LHCb-PUB-2014-026
May 23, 2014

A specialized track processor for the LHCb upgrade

A. Abba[1], F. Bedeschi[2], F. Caponio[1], M. Citterio[1], A. Cusimano[1], A. Geraci[1], F. Lionetto[2], P. Marino[2], M.J. Morello[2], N. Neri[1], D. Ninci[2], A. Piucci[2], M. Petruzzo[1], G. Punzi[2], F. Spinella[2], S. Stracka[2], D. Tonelli[3], and J. Walsh[2],

[1] *INFN Sezione di Milano and Politecnico of Milano, Milano, Italy*
[2] *INFN Sezione di Pisa, Scuola Normale Superiore, and University of Pisa, Pisa, Italy*
[3] *CERN, Geneva, Switzerland*

Abstract

We propose a specialized processor dedicated to efficient real-time reconstruction of charged-particle tracks in the upgraded LHCb detector. The main purpose of this *track processing unit* is to relieve the event-filter farm from the repetitive, mechanizable calculations associated with the pattern-recognition task, providing it with more time for higher-level trigger functions. We develop a design based on a biology-inspired pattern-recognition algorithm implemented into modern field-programmable-gate-array devices, yielding high-quality tracking at the full crossing rate, with submicrosecond latencies. This document describes the concept, technical implementation, simulation, performance, and costs of the project.

LHCb-PUB-2014-026
23/05/2014

Contents

1 Introduction

The operation of the LHC in 2010-2012 (Run 1) has provided the LHCb experiment with the world's richest samples of heavy flavor decays. The data are being analyzed to pursue the most complete and precise experimental program in the study of quark-flavor physics to date. Such a remarkable achievement is made possible by the excellent performance of a well-designed, dedicated detector and its trigger, which allows exploiting the high heavy-flavor production rate in the forward kinematic region of high-energy proton-proton collisions.

Among all collider experiments, LHCb features the unique ability to set its own instantaneous luminosity. This is crucial for flavor physics, as the amount of data produced by the LHC exceeds the capability of acquisition and analysis of any existing apparatus. The instantaneous luminosity reached values up to $L = 4 \times 10^{32}$ cm^{-2} s^{-1} during Run 1. In order to further and strengthen the physics program, the LHCb collaboration has proposed an extensive upgrade of its detector and DAQ system, aiming at efficiently operating at $L = 2 \times 10^{33}$ cm^{-2} s^{-1} with center-of-momentum energy $\sqrt{s} = 14$ TeV in Run 3, which is currently expected to start around year 2020.

In addition to detector upgrades, this goal requires a major restructuring of the trigger. The Run 1 LHCb trigger was organized into two decision levels [1,2]. A hardware level-zero (L0) trigger based on coarse transverse momentum information from the muon chambers (muon trigger) and transverse energy information from the calorimeters (hadron, electron, and photon trigger) had a fixed latency of 4 μs to reduce the rate from the 20 MHz bunch-crossing rate to 1.1 MHz. A two-staged high-level trigger based on `C++` and `python` computer code running in parallel on a CPU farm, reduced the accept rate of 60 kB-sized events down to 2-5 kHz for storing on permanent memory.

This approach is not adequate for the higher luminosities and collision energies of Run 3. Figure 1 shows that the L0 hadron triggers already suffered significant inefficiency due to the 1.1 MHz rate limitation, and would need a twenty-fold increase in accept rate to saturate the efficiency for collecting benchmark physics signals [3]. This impacts most severely the charm program, with nearly 80% of hadronic two-body charm signal rejected by the L0 hadron trigger in Run 1, but has significant effects for all channels with fully hadronic final states. At increased LHC collision energy and luminosities, these performances can only get worse.

To gain a significant improvement in physics reach from the conditions of increased luminosity and energy of the upgraded LHCb in 2020, the collaboration decided that relying on online selection requirements applied to simple, coarsely measured quantities is inefficient. Digitally reading out the whole detector at the full LHC crossing frequency is expected to provide the maximum gain. This would allow trigger selections based on rich, high-quality information from the whole detector that can be nearly as efficient as the offline selections. In addition, such selection based on digitized information can be very accurately simulated and reproduced. Such approach is also expected to be less prone to hard-to-model selection biases that can impact systematic uncertainties of the high-precision measurements foreseen for the upgrade era.

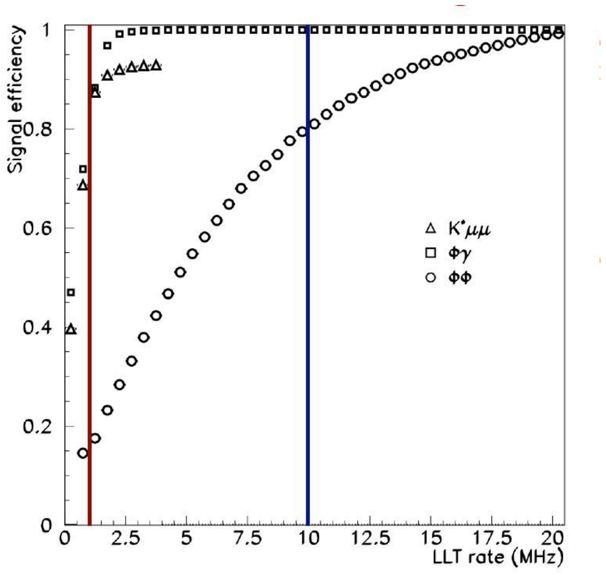

Figure 1: Efficiency of the LHCb low-level trigger on representative simulated signals as a function of event accept rate at instantaneous luminosity $L = 1 \times 10^{33}$ cm^{-2}s^{-1}. Plot reproduced from Fig. 3.2 in Ref. [3].

However, this approach imposes an enormous strain on the event-filter farm, which faces the challenge of crunching a 30 times larger rate of events, each involving a greater complexity due to higher luminosity and energy. At the same time, the rate-reduction factor must increase to maintain the output data rate within allotted constraints, and the quality of reconstruction must increase as well, to serve the needs of measurements of increasing precision.

The aim of the present proposal is to help LHCb reaching its goals by relieving a portion of the workload on the event-filter farm and moving it to a specialized track-reconstruction processor.

While some of the trigger-selection tasks executed in the farm necessarily require the flexibility and complexity that only a general-purpose CPU can provide, some other "mechanical" and repetitive computations can be very conveniently performed by specialized logic. The fast reconstruction of charged-particles trajectories (tracks) is a heavy task in hadron collisions, due to the need to perform massive pattern recognition in a very short time, and poses significant conceptual and technical challenges. Luckily, the uniformity of the associated algorithms fits very attractively an efficient *co-processor* solution, as demonstrated by this approach's success achieved in other experiments. Devices able to reconstruct tracks online in hadron collisions were employed since the early eighties [4]. In the nineties, the Collider Detector at Fermilab experiment used pattern-matching algorithms implemented into field-programmable-gate-arrays (FPGA) to reconstruct two-dimensional tracks from clusters of geometrically-aligned hits in the central tracking drift-chamber [5, 6]. A major breakthrough was brought in 2001 with the silicon vertex trigger [7–10], which implemented pattern-matching using a custom-made processor, the *associative-memory*, that connected the drift-chamber tracks with silicon information and made available two-dimensional tracks with offline-like resolution within the 20 μs latency of the second level of CDF's three-staged trigger. The associative-memory success over standard processors was due to the capability of processing in parallel all relevant track templates in the event. This allowed reconstruction of full events at a rate of up to 100 kHz, making it an ideal device for CDF's level-2 trigger able to match the full level-1 output rate of approximately 30 kHz.

A modern revamping of the same ideas, the FTK tracker device, is currently being implemented in the second trigger level of the ATLAS experiment [11, 12]. Thanks to more modern electronics and a strongly pipelined architecture, the ATLAS fast-tracker handles a much larger number of patterns ($> 100\times$) and processes the much more complex ATLAS events at a 100 kHz rate, with $\mathcal{O}(20)\mu$s latencies. At the time of this writing, a similar approach is being explored by the CMS collaboration as well.

The requirements for a similar device to be useful in the LHCb upgrade are even harsher than those associated with the above, due to the desire of reading out the whole detector at the full crossing frequency. In this scenario, there is no use for a specialized tracking processor unless it is capable of a throughput of 40 MHz as well. However, if such a device could be available it could greatly empower the physics impact and perspectives of the LHCb upgrade program. Having tracks reconstructed efficiently on the fly at the LHC crossing rate, as if they were produced directly as raw data from a virtual detector, is a game-changer capability. It will provide a leap toward the LHCb goal of achieving a thoroughly sophisticated online reconstruction that closely approaches the offline quality and will strongly accelerate the speed at which LHCb will be able to ramp up in luminosity and maintain leadership over concurrent experiments.

We present here the *track processing unit* (TPU), a proposal to address this challenge using a novel, massively parallel algorithm, inspired by the neurophysiology of the human vision [13]. This algorithm does not require a custom ASIC. It lends itself very naturally to a firmware implementation using commercially available FPGA chips, taking full advantage of the huge increase in power and internal bandwidth of the latest generation of these devices. Our approach follows current industry trends for high-end, computing-intensive applications as found in radars, CT scanners, or high-frequency trading. There, traditional CPUs are supplemented by FPGA coprocessors dedicated to highly parallel, bandwidth-intensive tasks, for which they are widely regarded as the most cost-effective solution. Our design exploits the same FPGA devices planned for use within the new LHCb readout system [14]. This ensures a seamless fit within the upgraded DAQ structure, and effectively makes the TPU a purely software (firmware) project, allowing the greatest flexibility and avoiding the introduction of any extra development or maintenance workload.

This document is structured as follows: Section 2 describes the conceptual design of the main TPU function; Section 3 details the technical implementation of the device, its fit in the LHCb readout architecture, and its timing performance; Section 4 describes the high-level simulation of the TPU and its tracking performance; projected benefits for the LHCb upgrade are discussed in Sec. 5; Section 6 provides a detailed estimate of the amount of needed hardware and costs; a summary is reported in Sec. 7.

