PROCEEDINGS OF SCIENCE

Real-time reconstruction of pixel vertex detectors with FPGAs

G. Punzi[*cf], **W. Baldini**[b], **G. Bassi**[ce], **A. Contu**[a], **M. Dorigo**[d], **R. Fantechi**[c], **F. Lazzari**[cg], **M.J. Morello**[ce], **S. Stracka**[c], **G. Tuci**[cf], **G. Vitali**[ce†]

[a]*INFN, Sez. di Cagliari,* [b]*INFN, Sez. di Ferrara,* [c]*INFN, Sez. di Pisa,* [d]*INFN, Sez. di Trieste,*
[e]*Scuola Normale Superiore, Pisa,* [f]*Università di Pisa,* [g]*Università di Siena*
E-mail: giovanni.punzi@pi.infn.it

The LHCb experiment is undergoing a major upgrade in view of Run-3, in which the complete detector will be read out, and events fully reconstructed, at the full LHC crossing rate (averaging 30 MHz). One of the key steps of event reconstruction is finding tracks in the new, high precision pixel vertex detector (VELOPIX). This step is the necessary starting point for most of the rest of the reconstruction, and requires a significant fraction (close to a half) of the total CPU time that will be availabe in the upgraded Event Filter Farm. We present the current status of a LHCb R&D project devoted to accelerating this computation by the use of an array of commercial state-of-the-art FPGA cards embedded in the DAQ system, performing pattern recognition in the vertex detector 'on the fly', while the detector is being readout at 30 MHz.

The 28th International Workshop on Vertex Detectors - Vertex2019
13-18 October, 2019
Lopud, Croatia

[*]Speaker.
[†]on behalf of the LHCb Real-Time Analysis project

PoS(Vertex2019)047

1. Introduction and background

The power and sophistication of the data processing in HEP has been steadily increasing throughout the history of the field. This growth has been propelled to a great extent by the quick pace of progress of general–purpose CPUs for the consumer market, providing flexible computing power and data storage at ever–decreasing costs. Today, the slowdown of Moore's law, and the features of a physics landscape calling for more and more precision measurements, have created a need for more effective, specialized computing solutions.

While this is an issue for most future high–intensity HEP programs (as apparent from other contributions to this same conference as well), it is of particular importance and urgency for the upgraded LHCb experiment, due to some of its specificities. LHCb, with its large rate of signal events, small but requiring complex analysis, is facing significant challenges already from the upcoming Run-3 of the LHC, due to start in year 2021 [1].

In Run-3, LHCb is making major changes, not only to its detector, but also to the structure of its data acquisition system, that is going to embrace a real-time analysis model [2]. The rate of reading out the complete detector will jump from about 1 MHz to 30 MHz, as there will be no Level-0 filtering anymore. The increase of luminosity to 2×10^{33} will lead to more complex event with multiple primary collisions, where heavy flavors will be produced in the majority of beam crossings. Each event will therefore need a complete reconstruction and analysis before the trigger can decide whether the crossing is worth saving to permanent storage; even then, the number of events to be stored will be so large that a much greater recourse to a reduced event format ("Turbo") is planned. In those data there will be limited information on which to base an offline re-reconstruction; that means the online reconstruction will need to be particularly good and complete. And it must also be remembered that, due to its physics goals, the p_T thresholds of LHCb need to be kept an order of magnitude lower than in High-Pt experiments, leading to track rates comparable to what they have at much higher luminosities ($\approx 5 \cdot 10^{34}$). All of those effects conjure in creating a particularly large strain on the computing resources of LHCb.

In view of these challenges, the LHCb collaboration is exploring various ways to increase its data processing capabilities by the adoption of more specialized computing solutions, that can relieve part of the load from the shoulders of general-purpose CPUs. One of these solutions is mentioned in LHCb's Expression of Interest for a future upgrades [3], that the LHCC has recently recommended to turn into a TDR on the timescale of a couple of years. The idea there is to implement a highly parallel computational architecture in state-of-the-art FPGA devices to perform reconstruction of *downstream* tracks[1] in real time, before the High-Level trigger process begins, and actually before the event is even built – moving in a direction that might ultimately lead to a reconstruction completely *embedded* within the detector readout.

