises a 'hold' signal, pausing the output of the previous entity [8].

## 2.1 Distribution network

The distribution network is the element of the "Artificial Retina" architecture charged with delivering the hits coming from different readout lines of the detector, to the relevant cells within the parameter space grid. It can be said to effectively operate a multidimensional change of coordinates; however, the corresponding function is a multi-valued one, as a single hit can conceivably be part of tracks having different parameters. This means that the *retina* network needs to create and deliver multiple copies of the same hit to different cells within the grid. This sets it apart from most other networks in use.

We implemented the Distribution Network as a modular design, with the dispatcher as the basic block. The dispatcher has two inputs and two outputs, being able to send any input to any output (possibly both) according to the pre-computed routing scheme. The basic components of the dispatcher are the splitter and the merger [8]. We call an entity composed of several dispatchers a switch. Combining a sufficient number of dispatchers, arranged in a number of layers, it is possible to build a distribution network with any desired number of inputs and outputs (Fig. 2). The network works as a pipeline, with each horizontal layer being a pipeline stage. This makes the throughput independent of the number of layers, while the latency is proportional to their number.

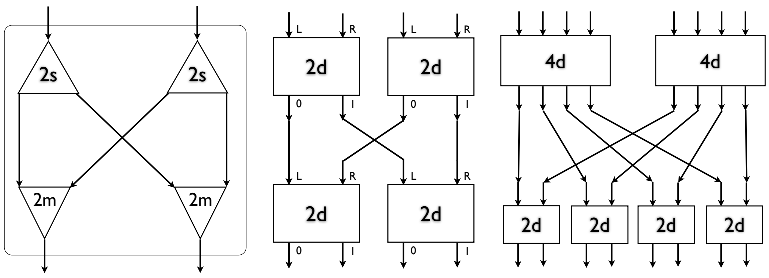

**Figure 2.** Interconnection between splitter (2s) and merger (2m) to build a dispatcher (left), between dispatchers (2d) to build a switch with 4 input/output (centre), and between 2 sub-switches with 4 inputs/outputs (4d) and 4 dispatchers to build a switch with 8 inputs/outputs (right).

The complete network is too wide to fit in a single electronic card, and it must be segmented vertically and distributed over multiple boards. Given the presence of many lateral connections, we collapsed all connections crossing segment boundaries into a single layer, separating two sections ("Pre-Switch" and "Post-Switch") neither of which contain any cross-segment connections, and as such can be implemented as arrays of separate entities [5]. The layer of lateral connections is implemented as a bundle of fast optical links.

This arrangement accommodates all the entities as separate blocks within a single board. The whole system is then formed by an array of boards of identical structure (although the content of internal storage elements are different for each board). Data comes in and out of each board through the PCIe interface with the host server. The optical connections are carried by a bunch of optical cables, ensuring point-to-point connections between boards. As such, it is implemented through an optical patch panel (Fig. 3), where multi-fiber cables are routed and internally spliced to reach all needed destinations, according to a reconfigurable topology.

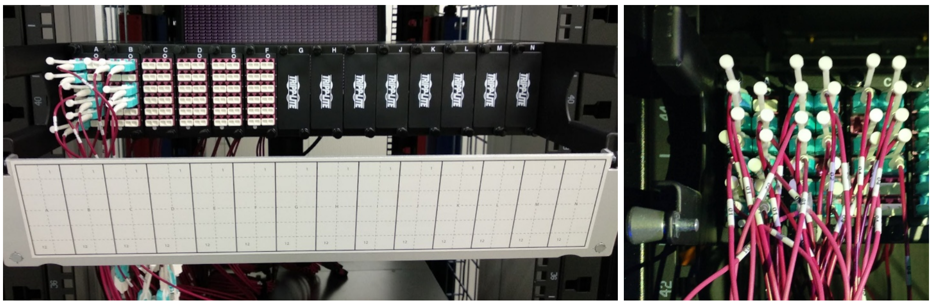

**Figure 3.** Optical patch panel (left). Cabling for a 8-nodes full-mesh network (right).

## 3 Demonstrator

The demonstrator was built with the complete functionality expected from a future device participating to LHCb data taking, including the capability to operate in realistic DAQ conditions. As a target detector for the present study, we chose the VELO pixel detector. The VELO detects charged particles in the region closest to the interaction point, aiming at reconstructing primary and secondary vertexes with a spatial resolution smaller than typical decay lengths of $b$- and $c$-hadrons in LHCb ($c\tau \sim 0.01 - 1$ cm), in order to discriminate between them. It consists of 52 modules positioned along the beam axis (26 per side), 38 of which are located downstream of the nominal interaction point. Each module is read-out by a DAQ Board.