2 Conceptual design

Fast tracking in a busy environment poses two main challenges: (i) resolving the pattern-recognition problem and (ii) distributing efficiently the needed information from the detector to the processors devoted to it. The conceptual solution chosen for the TPU are discussed in what follows.

2.1 The *artificial retina*

Achieving our goal of real-time tracking at the LHC crossing rate, with no trigger layer in-between, requires to design a device with a throughput 400 times greater than the already blazingly-fast associative memory. Hopes of success without a radical change of approach look slim. In fact, clock speeds accessible to associative-memory-based devices built to date, show that the event-rate–to–clock-period ratio is remarkably constant at around 10^3 cycles per event. For comparison, unspecialized CPU-like architectures of similar technology typically require about 10^8 cycles per event. What is needed for the LHCb trigger is a tracking machine capable of processing each event in a bare ≈ 25 cycles, assuming a clock frequency of 1 GHz. Our proposal is based on the so-called *artificial retina* algorithm. This algorithm was proposed in 1999 [13] as a possible solution for very fast, massively parallel, track reconstruction, based on what is believed to be the most likely low-level mechanism used by the mammalian visual brain areas to recognize lines and edges [15, 16]. This algorithm can be efficiently applied to find tracks sampled by a multilayer detector and extract their parameters.

A simple example of a single straight track traversing an array of n parallel detector layers is useful to illustrate the concept. The parameter space of the track is divided into a grid of receptive fields, the *cells*. The metric and dimensionality of this space depends on the problem at hand. In our simple model the track has two parameters, (p, q). Each parameter-space cell is uniquely associated with a set of coordinates (p_i, q_j). These coordinates correspond to the *intersections* that a track with parameters identified by the center of the cell has in the measurement planes. For each incoming hit, the algorithm computes the excitation intensity of the cell corresponding to (p_i, q_j)

$$R_{ij} = \Sigma_{k=1,r}^{k=n} \exp\left(-s_{ijkr}^2/2\sigma^2\right) \tag{1}$$

using the distance

$$s_{ijkr} = x_r^{(k)} - y_k(p_i, q_j) \tag{2}$$

between the hit position $x_r^{(k)}$ and the intersection of the (p_i, q_j) track $y_k(p_i, q_j)$ on layer k. The sum runs over all hits in all layers and the excitation R_{ij} is computed for all cells. The weight parameter σ is adjusted to optimize the sharpness of the response of the receptors. In this example a Gaussian excitation function is adopted for simplicity, but

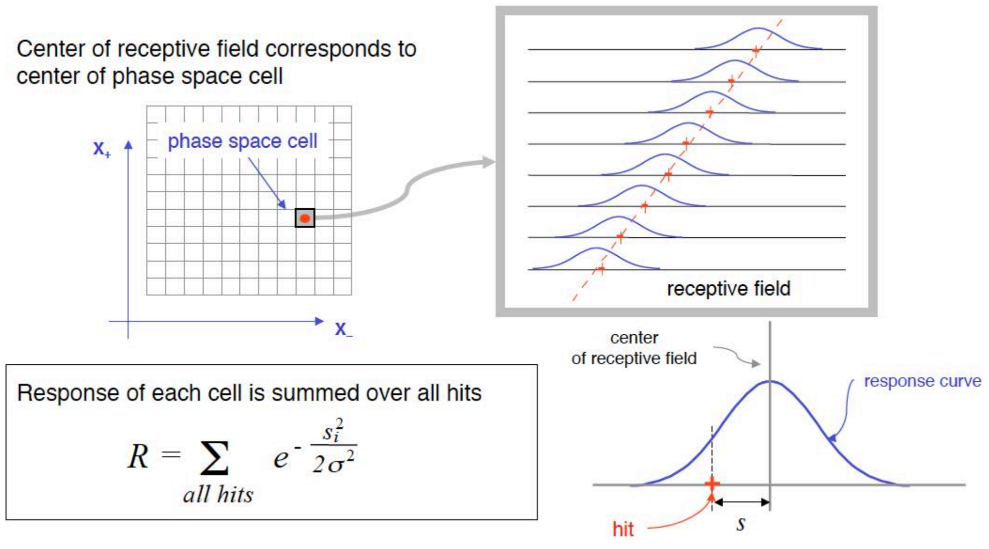
Center of receptive field corresponds to center of phase space cell
x_+
phase space cell
x_-
receptive field
center of receptive field
response curve
hit
s
Response of each cell is summed over all hits
$R = \sum_{all\ hits} e^{-\frac{s_i^2}{2\sigma^2}}$

Figure 2: Illustration of the retina-algorithm concept. (Reproduced with permission from Ref. [17]). The (x_+, x_-) labels that appear in the top-left quadrant correspond to the coordinates we refer to as (p, q) in the text.

other arbitrarily optimized functions can be used. After all hits are processed, tracks are identified as local maxima in the cell space, via a simple, local cluster-finding algorithm. Hence, for each incoming hit in an event, the algorithm computes the weighted distance of that hit from each intersection. The weighted distances are summed over the nearest cells and all event hits. The resulting set of excited cells is appropriately clustered and zero-suppressed to derive the corresponding tracks.

The algorithm relies on extensive parallelism and interconnectivity. The weighted interpolation allows preserving the native resolution on track parameters, while keeping the cell granularity, hence the hardware size of the system, reasonably small. The grid pitch can be significantly larger than the native detector spatial resolution, which can be recovered provided that a sufficient amount of bits are used to store each cell's response. The retina algorithm features several analogies with the Hough transform technique [18]. The capability of a continuous response function and the fully parallel implementation, however, offer significant additional advantages. In what follows we describe the design of a realistic real-time tracking system implementing this algorithm, capable of performing a significant portion of the LHCb tracking, and evaluate its performance.

2.2 Architecture

The functional architecture of the proposed TPU (Fig. 3) mirrors this conceptual description. The array of receptor cells is mapped into an array of cellular processors. Each processor evaluates and accumulates the excitation of one or more cells. Hits flow from the

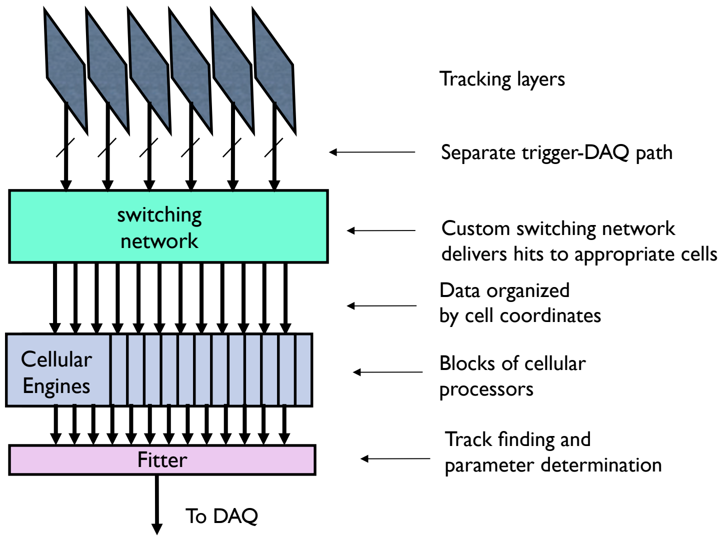
Tracking layers
Separate trigger-DAQ path
switching network
Custom switching network delivers hits to appropriate cells
Data organized by cell coordinates
Cellular Engines
Blocks of cellular processors
Fitter
Track finding and parameter determination
To DAQ

Figure 3: TPU architecture overview.

detector planes into a switching network, that delivers each hit to all relevant processors in parallel. Each cell is implemented as an independent block of logic (*processing engine*) that performs the necessary elementary operations. Local maxima are found in parallel in all processors, with some exchange of information between adjacent processors. The coordinates and intensity of the local maxima, and the intensities of their nearest neighbors are output sequentially. Finally, track parameters are extracted from the cell information.

The whole processing is envisioned to happen in a time short enough that it effectively appears to the rest of the DAQ as if tracks are coming out of the detector at the same time as the hits and all other raw data. Input data must be a duplicate of the information flowing directly to the DAQ – and eventually the reconstructed TPU tracks that are reconstructed are made available to the trigger and DAQ system, as if the TPU were an additional subdetector, only providing tracks as raw data rather than simple hits. This approach allows the maximum possible flexibility in the use of the track information, with the minimal perturbation of the rest of the architecture. At the time this architecture was first considered, the required massive parallelism was neither easily implementable, nor really necessary for the needed applications. Today, however, with the progress in digital electronics, and the more ambitious goals we are aiming at, this approach turns out to be feasible and very effective.Implementation is described in the next section.

3 Implementation

The TPU approach to tracking is flexible and linearly scalable with the number of tracking layers one wishes to use, allowing various possible tracking configurations for a retina-based system. As a use-case, we focus on the VELO-UT configuration in the present proposal, due to the central importance VELO-UT reconstruction assumes in the current upgrade-HLT strategy [19]. In this section we describe the organization and integration in the DAQ, of a system capable of tracking using **two telescopes**, each consisting of

- eight layers of the microvertex VELO detector and
- two additional silicon layers that measure the axial coordinates of tracks just upstream of the magnet, in the UT detector.

Other configurations for the TPU can be easily imagined, *e.g.*, including parts of the forward tracking, or even specifically dedicated to the reconstruction of downstream tracks.

