Journal of Physics:
Conference Series

PAPER • **OPEN ACCESS**

Real-time cluster finding for LHCb silicon pixel VELO detector using FPGA

To cite this article: Federico Lazzari *et al* 2020 *J. Phys.: Conf. Ser.* **1525** 012044

Real-time cluster finding for LHCb silicon pixel VELO detector using FPGA

Federico Lazzari[1,2,5], Giovanni Bassi[2,3], Riccardo Cenci[2,3], Michael J Morello[2,3], Giovanni Punzi[2,4]

[1] Università degli Studi di Siena, via Banchi di Sotto 55, 53100 Siena, IT
[2] INFN sezione di Pisa, Largo B. Pontecorvo 3, 56127 Pisa, IT
[3] Scuola Normale Superiore, Piazza dei Cavalieri 7, 56126 Pisa, IT
[4] Università di Pisa, Lungarno Pacinotti 43, 56126 Pisa, IT

E-mail: federico.lazzari@cern.ch

Abstract. In the Run-3 of LHCb, the High Level Trigger will have to process events at full LHC collision rate (30 MHz). This is a very challenging goal, and delegating some low-level tasks to FPGA accelerators can be very helpful by saving precious computing time. In particular, the 2D pixel geometry of the new LHCb VELO detector makes the cluster-finding process particularly CPU-time demanding. We realized and tested a highly parallel FPGA-based clustering algorithm, capable of performing this reconstruction in real time at 30 MHz event rate using a modest amount of hardware resources, that can be a viable alternative solution.

1. Introduction

The LHCb detector [1, 2] is a single-arm forward spectrometer covering the pseudorapidity range $2 < \eta < 5$, designed for the study of particles containing b- or c-quarks. One of the main limitations of the current detector is the maximum readout rate (1.1 MHz) of some sub-detectors. The Level-0 trigger, using the basic signatures available from calorimeters and muon chambers, reduces the event rate to the maximum rate at which the whole detector can be read out. The largest inefficiencies in the entire trigger chain, especially for purely hadronic decays, occur at the Level-0 decision [3]. Therefore, one of the main objectives of the LHCb upgrade is to remove this bottleneck by implementing a software High Level Trigger (HLT) capable of processing the full inelastic collision rate of 30 MHz.

The main challenge for a trigger-less readout of the entire LHCb detector is to build a cost-effective system that can handle the sizable throughput of 5 TB/s [4]. The LHCb collaboration has developed a new approach for the event building process based on high-bandwidth data-center link technology. The core part of the upgraded Event Builder consists of 500 dedicated servers located at ground level. These servers receive data from the front-end electronics via a readout unit embedded in each computer. Long distance optical links of 300 meters run between the front-end electronics located on the detector and the readout boards. The readout board input is via serial optical links, while the output is directed to the RAM using the PCI Express Gen3 protocol. All servers involved in the event building are connected by a large-scale network running on 100 Gbit/s bidirectional links. This allows the exchange of event fragments between

[5] Present address: INFN sezione di Pisa, Largo B. Pontecorvo 3, 56127 Pisa, IT

servers, with one of the computers elected to collect all the fragments that belong to the same collision and build the event. Then the elected computer sends the event data to the Event Filter Farm (EFF) that runs the HLT algorithms. The role of the event-builder is periodically rotated amongst all servers.

In the EFF, the full-software HLT reconstructs the events and reduces the event-accept rate consistently to the storage bandwidth of 10 GB/s. HLT is organized into two sequential stages, HLT1 and HLT2. This two-level structure helps coping with timing and selection requirements. HLT1 performs track reconstruction and kinematic/geometric selections. HLT2 adds offline-precision particle identification and track quality information to the selections. The key challenges of a full-software trigger come from the limitations due to CPU budget, amounting to 33 μs for a nominal farm size of 1000 HLT nodes [4]. Logically simple and repetitive tasks like reconstruction of pixel clusters can be very time consuming for general-purpose CPUs. Delegating some of the low-level processes to FPGA accelerators can be very helpful in saving computing time for higher level tasks.

2. Clustering of VELO pixels

The Vertex Locator (VELO) detects charged particle in the region closest to the interaction point. Its main purpose is to reconstruct primary and secondary vertexes with a spatial resolution smaller than typical decay lengths of b- and c-hadrons in LHCb ($c\tau \sim 0.01 - 1$ cm), in order to discriminate between them. It therefore plays a fundamental role to separate heavy flavors signals from the underlying background.