The reason for the focus on downstream tracking is it being particularly hard to perform on general purpose CPUs, which comes from the large amount of combinatorics involved, and its potential to unlock a significant amount of physics in the decay of long-lived particles and states that decay into them – but the same technology can be applied equally well to any parts of the tracking, or even to non-tracking detectors. While studies are ongoing to fully develop the design of this new ambitious device [4], aimed several years into the future, it is very important to develop

[1] These are LHCb tracks reconstructed with only the forward part of the tracking, with no use of vertex detector hits

and prove the technology by testing it in some smaller and cheaper pilot application that can be realized on a shorter timescale. This is the subject of the present contribution.

As a pilot application, we have chosen reconstruction of tracks within the LHCb's vertex detector (VELO) [5]. This is a crucial part of the detector, detecting charged particle in the region closest to the interaction point, with the purpose of reconstructing primary and secondary vertexes with a spatial resolution smaller than typical decay lengths of *b*- and *c*-hadrons in LHCb. Its reconstruction is the first necessary step of LHCb's reconstruction pipeline, and takes about a half of the total HLT1 time budget by itself, so it is by no means a small application. However, the amount of data involved is just about 10 % of the total, so it can be handled by an FPGA system of modest size and cost, making it an ideal first test-case.

2. General Architecture

The main technology at the core of our system is a highly-parallel architecture for pattern recognition going under the name of "artificial retina" [2] due to its attempt at mimicking some structural features and general principles found in the organization of the natural neural network responsible for fast vision processing in biological life [6]. The *retina* architecture was designed for real-time pattern recognition with very low latencies, avoiding any buffering or time-multiplexing, just as in natural vision systems. This is obtained by a design focused on extreme parallelism and generous use of the huge bandwidths made available by modern FPGA devices [7, 8]. These features lead naturally to a "transparent" implementation within the data acquisition of LHCb, allowing to perform reconstruction "on the fly" while the detector is being read out.

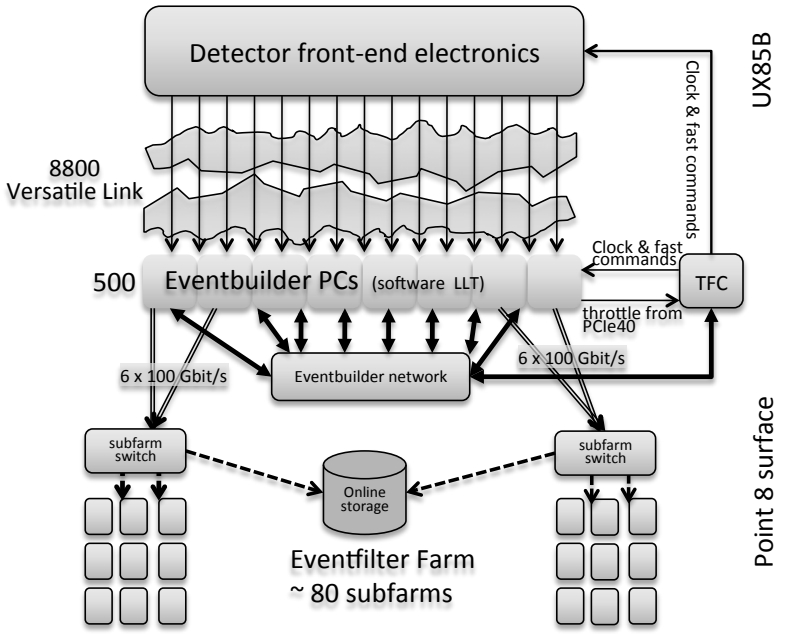

Figure 1: The architecture of Upgraded LHCb readout-system.

[2]For simplicity we will refer to it as the *retina* algorithm in the following.

The general structure of the Upgraded LHCb Data Acquisition, as described in the LHCb online TDR [2], is shown in Fig. 1. The LHCb Event Builder (EB) will be composed of about 500 rack-mounted PC boxes ("nodes"), each equipped with an optical Data Acquisition card (PCIe40), hosted by 16-lane Gen3 PCIe[2]. Each PCIe40 card receives data from the front-end over up to 48 optical links, reading out from a specific piece of the detector located in the experiment cavern. Data from several bunch-crossings are merged into a multi-event fragment packet (MEP) to reduce the message rate and ensure efficient network usage. For each MEP one PC is elected to be the event-builder PC; all non-elected PCs will send their MEP to this PC through the Event-Builder Network. Event-building is then performed by the receiving node by combining all MEP containing data from the same bunch-crossings in a single piece of data. Complete events are then sent via a LAN to the CPU farm (Event Filter Farm, EFF), running the High-Level Trigger software (HLT).