3.1 The switch

A crucial ingredient for TPU tracking to be realizable in practice is a system for distributing in real time the hit information coming from the detector layers to the array of processing engines. Given the 40 MHz throughput and the large flow of several Tbit/s of data from the tracker, this is a nontrivial task. The associative-memory approach of delivering every hit of every event to every cell would be hard and expensive. We therefore design an intelligent delivery system, with embedded (programmable) information allowing each hit to be delivered in parallel to all engines for which such hit is likely to contribute a significant weight, but no more than those, and allowing this to happen in parallel for all hits, without interrupting the flow. This functionality is similar to that of commercial network switches, although more specialized. The most effective choice for its implementation is the same, high-end FPGA devices of the latest generation that are especially designed with an emphasis on large internal bandwidth. A few definitions may help in following the switch description (see Fig. 4):

- A *group* is a geographic area in each detector layer; each hit is assigned to one and only one group, through a *group number* based on hit coordinates and stored in a lookup table;
- each hit belonging to a group is delivered to the union of all the engines affected by all the hits in that group, that is the parameter-space *region* corresponding to that group;
- hence, a *region* is the union of all cells in the track parameter space for which hits in a group contribute a significant weight;
- Groups are defined to optimize distribution, compromising between the overlap between regions and the granularity of the classification;

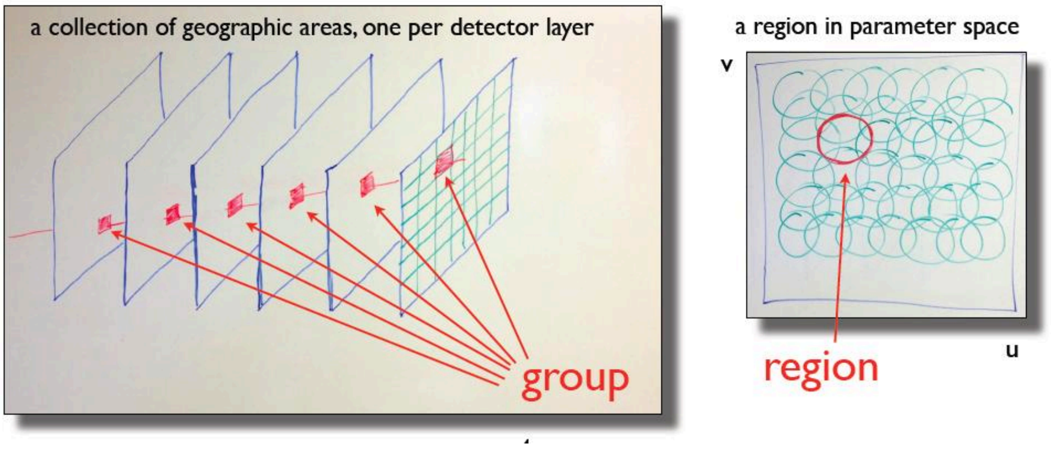
a collection of geographic areas, one per detector layer
group
a region in parameter space
v
u
region

Figure 4: Illustration of group and region.

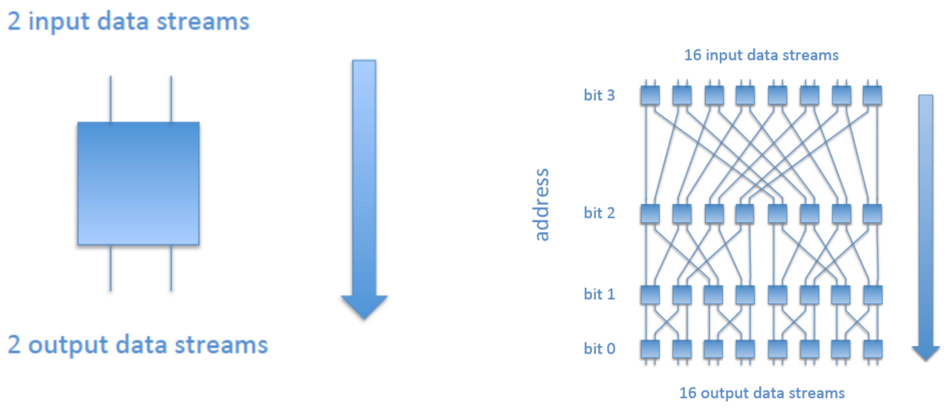

Figure 5: Basic logic unit of the switching network.

- Each hit is dispatched and duplicated as needed, according to its group number. The definition of the parameter region corresponding to each hit group is embedded in the switch, in a distributed form amongst its nodes.

The switch is built from a network of nodes. The basic building blocks are two-way sorters (Fig. 5, left), with two input data and two output data streams.

A two-way sorter merges left and right input data and dispatches hits to one or both outputs according to the group the input hits belong to. Left and right inputs are merged. If a stall from downstream layers occurs, one or both input streams are held. Such elementary building blocks are combined to build the needed network topology, with the

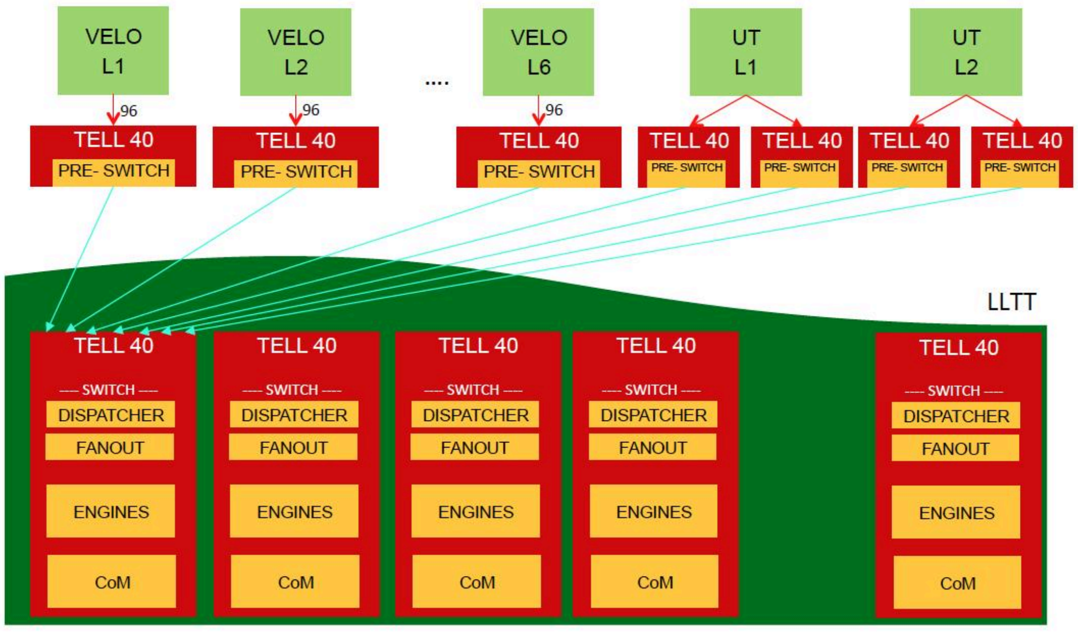

Figure 6: Organization of the TPU and connections to VELO and UT readout in the AMC/TELL40 scheme.

required switching capability (Fig. 5, right). An $N \times N$ network requires $\log_2(N)N/2$ elements. The modular structure has several advantages over a monolithic design. It allows easy scalability and reconfiguration of the system when necessary, and distributing the necessary addressing information over the whole network, storing the information only at the nodes where it is required.

Figures 6 and 7 show the integration of the system within the LHCb data flow in the model in which AMC/TELL40 or PCIe boards are used, respectively. Data flow out of the detector on GBT fibers, and are received by a layer of Altera Stratix-V chips that provide necessary reformatting and/or time reordering before being sent out to the DAQ. For maximum efficiency, the switching process starts inside this first layer of devices, right after the initial formatting and before sending the duplicated data stream out to the TPU. We therefore divide the switch network in a pre-switch stage, residing in the formatting chips, and a proper switch stage, residing in the same chips that perform the retina algorithm calculations. Connections are provided by a network of optical links, properly organized to provide the necessary fanout function. Each VELO layer is read out by four AMC40 boards, each UT layer by eight AMC40. Each tracking plane is divided into 24 equal (x, y) cells in the plane perpendicular to the beam. To sustain the data flow generated by the VELO and UT detectors, approximately 96 fibers per layer are needed. Therefore, reading out the information associated with the same (x, y) cell for the total of 10 detector layers associated with each of the two TPU telescopes, requires 40 fibers. Each readout AMC40 has 24 free fibers. These are used to transmit the data to the 36 inputs of the

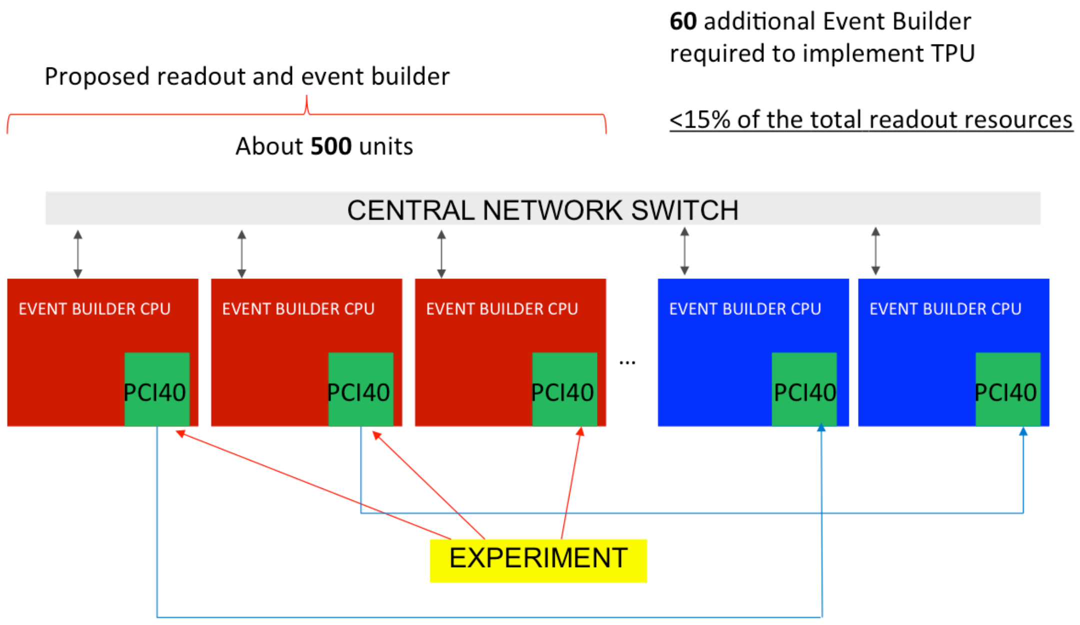

Figure 7: Organization of the TPU and connections to VELO and UT readout in the PCIe40 scheme.