The choice of the VELO for our demonstrator was motivated by several reasons: it is a complex, high resolution detector that is crucial to the experiment, and comes first in the reconstruction sequence; detailed simulations of retina reconstruction of the VELO were already available at the time of starting the demonstrator project [9]; its data are readout over a comparatively smaller number of lines in comparison to others LHCb's detectors, this allows us to build a demonstrator covering a meaningful portion of the target detector with an affordable quantity of hardware.

Our demonstrator system includes 16 downstream modules of the right-hand side of the VELO, corresponding to about a quadrant of VELO tracking space, using just 8 PCIe-interfaced FPGA boards. The whole system fits within a single server, and was installed and tested within the LHCb Coprocessor TestBed facility (Fig. 4). We used commercially available boards, each carrying one Stratix 10 FPGA device (1SG280HN2), produced by Bittware/Molex and commercialised under the name 520N [10]. Each board carries a PCIe

Gen3 x16 connector and 16 XCVRs, with a maximum bandwidth of 26 Gbps each. The chosen communication protocol is the Intel SuperLite II V4. This protocol is compatible with the back-pressure mechanism ant its source code is openly available [11]. The server make and model are the same as the ones used for LHCb's Event Builder, to ensure full compatibility with the LHCb Online environment.

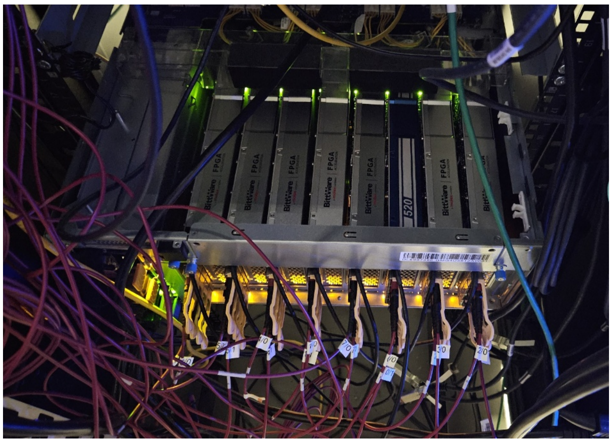

**Figure 4.** Photo (from above) of the retina demonstrator inside a single testbed server.

### 3.1 Tests with simulated LHCb data

Figure 5 shows the organisation of the Demonstrator firmware. Each FPGA stores hits from two VELO modules in an internal RAM. Hits are read in-loop to provide a continous flux of data. Hits are then distributed to the boards trough the optical network. Our network topology is a full-mesh, embodying one of four sections of a wider 'dragonfly' network required by an hypothetical complete system that could reconstruct the whole forward region of the VELO [5]. Our custom switch (see Sec. 2.1) routes the hits to the appropriate engines for track reconstruction. Finally, the output is sent back to the host server via PCIe. In these tests, we prescaled the output to make it possible to test its exact adherence to the results expected from our C++ bitwise simulation, as the unprescaled output rate of the device is way too high for the capabilities of our testing hardware.

We made the demonstrator to constantly check the consistency of the EE word and the alignment of data that flow inside the design. Thus the Demonstrator is able to detect EE corruption and mixing of hits from different events. The host then compares bit-by-bit the reconstructed tracks to the tracks generated by the C++ simulation of the Demonstrator. It produces an error in case of the occurrence of any missing or extra tracks, or of any tracks differing from the output of the software emulation.

Overall the firmware of each FPGA uses 285 kALMs: 31% of the available programmable logic, and 3265 memory blocks (28%). Of these used resources, most are required by the cells matrix: ~71% of logic and ~61% of memory.