In order to fit our project seamlessly within this system, we need to intercept data at the exit of the PCIe40 card, reconstruct it, and inject in the MEP before it is sent over for building. In this way we can attain our goal of a fully-embedded reconstruction, that remains transparent to the rest of the system. By residing before the event-building process, our system can be made to receive just the data coming from the detectors of interest and nothing more, leading to smaller and more modular computing solutions. We will implement our system as a set of identical PCIe-hosted FPGA cards, each supporting a fragment of both the switch and the processing blocks of the *retina* system, appropriately broken down and distributed across nodes. This leads to a conveniently uniform and scalable system. Cards with the needed features are commercially available today at reasonable costs due to the large diffusion of network/telecommunication technology. They have the low-latency response needed to operate within the EB requirements, and are easily integrated in the EB by insertion in an available PCIe slot in each EB node connected to the detector of interest.

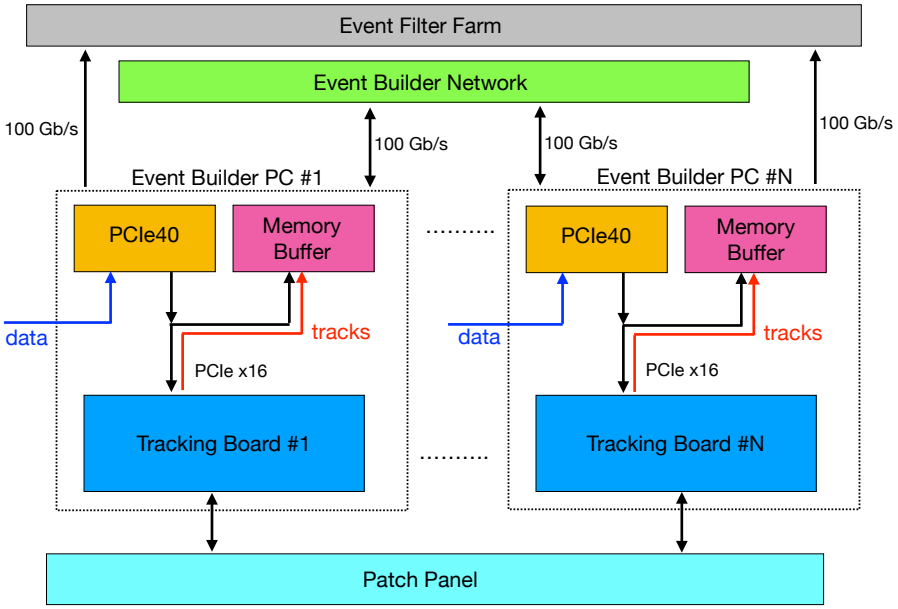

Figure 2: Scheme of processing system integrated in the Event Builder architecture.

Each of our cards (*Tracking Boards* in Fig. 2) needs a copy of the information delivered by the PCIe40 card, will process it through the *retina* structure collectively formed by whole array of boards, receive back the necessary data for some specific piece of the parameter space, and re-inject such data via PCIe in the buffer memory of the PC, for merging with the MEP for that event. The data output by each card of our system maps a specific piece of the track parameter-space, while its input data and the remainder of the event with which it is merged are mapped to a geographical portion of the detector. In order to perform this collective process, our FPGA cards need to communicate quickly among themselves in a manner independent from the EB. We achieve this by external optical connections, using an architecture bearing a close similarity with the "Catapult" system designed by Microsoft to accelerate *Bing* on clouds with distributed FPGA accelerators[9]. All optical connections will be routed via a passive patch panel (bottom of Fig. 2).