TPU AMC40. With this configuration, 32 AMC40 and 968 5 Gbit/s fibers are needed to implement the TPU for one VELO-UT telescope, leading to a total of 64 AMC40 and 1936 fibers for the complete system of two telescopes.

Within each FPGA, a switch consisting of a full-mesh 32×32 way dispatcher distributes the data into 32 fanout blocks, each serving 8 outs with 6 engines each. For ease of integration, we studied the implementation in the same Stratix V device chosen as default design for the AMC40/TELL40 boards, model 5SGXEA7N2F45C2ES. We designed the switch system in full detail in VHDL language, and performed placement and simulation using Altera's proprietary software tools. Results show that we can smoothly run the switch at frequencies of 350 MHz.

The pre-switch occupies 3.3% of the available logic in the Stratix V and completes its processing in 15 clock cycles. The switch needs 7.5% of logic and employs another 15 cycles.

3.2 The processing engine

Detailed simulations were performed to optimize the physical implementation of a retina algorithm in a realistic FPGA architecture. Each cell is defined as a logic module, the engine. Each subdetector hit is defined as a 41 bits-wide word encoding the corresponding geometric x and y coordinates, a layer identifier, and the associated timestamp. The target is to fill 80% or less of the FPGA logic with as many engines as possible, including also the logic needed to find the local maximum. The logic for cluster center-of-excitation calculation is accounted for separately.

The engine is implemented as a clocked pipeline. The intersections $x_0(k)$ and $y_0(k)$ for each layer k are stored in a ROM. The layer identifier associated with each incoming hit selects the appropriate set of $x_0(k)$ and $y_0(k)$ coordinates that are subtracted from the hit's x and y coordinates. The outcomes of the subtractions are squared, summed, and the result R is rounded by keeping the eight least significant bits. A sigma function, common to all engines, is mapped into a 8×256 bit lookup table. The rounded result R is used as address to the lookup table. The outputs of the lookup table are accumulated for each hit of the event. The same hit is cycled seven times in the engine logic, once for the central cell calculation, and six for the lateral cell calculations. Hence, seven accumulators are defined for each cell and one hits enters the engine every seven clock cycles. See Sec. 4 and Fig. 10 for a description of the cell in the track's parameter space.

Several variants of the architecture were implemented. Simple configurations include schemes in which hits arrive time-ordered to the engine, all with the same timestamp up to the EndEvent bit. More complex scenarios allow time-shuffling among hits, up to a maximum of 16 events simultaneously processed, which implies 16 groups of accumulators. Once readout of an event is completed, a word with the EndEvent bit arrives, prompting each engine to send the contents of its central cell to the neighboring engines. Immediately after, each engine compares the excitation in its central accumulator with the excitations received from the neighbors and raises a LookAtMe flag if it identifies its excitation as a local maximum. The chosen FPGA chip is the same used for the switch. The simplest design, time-ordered with seven accumulators, compiled in the Quartus II environment, shows that about 900 engines can be fit in one Stratix-V, leaving approximately 25% of logic available for other uses. We conservatively assume 750 engines per Stratix-V, to account for possible needs for extra logic in the connection of components and for time-alignment. This allows implementing a TPU with $64 \times 750 = 48\ 000$ engines for processing the two-telescope VELO-UT configuration (see Sect. 4 for a derivation of this figure). The maximum clock frequency (worst case) is of the order of 400 MHz. Hence, every engine in the system is able to accept one hit every 20 ns approximately (1/ (400 /8)).

3.3 Clustering

Including the logic that identifies the center-of-excitations in the cell matrix is a minor expansion of this scheme. Such logic looks at the LookAtMe flag and, if not busy, requires from the engine the content of all accumulators *and* the content of central accumulators of neighboring engines. These data are used to calculate the track parameters as follows. The center-of-excitation calculation is factorized into two separate processes. The calculation restricted to the track parameters in the plane transverse to the beam line (u, v, see Sec. 4.1) implies finding the center of mass of a 3×3 square; the calculation relative to the remaining track parameters (d, p, z) (see Sec. 4.1) requires computing the center of mass of a $3 \times 3 \times 3$ cube. Only a subset of coordinates in each dimension turns out to be relevant

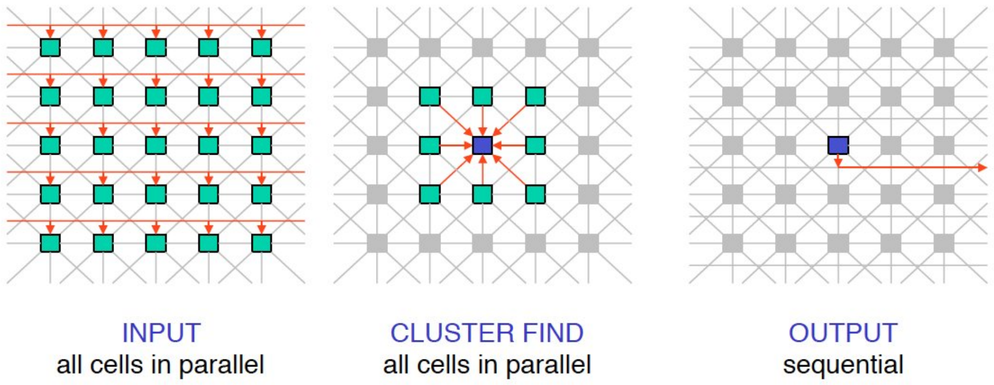

Figure 8: Illustration of the clustering data flow.

for the final result, hence the problem reduces to processing a smaller number of values. The operation for each coordinate is $u = u_0/d_k + (\sum_{i,j} u_i l_{i,j})/\sum l$, where u_0/d_k is a global translation that depends on the absolute position of the engine and is not calculated in real time, but stored in a lookup table. Two distinct weights are simultaneously produced for (u, v) and (d, p, z), respectively. In a possible architecture, the computation of the center of excitation takes 11 clock cycles along with another 10 cycles for fanout with a logic that occupies a fraction not larger than 15% of the Stratix V. To optimize resource sharing, a single center-of-excitation unit can serve multiple engines. Simulations show that a scheme with a unit serving each group of 12 engines is adequately sized for the hit occupancies expected. The search for the local maximum and the center of excitation use local copies of the accumulators so that the incoming hit flux is never stopped unless large fluctuations of the EndEvent word arrival times occur. In these cases the incoming hits are kept on hold and stored in the switch trees.

3.4 Logic simulation and timing

The design of each part has been fully validated through a Modelsim simulation. The results show that the TPU is capable of keeping up with an input frequency of a full 40 MHz of events, with the occupancy predicted by the full LHCb simulation, in the nominal luminosity conditions of the 2020 upgrade, $L = 2 \times 10^{33}$ cm^{-2}s^{-1}. The total latency budget is shown in Table 1. With clock frequencies of 350 MHz, the latency for reconstructing online tracks is less than 0.5 μs, thus negligible compared to other latencies in the DAQ data flow. This makes the response of the device effectively *immediate*, thus making tracks available right after the tracking detectors have been read out.

Task	Latency (cycles)
Switch in readout board	15
Switch in TPU – dispatcher	15
Switch in TPU – fanout	6
Engine processing	70
Clustering	11
Output data	10
Total	< 150

Table 1: TPU event-processing time break-down.

4 Tracking performance

A high-level software emulation of the retina algorithm is developed to assess the tracking performances and benchmark these against the VELO-UT algorithm from the LHCb offline tracking sequence.

4.1 Definitions

We consider the LHCb detector as an array of tracking planes, such as a combination of VELO and UT subdetector layers. We choose the following five parameters to uniquely identify tracks:

- $\boldsymbol{(u, v)}$ – spatial coordinates of the intersection of the track on a virtual plane, which can be assumed as an additional layer perpendicular to the beam (z) axis positioned between detectors layers, as shown in red in Fig. 9.
- $\boldsymbol{d}$ – signed transverse impact parameter defined as the distance of closest approach to the z-axis;
- $\boldsymbol{z}$ – z-coordinate of the point of closest approach to the z-axis;
- $\boldsymbol{k}$ – signed track curvature, defined as $q/\sqrt{p_x^2 + p_z^2}$, where q is the particle charge, and p_x and p_z are the momentum components perpendicular to leading magnetic field direction ($\vec{\boldsymbol{B}} = \boldsymbol{B}\hat{\boldsymbol{y}}$), at production time.

We divide the five-dimensional space parameter into a grid of receptive fields, the (*cells*). Each parameter-space cell $(u, v, d, z, k)_i$ is uniquely associated with a set of coordinates (x_{ij}, y_{ij}), where the index j runs over the n detector layers. These coordinates identify the intersections that a track with parameters corresponding to the center of the cell has with the measurement planes.

A full-fledged approach would require to subdivide the parameter space with sufficient granularity, depending on the capability of the retina of distinguishing hundreds of tracks in the same event and on the target resolution on final track parameters. This would yield a very large number of cells, which are difficult to fit in a cost-effective physical device.