The demonstrator was input bunches of events generated by the official LHCb simulation at Run3 conditions, center of mass energy $\sqrt{s} = 14$ TeV and instantaneous luminosity $\mathcal{L} = 2 \times 10^{33}$ cm$^{-2}$ s$^{-1}$). The system was capable of processing those events a rate of 16.2 MHz, in runs lasting uninterruptedly for 10 days without errors, before stopping on detection of

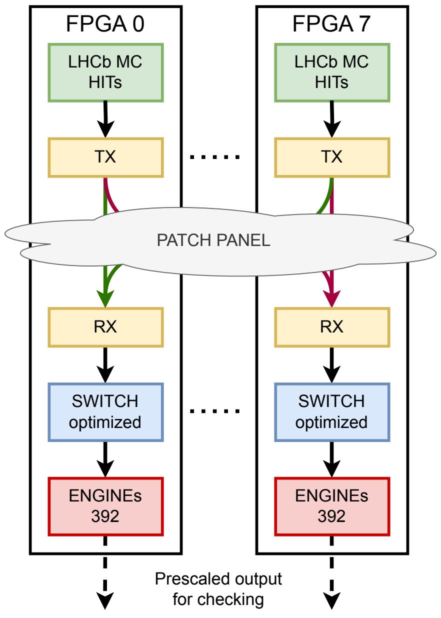

**Figure 5.** Structure of the Demonstrator's firmware.

a single EE corruption event occurred in the communication between the boards. Figure 6 shows an example display of a single event, showing generated and reconstructed tracks.

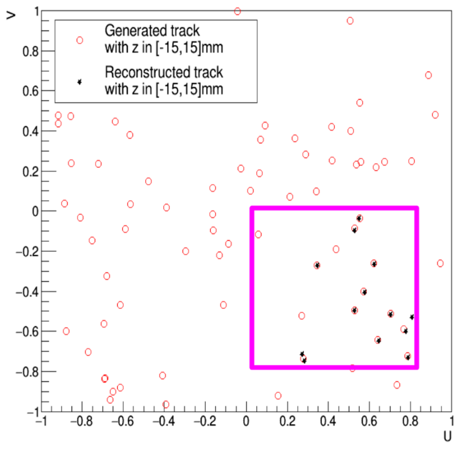

**Figure 6.** Generated and reconstructed tracks for a single event. The box highlights the covered track parameters space.

This type of test has been repeated several times, and confirmed the conclusion that the system can run reliably without errors for several days without interruption. In specific tests we quoted an upper-limit to the Bit Error Ratio (BER) of $4.2 \cdot 10^{-17}$ ($CL = 95\%$), which is a sufficient level of performance for physics data taking. In addition, tests performed at higher ambient temperature and/or slower airflow showed a clear relationship between high temperature and error rate. This has been tracked down specifically to over-heating of serializers/transceivers. This is directly connected to the current lack of a cooling system in the testbed location, and will not be an issue for the production system, that is planned to operate in the temperature-controlled environment of the LHCb Data Center.

### 3.2 Running the retina demonstrator on live LHCb data

While the previous tests aimed to test the ability of our system to operate at full speed reliably and repeatedly on a limited-size sample of events, we also wanted to test its behavior when input with ever-changing real-collision data, even if the rate of such events available at the testbed is currently limited.

In order to do so, a dedicated data-chain has been developed (Fig.7): RawEvents arriving from the LHCb's monitoring farm at the rate of 1 kHz are buffered and stored on disk in files of approximately 2 GB each (corresponding to ~2 minutes of data taking). VELO hits are extracted from the file. These hits are clusters of active pixel computed inside the VELO readout boards using an algorithm derived from the "Artificial Retina" [12].

As in previous tests, each FPGA handles data from two VELO modules; however, hits are loaded directly into the internal input FIFOs rather than in the FPGA RAM, using the stock PCIe driver provided by the board manufacturer. This makes the demonstrator process the real data exactly as it did with the simulated data, keeping monitoring possible EE corruption and mixing of events. The final output FIFOs of the demonstrator are then read and permanently stored onto local disk for later analysis. During the process, the Run Number and the original name of the incoming file are persisted, to allow, in future studies to be performed offline, a direct comparison between the "Artificial Retina" reconstruction of real collision data and the standard LHCb processing. Our monitoring system allows also periodic injection of simulated events of known characteristics to check reliability of operation.

The system is currently in data-taking mode, running smoothly without errors, and planned to continue through the end of the 2023 LHCb run.