3. Implementation

As part of the LHCb upgrade, the current VELO will be replaced by a new detector, based on silicon pixel technology [5]. The new VELO will consist of 52 modules positioned along the beam axis, both upstream and downstream of the nominal interaction point. Pixels are $55\mu m \times 55\mu m$ in size, but a particle crossing a VELO layer will often activate more than one, so that to reconstruct the position of the hit several contiguous pixels may need to be grouped into a cluster. This is currently achieved by a clustering algorithm running as part of the HLT1 reconstruction code, but in order to execute our project of reconstructing VELO tracks upfront of the HLT1, we need to realize this function in the FPGA as well. We therefore had to develop a piece of firmware for this specific purpose. This has become a significant part of the overall project because, although simpler than tracking, it is still non-trivial to perform clustering in a 2-dimensional space at the rate needed to process the whole VELO at the full crossing frequency of the LHC. In fact, it takes some non-negligible fraction of the HLT1 CPU-time budget to perform (about 20%). This has already been described elsewhere [10], and will be the subject of a full paper to appear, so it will only be briefly summarized here.

3.1 Clustering of VELO pixels with FPGA

The main idea to make the algorithm fast enough was to allocate individual logic units to each pixel, and have them communicate with each other to perform the cluster pattern recognition in a fully parallel way; this exploits some of the same concepts of the *retina* tracking. The 2D clustering firmware we produced has rather general applicability, and we made the core code available via an open-access repository [11].

Pixel data are read out from the VELO in the format of 2×4 blocks, called SuperPixels (SP). Clusters produced by real particles typically consist of just a few pixels (1-4). Due to this, a significant fraction (about 2/3) of the clusters are contained within a single SP ("isolated SuperPixels"). This makes it convenient to deal with this type of clusters separately. During readout, isolated SPs are flagged, and for each of the 256 possible SP configurations, a pre-calculated cluster position is available in a lookup table (LUT). In this way, reconstructing clusters contained in a single SP requires a very small amount of FPGA resources and is very fast. Finding clusters involving SPs

with neighbors (i.e. active SPs next to at least an active SP) requires instead to actually execute a clustering algorithm, and are dealt with separately.

Given the large number of pixels in the VELO, and the fact that in each individual event only a small fraction of pixels are activated ($< 0.1\%$), we gave up on allocating a large matrix encompassing the 4.1×10^7 pixels of the VELOPIX detector, and created instead a limited set of smaller matrices (5×3 SPs, i.e. 10×12 pixels), to be mapped in real-time to the relevant locations of the detector. Inside each matrix, for every pixel configuration matching a pattern, the 3×3 cluster candidate is then resolved by a lookup table. This makes our firmware significantly more compact, and a good candidate to be fitted inside the readout FPGA itself, thus realizing not only an economy of space and time, but also a greater adherence to our long-term vision of embedding data reconstruction within the readout chain to the maximum possible extent.

This algorithm has been implemented in VHDL language, and deployed to a development board endowed with two large Intel Stratix-V FPGAs, each holding about 1 MLE. Simulated pixel data produced by the official simulation of the VELO detector have been stored in a circular buffer and fed to the clustering firmware in a continuous loop, with tunable frequency. The output is buffered and compared bit-by-bit to the output produced by a detailed high-level software emulation of the algorithm. Using data coming from the busiest silicon sensors (those closest to the beamline) it was found that our clustering firmware is capable of running without errors up to a sustained rate of of 39 Millons of events/s. This is an underestimate of the performance we will have in the actual application, that uses a faster FPGA than the Stratix-V model used in this test.

This has established the technical feasibility of doing cluster reconstruction in firmware at the LHC crossing rate, and integration of our code with the VELO readout firmware that will run in the PCIe40 readout cards in Run-3 is now in an advanced stage. However, verification of physics performance is also necessary, since our algorithm has some intrinsic differences from the CPU implementation. For this purpose, a bit-level simulation of the FPGA clustering algorithm has been written and integrated in the official LHCb simulation and reconstruction software environment. In this way it has been possible to feed the clusters produced by our algorithm to the official HLT1 tracking code, and compare high level performance measurements like reconstruction efficiencies, clone and ghost rates with those obtained with the standard CPU-based clustering code, using exactly the same comparison code and criteria [12]. In order to stress-test the algorithm, we processed a sample of 25000 minimum-bias events simulated with the nominal conditions of the LHCb upgrade: center of mass energy $\sqrt{s} = 14\,\text{TeV}$ and luminosity $\mathscr{L} = 2 \cdot 10^{33}\,\text{cm}^{-2}\text{s}^{-1}$. We looked not only at the average efficiency of all clusters, but also specifically at the first three hits on each track, that have the highest relevance to the accuracy of determination of the track vertex of origin, that is of course the most crucial quality parameter in heavy flavor studies. The results, displayed in Table 1, show that the tracking performances obtained using our FPGA clustering algorithm differ by just fractions of a % from the full-fledged CPU algorithm, making them essentially equivalent for all practical purposes.