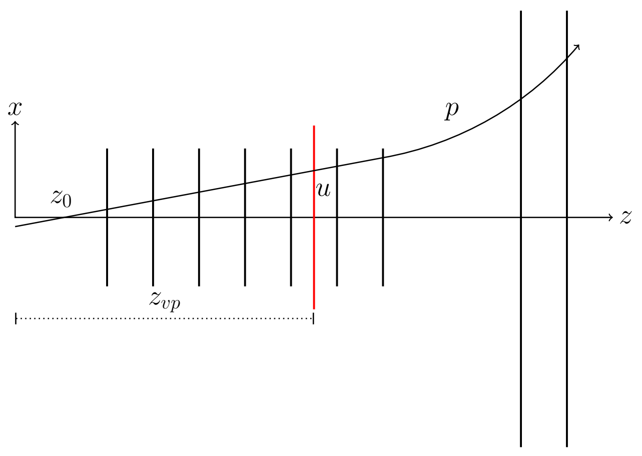

Figure 9: Schematic illustration of track parameters.

We opt for an alternative, prioritized approach by factorizing the task into two separate operations: (i) in the *pattern recognition* step real tracks are distinguished from accidental combination of random hits using a subspace of track parameters; (ii) in the *parameter extraction* step, track parameters are determined extending to the full dimensionality. The geometry of typical tracks in LHCb allows factorization of the parameter space into the product of two subspaces with rather distinct dimensional scales, since variations in the (d, z, k) parameters can be approximated to small perturbations of the main (u, v) parameters. This allows performing the pattern recognition using only a two-dimensional retina in the (u, v) space, where other parameters fixed to zero $(d = z = k = 0)$. At this level, a track is identified by a cluster over threshold in this two-dimensional space. In the second step, a preliminary estimate of track parameters is performed within the Stratix-V chip using the strategy discussed in Sec. 3.3, by balancing the excitation found in the lateral cells for each compact dimension. For each main cell $(u, v, 0, 0, 0)$ we fill six lateral cells $(u, v, \pm\delta d, \pm\delta z, \pm\delta k)$, where $\delta d = 1$ mm, $\delta z = 150$ mm and $\delta k =1$ GeV^{-1} as shown in Fig. 10. The u, v calculation implies finding the center of mass of a 3×3 square, whereas the extraction of (d, z, k) requires computing the center of mass of a $3 \times 3 \times 3$ cube whose only a subset of coordinates in each dimension are nonzero, thus reducing the problem to the processing of seven values. A further refinement of track parameters estimation is achieved with a linearized track fitting algorithm [9, 10] by using detector hits associated with the maximum excitation. This straightforward computation of scalar products can be executed either within the Stratix-V chip or the available CPU in the event builder box with negligible usage of logic. By identifying each track using m parameters, we write

$$\mathbf{x} = \mathbf{x}(p_1, ..., p_m) = \mathbf{x}(u, v, d, z, k),$$

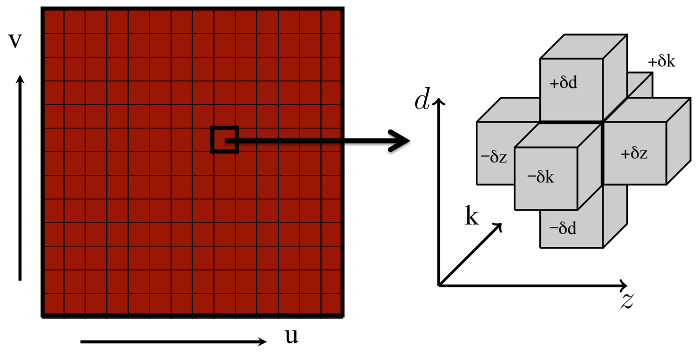

Figure 10: Illustration of the main cells in the (u, v) space with $d = z = k = 0$ along with the lateral cells associated with compact dimensions.

where $\mathbf{x}$ is a vector of track hits (coordinates). This relationship is inverted, at least locally, to obtain the parameters as functions of coordinates

$$p_i = p_i(\mathbf{x}).$$

These functions are approximated with m linear functions

$$p_i \approx \mathbf{w}_i \cdot (\mathbf{x} - \mathbf{x}_0) + p_i(\mathbf{x_0}) = \mathbf{w}_i \cdot \mathbf{x} + q_i,$$

where the constants $(\mathbf{w}_i, q_i)$ are obtained from simulated tracks with known parameters and $\mathbf{x_0}$ is the vector of hits corresponding to the main cell found $(u_0, v_0, 0, 0, 0)$. In principle, a set of constants is needed for each (u, v) cell; however, the linearization works accurately in a much larger area: a limited number a subregions (≈ 100) is sufficient to extract parameters with offline-like quality. For the purpose of the high-level emulation, we discretize the main (u, v) space into about 45 000 cells (22 500 cells for each of the two telescopes), with a granularity $\mathcal{O}(100)$ larger than the expected maximum number of tracks in the event. We simulate the retina algorithm with a `C++` program that accepts in input hits from the official LHCb upgrade simulation.

To benchmark the performance of the retina against the standard VELO-UT offline reconstruction, we compare tracks reconstructed by the retina with generated tracks and offline-reconstructed tracks by matching them based on proximity in the space of the u, v parameters: tracks that have compatible values of these parameters are considered *matched* and treated as the same element in the representation of track sets shown Fig. 11. Parameters u and v of two tracks are compatible if their separation does not exceed five cells. Given the significant conceptual differences between the retina algorithm and

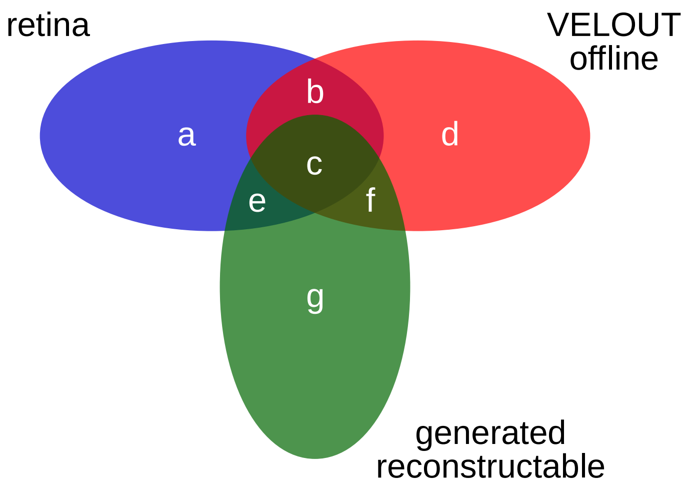

Figure 11: Sketch of the retina, offline, and generated track sets used in the performances studies.

the offline reconstruction algorithms, a matching based on parameter-space distance is conceptually the simplest, that minimizes the need for extra arbitrary parameters. Similar studies made using standard LHCb matching criteria based on fractions of shared hits between tracks reconstructed with different algorithms yield equivalent results. In fact, we observe that prior to any optimization, once a track is reconstructed by the TPU, the fraction of VELO and UT hits associated to it that were genuinely produced by the corresponding charged particle is in excess of 90%.

Based on Fig. 11 we define the following classes of tracks:

- Class a: tracks reconstructed by the TPU only, but not generated;
- Class b: tracks reconstructed by both TPU and offline, but not generated;
- Class c: generated tracks reconstructed by both TPU and offline;
- Class d: tracks reconstructed by the offline only, but not generated;
- Class e: generated tracks reconstructed by the TPU only;
- Class f: generated tracks reconstructed by the offline only;
- Class g: generated tracks reconstructed neither by the TPU nor by the offline.

Pathological classes, which cannot be represented using a Venn diagram, are also inspected. These include tracks that are reconstructed by the retina and the offline reconstruction, that match with the same generated track, but that do not match to each other, and other similar cases. The contributions from all of these are negligible in the efficiency computation. Benchmark quantities that encode the performances of the retina and VELO-UT offline algorithms are defined as follows

- TPU efficiency, ϵ_{ret} = (c+e)/(c+e+f+g);
- TPU ghost rate, g_{ret} = (a+b)/(a+b+c+e);
- TPU overefficiency with respect to the offline, $\eta_{\text{ret}|\text{off}}$ = e/(c+e+f+g);
- Offline efficiency, ϵ_{off} = (c+f)/(c+e+f+g);
- Offline ghost rate, g_{off} = (b+d)/(b+c+d+f);
- Offline overefficiency with respect to the TPU, $\eta_{\text{off}|\text{ret}}$ = f/(c+e+f+g).

The denominator for the ghost rate is based on tracks selected through significantly loosened criteria. Specifically, the ghost rate is determined as the ratio between the number of all tracks reconstructed by the considered algorithm (retina or VELO-UT) and the number of generated charged particles that leave at least one hit in one telescope's layer. A track is *longable* if it is associated with three or more hits on different VELO stations and at least one axial and one stereo hit on each of the scintillating-fiber detector stations. In the standard offline reconstruction a track is defined as *reconstructable* if it is not associated with an electron, has three or more hits on different VELO stations, and at least one or more hits on each UT station.

4.2 Tracking layer configuration

To determine the optimal choice of VELO layers to be included in the retina, the polar coverage ranges of each frontside layer are calculated and reported in Fig. 12. Approximately 85% of longable tracks are covered by a polar acceptance of 150 mrad. Figure 13 shows a choice of VELO layers configuration that fully covers the UT acceptance. In this configuration, the tracking layers are grouped into two partially overlapping telescopes: a near telescope (25–330 mrad coverage) and a far telescope (8–90 mrad).

As a first study, we assess the performance of a minimal retina system involving one telescope consisting of eight VELO layers and two axial UT layers. All studies reported in the following are based on the far telescope only. Preliminary studies show that extension to the full, two-telescope system is straightforward. The only practical effect of using a single TPU telescope is the reduction of the geometrical acceptance by approximately 50%, which is expected to be fully recovered once the second telescope is included.