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. A particle crossing the VELO usually activates more than one pixel per module hit. In order to reconstruct the position of the hit, contiguous activated pixels must be grouped to form clusters. The 2-dimensional pixel geometry makes this task particularly time demanding. The current code performing cluster reconstruction on VELO data uses 6.1 μs of CPU time, corresponding to ∼18% of the total CPU time budget. This time can be effectively removed from HLT sequence if VELO clustering is performed in real-time on the data being read out, before the start of HLT processing. FPGAs are ideally suited for this kind of task due to their large bandwidths and capability of supporting highly parallelized algorithms.

3. A FPGA-friendly clustering algorithm

Our clustering algorithm is of rather general applicability, but some details have been optimized for the specific features of the VELO detector. 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 as shown in Figure 1). 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. For each of the 256 SP configurations, we pre-calculate the cluster position and load 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 a more structured approach. For this task, we created an algorithm based on a design developed within the INFN-RETINA R&D project[6]. The RETINA architecture is an approach to fast track finding based on a massively parallel architecture, that is very suitable for implementation in FPGA devices, and includes, amongst other things, a parallel cluster finder based on a matrix of cellular processors. Cellular processor is the elementary block that performs operation over the input data, stores the result and compares it with neighbouring

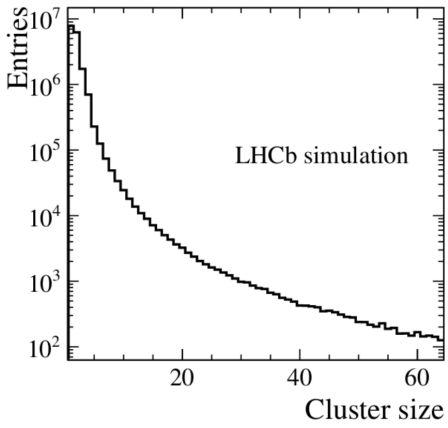

Figure 1. The distribution of cluster sizes in 10000 simulated minimum bias, at upgrade luminosity $\mathcal{L} = 2 \cdot 10^{33}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$.

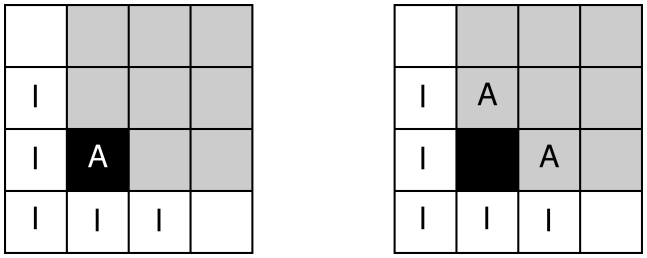

Figure 2. Pixel patterns associated to clusters. *A* and *I* indicate active and inactive pixels respectively, while the state of pixels without a label does not matter for pattern matching. Gray pixels constitute the cluster candidate while black pixels are the pixels of the cellular processor performing the cluster finding.

cellular processors. The cellular processor written for the clustering stores the status of a single pixel and, reading the status of neighbouring cellular processors, it finds cluster.

Originally the entire space of parameters is mapped with a matrix of cellular processor. 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 modified the original method to save space, creating a chain of smaller matrices (5×3 SPs that is 10×12 pixels). These matrices do not map to a specific VELO region until they are initialized. When an uninitialized matrix receives a SP in input, it fills the central region of the matrix with the pixels status and calculates the coordinates of the neighbouring SPs. Further SPs input to the matrix are compared with the previously calculated coordinates. In case of a match, the pixels status is used to fill the right position in the matrix, otherwise the SPs are passed on to the next matrix in the chain. After processing all input SPs, each cellular processor of every existing matrix, operating in parallel, checks for neighboring pixels forming one of the patterns in Figure 2. For every pixel matching a pattern, the 3×3 cluster candidate is then resolved by a lookup table.