Further performance checks and final verification of compatibility with the LHCb readout are now being performed, ahead of a decision on whether to adopt this firmware as the new Run-3 baseline for cluster finding in the VELO detector. This is independent of the possible use of clustering in a future FPGA-based tracking project, whose status is described in the next section.

		VeloClusterTracking (CPU)	FPGA + VeloClusterTracking
Velo tracks	**efficiency**	95,519% ± 0,014%	95,258% ± 0,014%
	clone	1,365% ± 0,008%	1,421% ± 0,008%
	hitEffFirst3	95,21%	94,51%
Long tracks	**efficiency**	97,705% ± 0,013%	97,493% ± 0,014%
	clone	1,291% ± 0,010%	1,361% ± 0,010%
	hitEffFirst3	95,70%	94,96%
Ghost tracks		0,8745% ± 0,0041%	1,0217% ± 0,0045%

Table 1: Summary of physics performances of the HLT1 tracking algorithm for different types of track, using clusters produced inside HLT1 and the FPGA. VELO-tracks are defined as having clusters on three or more VELO layers. Long tracks additionally have at least one x and one *stereo* cluster in each tracking station downstream the LHCb magnet, and are the most commonly used tracks in LHCb physics analyses.

3.2 Track Pattern Recognition

Having established a way to perform clustering in real time enables us to consider the possibility of performing track pattern recognition in FPGAs as well. This is the heaviest part of HLT1 reconstruction; together with clustering, it consumes about half of the HLT1 time budget.

The complete VELO is composed by 52 modules, arranged in 26 layers. Each module is read from a separate readout card into the EB, so that there are 52 readout units for the VELO. For the purpose of the current studies aimed at a pilot system that can be implemented within a limited time and budget, we elected to use only data from layers 8 to 26. Those modules are positioned at positive values of z, where $z = 0$ corresponds to the nominal interaction point (Figure 3), and they hold most of the data relevant to the reconstruction of forward-going tracks produced in the vicinity of the primary interaction, that are the crucial ingredient for the HLT1 trigger. The remaining 7 layers located at negative values of z are important for the reconstruction of segments of tracks going in the backward direction, whose main use is to optimize the accuracy of the determination of the primary vertex, and are dealt with separately in the current CPU reconstruction software. We decided to postpone their processing to a separate study.

3.2.1 Optical network

Obviously, in order to find tracks it is not sufficient to work on local detector information, but it is necessary to move data between different modules appropriately. We do this over a fast optical network, according to a plan dictated by the *retina* architecture. While the high-level design of such network is well understood, its practical realization is not without challenges, and a demonstration of its feasibility was identified as one of the main milestones during LHCb's internal review of the project, so it is a significant focus of our activity. The plan is to connect the 38 cards of the VELO system to build 4 or 5 sub-nets connected by limited–bandwidth bridges. Each sub-net will be a 8–node full-mesh bi-directional networks of optical link, each carrying between 12 and 28 Gb/s of data in each direction. This is not too different from the network topology of the previously mentioned CATAPULT system [9]. In order to demonstrate the feasibility of this network, a test has been performed on a prototype of one such sub-net, which is the crucial part to demonstrate due to the large density of connections. The test has been performed on the same development platform used for the clustering tests (with 96 bidirectional optical links), by implementing 4 virtual nodes on each physical FPGA device.

The test successfully demonstrated operation of the whole network up to its full bandwidth capacity (in excess of 1 Tb/s), with low Bit Error Rates (typically $< 10^{-16}$), low connection latency ($\simeq 250$ ns), and no reduction of the throughput of the overall RETINA processing. Performance was unaffected even when large fixed skews were introduced between the inputs, to test the robustness of the system to timing skews in the inputs. This is a very promising step towards the final demonstrator, currently being assembled within the LHCb DAQ Vertical Slice, that will verify the behavior and stability of the system in conditions as close as possible to the actual running conditions of Run-3.