The detector's geometrical layout and performance relies on the latest available version of the LHCb upgrade simulation and reconstruction software. We use a sample of minimum-bias events generated with PYTHIA8, with beam energy E_{beam} = 7 TeV; $\nu = 7.6$ and $\nu = 11.4$ interactions per crossing, corresponding to instantaneous luminosities of $L = 2 \times 10^{33}\text{cm}^{-2}\text{s}^{-1}$ and $L = 3 \times 10^{33}\text{cm}^{-2}\text{s}^{-1}$, respectively; 25 ns of interbunch crossing time with spillover; and a single (down) magnet polarity. The current TPU software

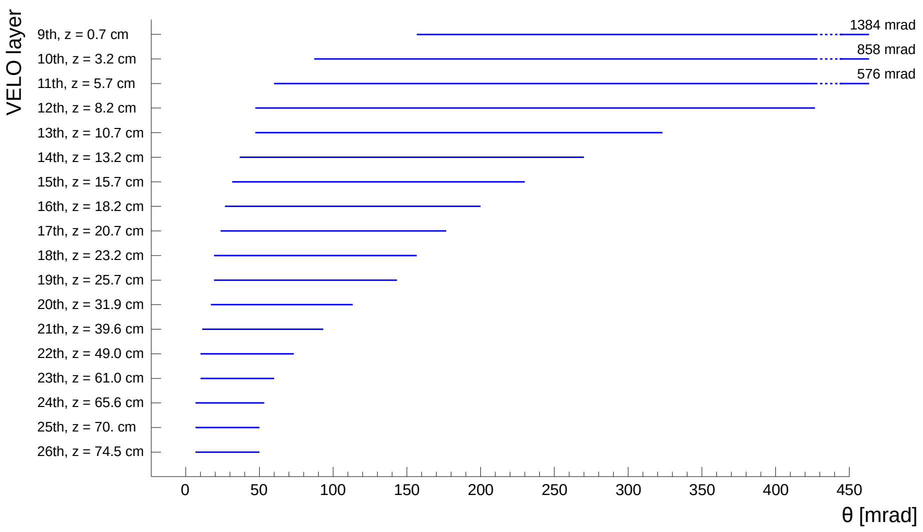

Figure 12: Polar acceptance ranges of frontside VELO layers.

emulation takes about 700 ms per event on a laptop computer[1], prior to any attempt at timing optimization.

4.3 Performance

For a consistent comparison with the offline algorithm that reconstructs tracks in the VELO and UT detectors, the denominator of the efficiency is chosen to be the same for both samples of tracks and as similar as possible to that used in studies of the offline reconstruction [20]. Since the retina simulation uses only the two UT axial layers, an alternative definition is used based on requiring one hit on both the UT axial layers. This variant is observed not to yield significant differences, thus preserving the consistency of the offline-retina comparison. Therefore, in what follows, a track is **reconstructable** if

- is not associated with an electron;
- is associated with three or more hits on different VELO stations among those considered in the retina;
- is associated with one or more hits on both UT axial layers. (More than one hits per layer may occur in regions of overlapping UT pads)

Regardless of the reconstructability definition, the TPU processes all hits belonging to an event, including those not associated with reconstructable tracks. The reconstructability criterion is only introduced to have a common denominator for meaningful comparison with offline studies.

[1]Intel Core i5-420M with clock frequency of 2133 MHz.

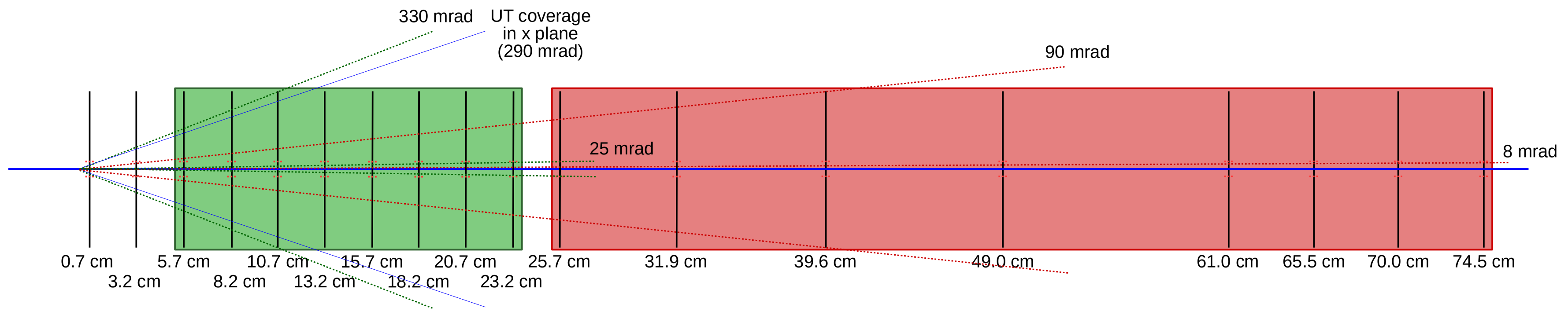

Figure 13: Acceptance regions of the two VELO layer telescopes requiring at least three VELO hits to reconstruct a track. Layer disposition on the z coordinate is to scale.

Assuming the placement of the virtual (u,v) plane at $z = 593$ mm, the (u,v) surface illuminated by reconstructable tracks is a square of approximately 0.5 units side. Since the probability for tracks to be reconstructable degrades for tracks pointing away from the center of the virtual plane, because they intercept a reduced number of physical tracking planes, fiducial cuts of $\max(|u|,|v|) < 0.35$ (equivalent to $\approx \theta < 50$ mrad) and $|z| < 150$ mm are applied. The effect of this acceptance-related inefficiency is expected to be negligible when the full two-telescope configuration is used. Generated tracks are required not to be electrons. Figure 14 compares the TPU and offline performances on *longable* tracks within fiducial criteria, and according to the previous reconstructability definition. Momentum thresholds of $p > 3$ GeV/c and $p_T > 0.5$ GeV/c are required to select a sample of tracks of interest for trigger purposes, but the TPU performance remains unchanged even by lowering the p_T threshold to 200 MeV/c.

Table 2 shows the algorithm efficiency performance. Table 3 shows ghost rates.

$\nu = 7.6$	ϵ_{ret}	ϵ_{off}	$\eta_{\text{ret}\vert\text{off}}$	$\eta_{\text{off}\vert\text{ret}}$
Longable	0.95	0.95	0.04	0.03
$B_s^0 \to \phi\phi$ signal tracks, longable	0.97	0.97	0.02	0.02
$D^{*+} \to D^0\pi^+$ signal tracks, longable	0.97	0.97	0.03	0.02
$B^0 \to K^*\mu\mu$ signal tracks, longable	0.98	0.98	0.01	0.01

$\nu = 11.4$	ϵ_{ret}	ϵ_{off}	$\eta_{\text{ret}\vert\text{off}}$	$\eta_{\text{off}\vert\text{ret}}$
Longable	0.95	0.94	0.05	0.04
$B_s^0 \to \phi\phi$ signal tracks, longable	0.97	0.97	0.02	0.02

Table 2: Performances of TPU and VELO-UT offline algorithms averaged on 10^4 minimum-bias events and benchmark signal samples generated at $\nu = 7.6$ at $L = 2 \times 10^{33}$ cm^{-2}s^{-1} (top) and $\nu = 11.4$ at $L = 3 \times 10^{33}$ cm^{-2}s^{-1} (bottom). Momentum criteria of $p > 3.0$ GeV/c and $p_T > 0.5$ GeV/c are applied on generated particles only. Generated particles are required not to be electrons and be associated with at least three hits on selected VELO layers and one hits on both the axial UT layers. Only tracks in the fiducial region of $\max(|u|,|v|) < 0.35$ are considered; a fiducial requirement of $|z| < 150$ mm is applied on generated tracks.

	g_{ret}	g_{off}
$\nu = 7.6$	0.08	0.06
$\nu = 11.4$	0.12	0.08

Table 3: Ghost rates of TPU and VELO-UT offline algorithms, averaged on 10^4 minimum-bias events generated with $\nu = 7.6$ at $L = 2 \times 10^{33}$ cm^{-2}s^{-1} and $\nu = 11.4$ at $L = 3 \times 10^{33}$ cm^{-2}s^{-1}.

As a proof of principle, we also extract parameters of minimum-bias tracks restricted to a (u,v) region defined as $30 < \theta < 35$ mrad, $7\pi/30 < \phi < 8\pi/30$ rad by using linearized