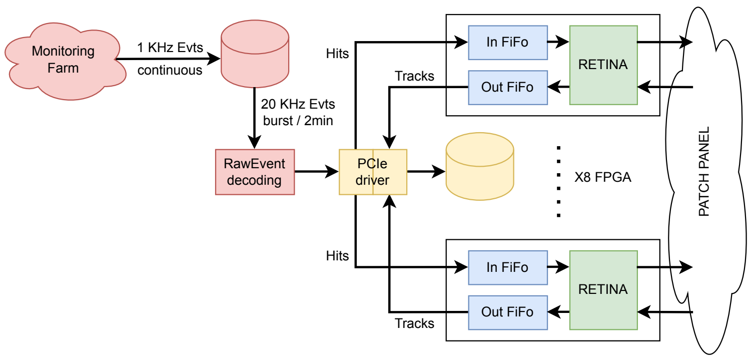

**Figure 7.** Dataflow from the monitoring farm to the demonstrator in the live data test.

## 4 Conclusion

We designed a FPGA-based track-finder demonstrator for the LHCb VELO detector, implementing and testing in a single system all functionalities of the "Retina" architecture. When processing simulated data at Run3 conditions ($\mathcal{L} = 2 \times 10^{33}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$), the demonstrator achieved an event throughput of 16.2 MHz with good long-term reliability. This is an unprecedented throughput, and it falls short of the LHC average beam crossing rate of 30 MHz by just a factor of 1.85x. Indeed, the 30 MHz throughput figure is expected to be within reach of the current hardware with some optimization of firmware parameters, like clock frequency, event-header overhead, hit duplication in the switch and in the engine distribution, and the final output logic chain. Each of those factors has been studied, and the combined effect of their optimisation estimated to be in excess of a factor of 2.0x.

After verifying its correct operation, the demonstrator was connected to the LHCb's monitoring farm through the testbed facility infrastructure, and is currently processing collisions happening live at LHCb during physics data taking, without interfering with the regular DAQ. A detailed evaluation of the real data running experience will make the topic of a future work.

The results obtained in the current work strongly support the conclusion that the technology is viable for real-world applications in future HEP experiments. A specific proposal is currently being considered by the LHCb collaboration, for a first implementation in the upcoming Run 4 of the LHC, targeting the reconstruction of tracks originating outside of the VELO acceptance. This type of reconstruction has a intrinsically high computational cost, due to the large combinatorics involved, and has the potential of improving acceptance and trigger efficiency for a broad category of physics process going under the general definition of "Long-lived particles decays" (LLPs). This includes common particles such as $K^0_S$ and $\Lambda$, as well as many hypothetical BSM states ("exotics"). The potentials and the technical hurdles associated with this trigger enhancements have been analysed in a few past reports [13–15].

We gratefully acknowledge the R&D funding received from INFN, without with this project would not have been possible.

## References

[1] LHCb Collaboration, *LHCb Upgrade GPU High Level Trigger Technical Design Report* (CERN, Geneva, 2020)
[2] L. Ristori, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **453, 1**, 425-429 (2000)
[3] R. Cenci et al., Proceeding of Science **TWEPP-17**, 136 (2018)
[4] F. Lazzari et al., Zenodo **CTD2020**, https://doi.org/10.5281/zenodo.4034418 (2020)
[5] F. Lazzari et al., Journal of Instrumentation **17, 4**, C04011 (2022)
[6] P. Hough, Proc. Int. Conf. High Energy Accelerators and Instrumentation **C590914**, (1959)
[7] G. Punzi et al., Proceeding of Science **Vertex2019**, 047 (2020)
[8] F. Lazzari, *Improving charm CPV measurements with real-time data reconstruction* (Ph.D. thesis presented at Università degli Studi di Siena, Siena, 2022) (https://cds.cern.ch/record/2813167)
[9] G. Tuci and G. Punzi, EPJ Web Conf. **245**, 10001 (2020)
[10] https://web.archive.org/web/20230603104042/https://www.bittware.com/products/520n/
[11] https://community.intel.com/t5/FPGA-Wiki/High-Speed-Transceiver-Demo-Designs-Intel-Stratix-10-GX-Series/ta-p/735749
[12] G. Bassi et al., IEEE Trans. Nucl. Sci. **70, 6**, 1189-1201 (2023)
[13] M. J. Morello et al., J. Phys. Conf. Ser. **1525, 1**, 012101 (2020)
[14] L. Pica, *Selecting LLP in the first level trigger at LHCb* (CERN, Geneva, 2022)
[15] L. Pica, *Selecting long-lived particles in the first trigger level at the LHC* (CERN, Corfu, 2023)