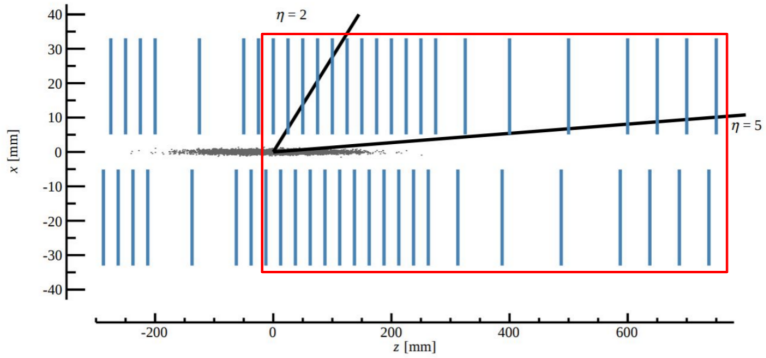

Figure 3: Layout of the upgraded VELO. The detector consists of two retractable halves, each housing an array of 26 silicon pixel detector modules. The red rectangle indicates the modules of interest for forward-track reconstruction

3.2.2 Parameter space matrix

The intrinsic 2D nature of the RETINA architecture makes it complex to operate in a large multi-dimensional parameter space. We therefore segmented the 3D track parameter space in 10 slices running along the z axis. In this way we can have sufficient reconstruction acceptance over the entire (rather wide) distribution of primary vertex along the LHCb beam crossing point. We instantiate several retina matrices, each tuned to a different z-slice, and each of them running the RETINA algorithm in parallel, independently of the others. Each retina matrix represents a 2D piece of the parameter space of a track, using for coordinates the x and y intercept of the track with a fixed-position reference layer. Tracks with different z_{vtx} may happen to be reconstructed in more than one z-slice; the possibility of multiple solutions is avoided by having each matrix perform a check on the z of minimum distance to beam of every found track, to ensure it belongs to the correct fiducial region. As a final step, remaining ambiguities need to be sorted out and possible spurious hits rejected before performing the final fit of track parameters. This might be better left to the HLT1 with the benefit of the most accurate alignment information; a decision has not yet been made on this point.

The size of the acceptance region in z depends on the individual cell sizes, that in turn is constrained from below by the need of limiting the needed amount of FPGA resources, and constrained from above by the multiplicity of hit combinations/ambiguities within each cell. We choose to allocate to each slice a matrix of $100 \times 100 = 10000$ cells, as this produces a good uniformity of acceptance along z and, at the same time, a limited number of multiple combinations within the

same cell. There is still a significant space of possible configurations to search, and the setting we currently use might be further optimized in future.

To test the physics performance of our system we developed a standalone, bit-level software emulation of our retina processor. As for the clustering emulator, this software package doubles as a diagnostic/monitoring tool (for experts) and as a detailed simulator for physics analysis users. We wrote another small piece of code to import the list of track candidates produced by our emulator in the LHCb framework; the final combinatory reduction is then run, and the produced track is finally formatted and plugged into the standard LHCb HLT1 sequence. The rest of the standard reconstruction is then run, using our own tracks in place of the tracks produced by the CPU-based VELO reconstruction. Again this allows us to run exactly the same performance diagnostics on exactly the same samples, switching between the two reconstruction methods. We define a fiducial region where we expect uniform acceptance. For the current configuration, this is $2 < \eta < 5$ and $-20\text{ cm} < z_{vtx} < 20\text{ cm}$, where η is the pseudorapidity. We have additionally restricted the set of tracks to those with at least 5 hits, as those with less hits are of lower quality and rarely enter physics analyses.

The performances obtained on a sample of simulated $B_s \to \phi\phi$ decays (Table 3.2.2) demonstrate that the HLT1 algorithm performance is only negligibly affected when running on tracks produced by FPGAs. Even if we include marginal quality tracks with less than 5 hits, the efficiency of the FPGA reconstruction is still within less than a percent from the CPU reconstruction in the majority of cases.