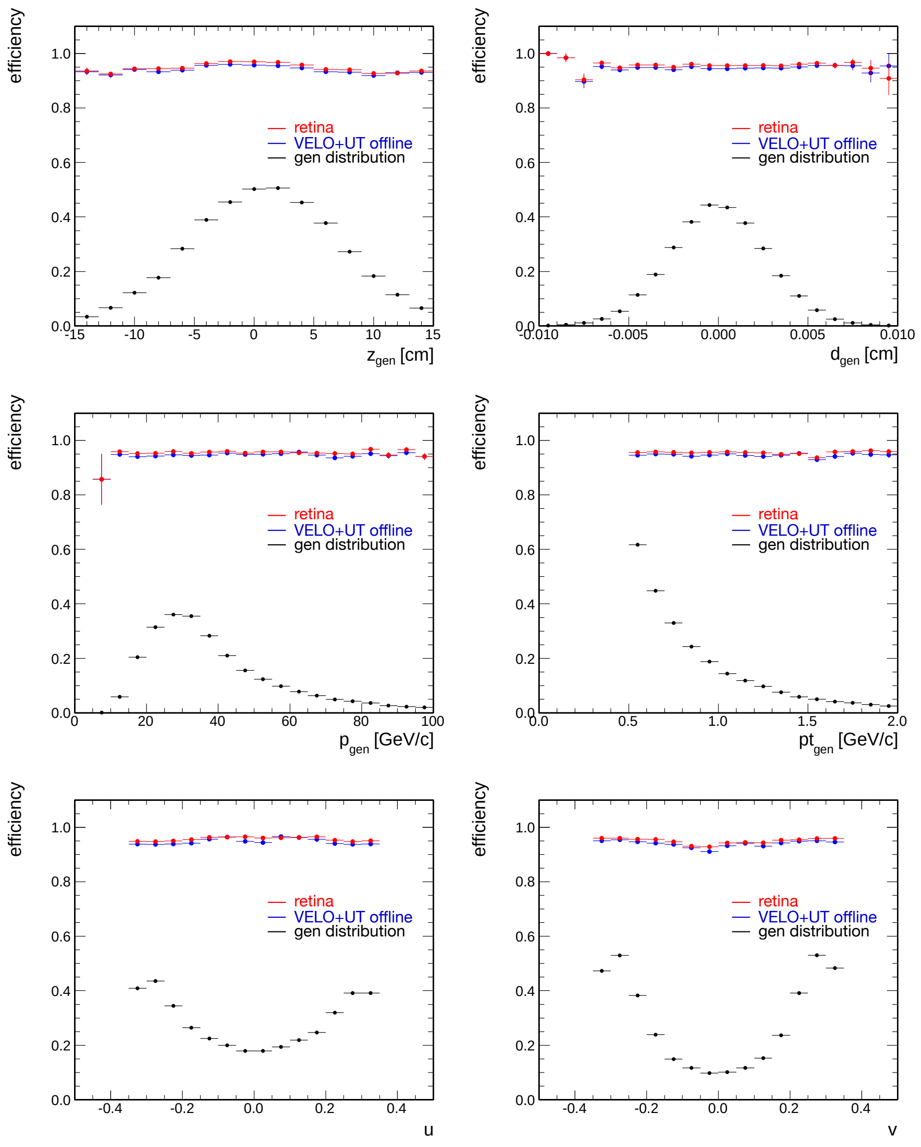

Figure 14: TPU and VELO-UT offline efficiencies as functions of generated parameters of longable tracks, averaged on 10^4 minimum-bias events generated with $\nu = 7.6$ at $L = 2 \times 10^{33}$ cm^{-2}s^{-1}). Momentum criteria of $p > 3.0$ GeV/c and $p_T > 0.5$ GeV/c are applied on generated particles only. Generated particles are required not to be electrons and to have at least three hits on selected VELO layers and one hits on both axial UT layers. Only tracks in the fiducial region of $\max(|u|, |v|) < 0.35$ are considered; a fiducial requirement of $|z| < 150$ mm is applied on generated tracks.

track fitting. Figure 15 shows the resulting TPU curvature resolution compared to the resolution from the VELO-UT offline algorithm for particles with momentum spectrum typical of minimum-bias samples. The modest 25% degradation in resolution occurs

	$\nu = 7.6$	$\nu = 11.4$
Number of physical hits	880	1220
Number of hits delivered to the engines	32805	48976
Number of clusters	121	223
Number of hits per engine	1.3	1.95

Table 4: TPU occupancy averaged over 10^4 minimum-bias events generated with $L = 2 \times 10^{33}\text{cm}^{-2}\text{s}^{-1}$ at $\nu = 7.6$ and $L = 3 \times 10^{33}\text{cm}^{-2}\text{s}^{-1}$ at $\nu = 11.4$.

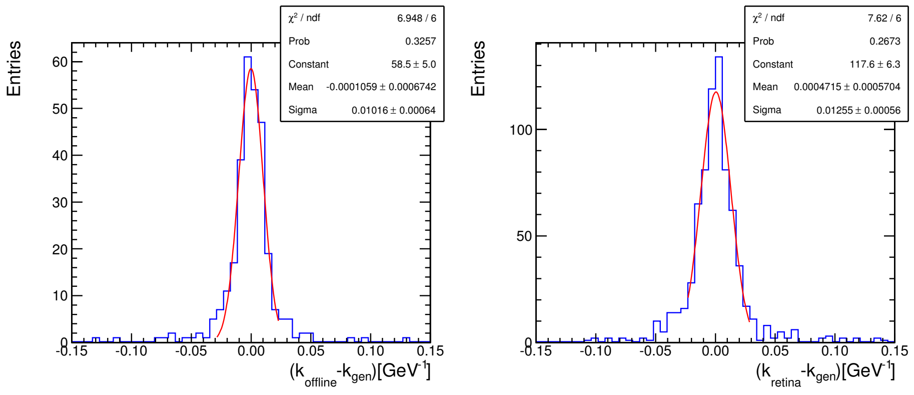

Figure 15: Comparison between curvature resolution achieved with the TPU (right panel) and with the VELO-UT offline reconstruction algorithm (left panel).

because the TPU uses only two UT layers compared to the four layers used offline. Part of this is likely to be recoverable with optimized track fitting configurations. In addition, a layout in which the computation-light linearized track fitting is promoted into the FPGA would leave the event builder PC free to add the information relative to the other two layers and recover full offline resolution. Similar results are achieved for all other track parameters.

Tables 2 to 4 show that the tracking performance of the TPU is very robust against increases in event complexity associated with higher instantaneous luminosity. The TPU track finding preserves substantially constant efficiency and has only a minor increase in ghost rate.

5 Impact of the TPU on the LHCb upgrade

The availability of tracks provided by the TPU may have a significant impact on the operations and physics reach of the LHCb upgrade program. TPU tracks can be used at different stages in the DAQ.

5.1 Timing impact

TPU tracks can be used in the HLT just as the VELO and VELO-UT tracks would normally be used. This allows saving the time spent for executing the corresponding track reconstruction using HLT code. Reconstruction of VELO tracks in minimum bias events (with no requirements on global event charged particle multiplicities), using the most recent version of the HLT upgrade simulation, takes 2.3 ms/event of CPU-2012 time, and extension to VELO-UT tracks takes another 1.4 ms/event, for a total of 3.7 ms per event. However, a strictly consistent comparison with the TPU is not straightforward, since the HLT code performs additional work, such as reconstructing the subset of tracks that are not in the UT acceptance, which is not done in the TPU configuration considered. More refined evaluations, in which similar choices of tracking layers are used in the HLT code and the TPU, yield an estimate of the CPU-time equivalent for the TPU computation of between 2.3 and 2.9 ms per event, depending on the details of the specific assumptions made.

The HLT software group provided an example of an actual tracking sequence in which the total time saving from the TPU is 2.1 ms/event [21]. In this sequence, tracking is performed standalone in the ten layers not included in the examined TPU layout, spending additional 1.4 ms/event. This code and tracking sequence could be optimized, possibly exploiting information from available TPU tracks to speed up pattern recognition; hence this figure is considered a lower bound to possible savings. With a dedicated optimization, and leaving the 0.3 ms/event for decoding and formatting, or part of it, to the event building stage, the time-saving should approach the above-estimated CPU-time equivalent. Given that a total of 6.6 ms/event is expected to be spent in HLT tracking, we conclude that the current TPU layout saves 1/3 of HLT tracking time.

5.2 Structural DAQ impact

Owing to the TPU's low latency, TPU tracks are immediately available to the event builder, even before the events are transferred to the farm. They will actually be available even prior to event building. This has a further impact, additional to the pure timing reduction. The event builder has some embedded CPU power, estimated of order 1-2 ms per event; the possibility of running the online code on these nodes has been demonstrated [14]. Providing the EB CPUs with readily usable tracks allows some significant HLT preprocessing to be performed locally on all events, prior to moving them to the farm. In a staged-processing scenario, the TPU offers the possibility of implementing a HLT1-like selection inside the EB itself, thus making it possible to control the event rate into the farm, without having to rely on low-level selection criteria.

One simple and important application of this concept is *lepton confirmation.* Studies have shown that the rate of events with muon primitives will be challengingly high in the upgrade conditions. Matching the muon stub with a TPU track in the EB, just as it has been done by HLT inside the farm in Run 1, will provide a natural knob to control the muon-event rate prior to the farm, thus allowing higher trigger efficiency for both muon and hadronic samples. Such kind of additional flexibility can be exploited by the HLT in many other ways.

5.3 Additional impact

The capability of a pre-farm rate-reduction could prove vital in the early Run 3 operation, when the computing potential of the farm might not be fully deployed and online event selection criteria might yet need to be optimized to cope with the upgrade conditions. It could also provide a gateway toward the possibility of efficient triggering at higher than currently foreseen instantaneous luminosities, should this become an option scientifically attractive.

The TPU approach appears as an alternative to the adoption of criteria based on global event charged-track multiplicity or calorimetric criteria for controlling the rate in case of need, as happened in LHCb Run I. With performances equivalent to offline tracking, and being fully simulable in software, the TPU option is free from unwanted potential systematic biases associated with discarding sizable fractions of signal events based on global charged-particle activity. It also avoids further complications in the data analysis associated with the need to vary those criteria over time, should the need to do so arise. This will provide an online selection based on tracks, hence much closer to the offline selection.

6 Cost

6.1 Hardware cost

The TPU implementation is based on a number of FPGA cards, appropriately connected via optical links. The system aims at reconstructing VELO-UT tracks with good coverage for long tracks, which are of most interest for the trigger. Such functionality is most conveniently implemented by factorizing the TPU into two subsystems covering two complementary solid angle regions, requiring a total of 32+32=64 FPGA devices.

The documentation provided with the readout review held on Feb 25, 2014 shows that the costs of a PCIe-based implementation and an AMC-based implementation are similar. For ease and reliability, we report here an estimate based on the PCIe system, for which full details are available. Our 64 units represent 12% of the boxes needed for the entire readout system. The difference between TPU boxes and regular EB boxes is that the TPU cards only produce a modest amount of information in output ($\approx 10\%$ of the input data) and therefore they contribute a negligible increase of event size ($\approx 1\%$). For this reason, the TPU boxes do not need to build events, and only send a small data flow towards other boxes; they can work with a single CPU and are essentially little more than physical support for the card themselves. The system could be certainly made cheaper by using multiple-slot boxes, probably significantly so; however, for the sake of simplicity, in this estimate we have conservatively assumed the price of a standard EB box as a guideline.

The total price estimate, shown in Table 5, accounts for extra fibers and patch panels, and is based on current market prices, and should be considered conservative.