One adverse effect of the multiple z-slices is the creation of a larger number of clones than in the CPU case. This is well understood, and it is due to the fact that we have not yet introduced any information exchange between the different z-slices, so some tracks get duplicated in more than one slice. We plan to deal with this effect within the same solution we will use to deal with the residual ambiguities mentioned previously.

Track type	ε CPU pat-reco (%)	ε FPGA pat-reco (%)	
		all z	fiducial z
Long tracks with $p > 5$ GeV/c and hits in VELO> 5	99.84 ± 0.02	99.27 ± 0.06	99.45 ± 0.05
Long tracks from b with $p > 5$ GeV/c and hits in VELO> 5	99.61 ± 0.13	99.24 ± 0.21	99.41 ± 0.18
Long tracks from c with $p > 5$ GeV/c and hits in VELO> 5	99.89 ± 0.12	98.50 ± 0.53	98.62 ± 0.53

Table 2: Summary of efficiencies of the VELO tracking algorithm for different type of tracks using both the CPU-based and the FPGA-based pattern recognition algorithm. Numbers obtained on 1000 $B_s^0 \to \Phi\Phi$ events. The efficiency is calculated using Long tracks with $2 < \eta < 5$, $p > 5$ GeV/c and with more than 5 hits (Monte Carlo truth) in the VELO detector. Tracks belong to the fiducial region if the z coordinate of the origin vertex is located between -200 mm and 200 mm.

4. Conclusions and Outlook

After several years of R&D, we are now getting close to the first realization of a FPGA-based 30-MHz tracking, based on extreme-parallelization "retina" architecture. Detailed simulation studies show that this extremely fast tracking is achieved with quite good tracking performance, and there is still more room for optimization.

The whole system is implemented with commercially available parts, that makes it easily upgradable as FPGA devices continue to improve and get cheaper. The approach seems promising for applications to Phase-II LHCb real-time reconstruction needs and other future experiments.

Integration of the optical network prototype has started in the LHCb DAQ Vertical-Slice test setup. Extensive tests will be carried out over the next few months, that are expected to provide precious on-the-field experience with this novel methodology.

5. Acknowledgements

This work would not have been possible without the generous support of INFN (also via the targeted R&D project 'RETINA'), of Scuola Normale Superiore (Pisa), and the active cooperation of the members of LHCb's Real Time Analysis, Online, and VELO projects.

References

[1] The LHCb Collaboration, *Framework TDR for the LHCb upgrade*, CERN/LHCC 2012-007, 2012.

[2] The LHCb Collaboration, *LHCb Trigger and Online Upgrade Technical Design Report*, CERN/LHCC 2014-016, 2014.

[3] The LHCb Collaboration, *Expression of Interest for a Phase-II LHCb Upgrade: Opportunities in flavour physics, and beyond, in the HL-LHC era*,CERN/LHCC 2017-003, 2017.

[4] M.J.Morello *et al.*, *Real-time reconstruction of long-lived particles at LHCb using FPGAs*, presented at ACAT 2019, to appear in J. Phys.: Conf. Ser.

[5] The LHCb Collaboration, *LHCb VELO Upgrade Technical Design Report*, CERN/LHCC 2013-021, 2013

[6] L. Ristori, *An artificial retina for fast track finding*, Nuclear Instruments and Methods in Physics Research A453 (2000) 425.

[7] A. Abba *et al.*, *A specialized track processor for the LHCb upgrade*, Tech. Rep. CERN/LHCb-PUB 2014-026, 2014.

[8] R. Cenci *et al.*, *Development of a High-Throughput Tracking Processor on FPGA Boards*, PoS(TWEPP-17) 136.

[9] A. Caulfield *et al.*, *A Cloud-Scale Acceleration Architecture*, in Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture, 2016.

[10] F. Lazzari *et al.*, *Real-time cluster finding for LHCb silicon pixel VELO detector using FPGA*, presented at ACAT 2019, to appear in J. Phys.: Conf. Ser.

[11] G. Bassi *et al.*, *FPGA implemention of a fast 2D clustering algorithm* doi:10.15161/oar.it/23524

[12] The LHCb Collaboration, *LHCb Tracker Upgrade Technical Design Report*, CERN/LHCC 2014-001, 2014