	Entities	Unit cost	Total cost
PCIe40(30/0)	64	6.6	425
Bidirectional switch ports	64	0.8	48
PC server	64	3.5	224
Infrastructure and spares			243
Total			940

Table 5: TPU system cost estimate in kCHF.

6.1.1 Cost-effectiveness of the technology

We evaluate the use of a different technology to perform part of the event reconstruction in terms of comparative cost advantages. This comparison is based purely on computing power cost, and ignores more structural benefits to the DAQ. To evaluate the general cost-effectiveness of an FPGA-based processor, we compare it with the cost of a standard tracking implementation using commercial CPUs. To avoid uncertainties associated with extrapolating performance and prices to a future date, we perform the comparison at the current time.

The computing capability of our proposed TPU system is to perform the VELO-UT offline-like tracking reconstruction, with 16+2 layers, at a full 40 MHz event rate. The equivalent piece of offline code, executed on the same event sample, consumes an estimated 2.1 ms to 2.9 ms per event of CPU time, when run standalone on a single 2012 CPU core. Given that in 2012 one millisecond at 40 MHz costs about 4.8 MCHF, it is clear that at present the TPU is strongly cost effective. Taking 2.5 ms/event as a conservative estimate of timing gain, the bare computing power of the TPU equals 50 000 physical cores (corresponding to 100 000 logical cores via hyperthreading), even neglecting the slowdowns deriving from running concurrently on multiple cores. We estimated the current price of each CPU physical core at about 250 CHF. This leads to today's price for an equivalent CPU system of 12.5 MCHF. Comparing this price tag with the current market price for the FPGA system clearly shows that, as of today, the technology is worthwhile even just from the point of view of the computing power; that is, neglecting the advantages of ultralow latencies. Similar considerations hold true for power consumption, which is an important contributor to total cost, even if not directly charged to LHCb. A reasonable estimate of power consumption for this kind of FPGAs is 50 W per chip, yielding a total of 3 kW for the entire TPU system. For the corresponding system implemented in today's CPUs, one would need perhaps 4 200 nodes (12 physical cores) each consuming 100 W or more, leading to a total of ≈ 0.4 MW.

These findings show that FPGAs are a cost-effective solution for this kind of computations. The only significant obstacle to their general use is their lack of adaptivity to a generic problem, and the need for careful and problem-specific programming. The latter is indeed the issue the we tried to address with our algorithm-development effort described in this document. It is interesting to note that properly programmed FPGAs are a common solution in the industry for performing the Hough transform, which is computationally quite similar to the retina in its demands to track reconstruction problems, resulting in huge gains in comparison with CPU-based systems [22].

6.1.2 Projected costs

Estimating cost-effectiveness in future is harder, given the need to rely on several assumptions. An estimate by readout experts and some of the proponents of the HLT full software solution [23] evaluates in 8 ms/event the time-saving needed in 2020 for the TPU (of assumed price 1 MCHF) to be cost-effective relative to the HLT software approach. This estimate relies on some important assumptions: (i) FPGA (and TPU) price remaining constant at the current estimates up to buying time, while CPU prices drop by a factor 16$\times$; (ii) the timing performance of one farm node loaded with 400 concurrent jobs is the same as if only one job was running; (iii) a factor of 2$\times$ advantage is ascribed to CPUs because they can be used for deferred data processing and Monte Carlo generation during quiet time; (vi) power consumption, cooling, and infrastructure for CPUs are not accounted for in the comparison.

The future evolution of the market will certainly see the prices of FPGAs systems to decrease. All data show that in recent years the price evolution of FPGA devices has been at least as fast as that for commercial CPUs, and similar trends hold for power consumption. We already know that prices quoted in the previous section, based on a Stratix-V implementation, will reduce by moving to the Arria 10 device, which is already on the market with a price per logic gate reduced by 50% with respect to the Stratix-V. Further reduction is likely with the commercial introduction of 14-nm 3D gate technology from Intel (Stratix-10 family), expected sometime in 2014, which will unavoidably lower the price of the Arria series, based on older 20-nm planar technology. The cost of the PC infrastructure will be more stable, but given the lower bandwidth demands of the TPU boxes, we can easily conceive to fit more PCIe cards in the same CPU box. In conclusion, given that the cost of the TPU device is currently very competitive with CPU-based solutions (an order of magnitude lower), and although it may not decrease as rapidly as raw CPU prices in the future, we expect that it will still be very competitive in the upgrade era.

7 Summary

- We designed a system capable of track reconstruction at 40 MHz with offline-like performance and submicrosecond latencies.
- The track processor acts as a virtual detector that outputs tracks into the event builder, allowing local HLT preprocessing.
- The track processor cost is clearly competitive with today's CPU solutions in terms of processing time.
- Cost projections and savings compared to the CPU options in the future can vary significantly, depending on many assumptions.
- However, we showed that the baseline VELO-UT configuration of the TPU saves 1/3 of the HLT tracking time. This may result in significant savings if the LHC Run 3 operating conditions will be more challenging than those predicted by current simulations, and may allow efficient operation of the LHCb trigger at higher luminosities than currently planned.

Adoption of the TPU is a cost-effective way of increasing the available computing power and helps LHCb attaining its objective of an online selection as close as possible to the offline selection more completely and rapidly than otherwise possible. Moreover, as it often occurs when significantly extended experimental capabilities are added to an existing system, it is likely that the specific details of the TPU's optimal usage will become manifest only when the system will actually be operated within the real environmental conditions of hadron collisions in LHC Run 3.

Acknowledgements

We are very grateful to Luciano Ristori, whose ideas, suggestions, and advice have been of invaluable help throughout the development of the project. We thank Richard Jacobsson and Federico Alessio for providing a lot of detailed informations on the LHCb readout architecture and patiently answering all of our questions. We also thank the LHCb HLT and online groups for discussions and exchanges of information.

References

[1] R. Aaij *et al.*, *The LHCb Trigger and its Performance in 2011*, JINST **8** (2013) P04022, `arXiv:1211.3055`.

[2] J. Albrecht, V. Gligorov, G. Raven, and S. Tolk, *Performance of the LHCb High Level Trigger in 2012*, `arXiv:1310.8544`.

[3] *Letter of Intent for the LHCb Upgrade*, Tech. Rep. CERN-LHCC-2011-001. LHCC-I-018, CERN, Geneva, Mar, 2011.

[4] C. Daum *et al.*, *Online Event Selection With the Famp Microprocessor System*, Nucl. Instrum. Meth. **217** (1983) 361.

[5] G. Foster, J. Freeman, C. Newman-Holmes, and J. Patrick, *A Fast Hardware Track Finder for the CDF Central Tracking Chamber*, Nucl. Instrum. Meth. **A269** (1988) 93.

[6] E. J. Thomson *et al.*, *Online track processor for the CDF upgrade*, IEEE Trans. Nucl. Sci. **49** (2002) 1063.

[7] M. Dell'Orso and L. Ristori, *VLSI structures for track finding*, Nucl. Instrum. Meth. **A278** (1989) 436.

[8] F. Morsani *et al.*, *The AMchip: A VLSI associative memory for track finding*, Nucl. Instrum. Meth. **A315** (1992) 446.

[9] L. Ristori and G. Punzi, *Triggering on heavy flavors at hadron colliders*, Ann. Rev. Nucl. Part. Sci. **60** (2010) 595.

[10] CDF Collaboration, B. Ashmanskas *et al.*, *The CDF silicon vertex trigger*, Nucl. Instrum. Meth. **A518** (2004) 532, `arXiv:physics/0306169`.

[11] A. Andreani *et al.*, *The fasttracker real time processor and its impact on muon isolation, tau and b-jet online selections at atlas*, .

[12] M. Shochet *et al.*, *Fast TracKer (FTK) Technical Design Report*, Tech. Rep. CERN-LHCC-2013-007. ATLAS-TDR-021, CERN, Geneva, Jun, 2013. ATLAS Fast Tracker Technical Design Report.

[13] L. Ristori, *An artificial retina for fast track finding*, Nucl. Instrum. Meth. **A453** (2000) 425.

[14] N. Neufeld *et al.*, *The LHCb Readout for the Run 3*, .

[15] D. Hubel and T. Wiesel, *Receptive fields of single neurones in the cat's striate cortex*, J. Physiol. **148** (1959) 574.

[16] D. Hubel and T. Wiesel, *Receptive fields, binocular interaction and functional architecture in the cat's visual cortex*, J. Physiol. **160** (1962) 106.

[17] L. Ristori, *An artificial retina for fast track finding*, Tech. Rep. Slides of the 7th International Conference on Instrumentation for colliding beam physics (INSTR99), INFN Pisa, Pisa, Nov, 1999.

[18] P. Hough, *Machine Analysis Of Bubble Chamber Pictures*, Conf. Proc. **C590914** (1959) 554.

[19] J. Albrecht *et al.*, *Impact of hardware choices on the LHCb upgrade trigger strategy*, Tech. Rep. LHCb-INT-2013-035. CERN-LHCb-INT-2013-035, CERN, Geneva, Jun, 2013.

[20] E. Bowen and B. Storaci, *VeloUT tracking software for the LHCb Upgrade*, Tech. Rep. LHCb-PUB-2013-023. CERN-LHCb-PUB-2013-023. LHCb-INT-2013-056, CERN, Geneva, Dec, 2013.

[21] J. Albrecht, V. Gligorov, and G. Raven, *Review Document: Full Software Trigger*, Tech. Rep. LHCb-INT-2014-017. CERN-LHCb-INT-2014-017, CERN, Geneva, Mar, 2014. On behalf of the the HLT software group.

[22] Z.-H. Chen, A. W.-Y. Su, and M.-T. Sun, *Resource-efficient fpga architecture and implementation of hough transform.*, IEEE Trans. VLSI Syst. **20** (2012), no. 8 1419.

[23] J.-P. Cachemiche, B. Jost, R. Le Gac, and N. Neufeld, *Estimation of the TPU cost*, Tech. Rep. EDMS1360078C, CERN, Geneva, Feb, 2014.