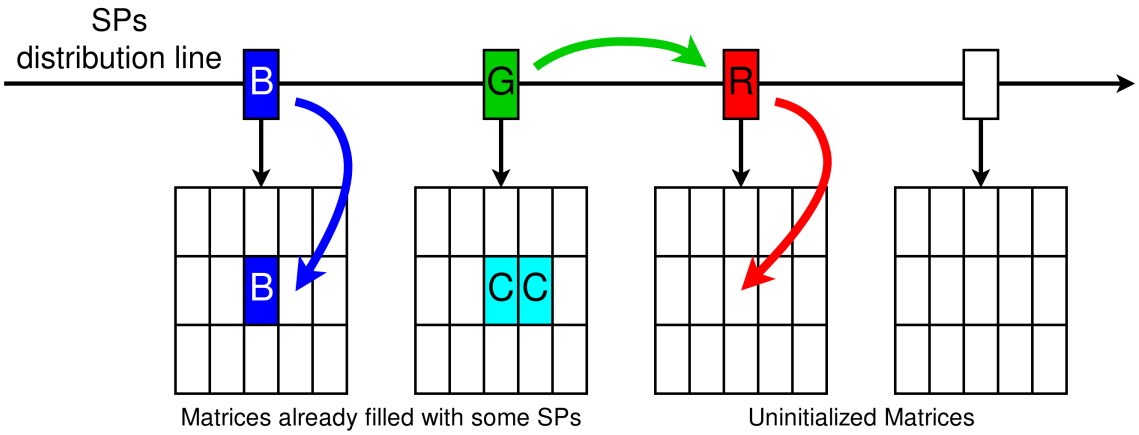

Figure 3. Example of SPs flowing along a matrix chain. SPs with same color (label) are neighbors.

Figure 3 shows how the chain of matrices is connected. At every clock cycle, all matrices move the input SP according to the rules previously discussed. The blue SP (B) in the distribution line belongs to the matrix, it fills the matrix. The green SP (G) does not belong to the matrix, it moves forward. The red SP (R) has reached an non-initialized matrix, so it is used to fill the matrix center.

4. Physics performance of the algorithm

The algorithm described in the previous section is different from CPU implementations in that in the case of clusters larger than the 3×3 mask, only a subset of pixels are used in determining the cluster characteristics. Although such clusters are uncommon, it is important to measure the consequences of this approximation, to verify their possible impact on the physics performance of the overall tracking reconstruction.

For this purpose, a bit-level simulation of the FPGA clustering algorithm has been produced 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.

These comparisons use the standard LHCb matching criteria [7] defining correctly reconstructed tracks: a *reconstructed track* matches a Monte Carlo (MC) track if 75% of the clusters used in the reconstructed track belong to the MC track; a *clone track* is any additional reconstructed track matching the same MC track; a *ghost track* is a reconstructed track not associated to any MC track. Based on this, we define reconstruction efficiency, clone rate and ghost rate as:

$$\varepsilon = \frac{\#\,tracks\,matched\,not\,clone}{\#\,tracks\,generated} \quad Clone = \frac{\#\,clone\,tracks}{\#\,tracks\,reconstructed}$$

$$Ghost = 1 - \frac{\#\,tracks\,matched}{\#\,tracks\,reconstructed}$$

We performed our tests on a sample of 25000 minimum-bias events, simulated at upgrade conditions: center of mass energy $\sqrt{s} = 14\,\text{TeV}$ and luminosity $\mathcal{L} = 2 \cdot 10^{33}\,\text{cm}^{-2}\text{s}^{-1}$.

Table 1. Summary of physics performances of the HLT1 tracking algorithm for different types of track, using clusters produced inside HLT1 and the FPGA.

Track type		HLT1	FPGA	FPGA – HLT
VELO	ε	97.720% ± 0.010%	97.636% ± 0.010%	−0.083% ± 0.014%
	clone	2.864% ± 0.011%	2.907% ± 0.011%	0.043% ± 0.015%
Long	ε	99.3439% ± 0.0070%	99.2727% ± 0.0074%	−0.071% ± 0.010%
	clone	1.281% ± 0.010%	1.339% ± 0.010%	0.058% ± 0.014%
Long from b	ε	99.85% ± 0.11%	99.55% ± 0.18%	−0.30% ± 0.21%
with cut on p	clone	1.04% ± 0.28%	0.97% ± 0.27%	−0.07% ± 0.39%
	Ghost	0.4462% ± 0.0031%	0.5081% ± 0.0033%	0.0619% ± 0.0045%

Table 1 reports a summary of physics performance comparisons regarding different types of tracks in LHCb. VELO-tracks are defined to have clusters on three or more VELO layers. T-tracks have at least one x and one *stereo* cluster in each tracking station downstream the LHCb

magnet. If a track is a VELO- and T-track at the same time, then it is a long-track. Long-tracks daughters of b-hadrons with $p > 3\,\text{GeV}/c$ and $p_T > 0.5\ \text{GeV}/c$ are labeled "Long from b with cut on p"; they are the most commonly used track in LHCb b-physics analyses.

All comparisons in this table show that the tracking performances obtained using clusters produced by our fast reconstruction algorithm are essentially equivalent for all practical purposes to those obtained from the default CPU algorithm.

5. Performance of hardware implementation

Having verified the good performance of our new algorithm, we proceeded to its implementation in an actual FPGA system. The necessary firmware was written in VHDL language in order to fully exploit the FPGA potential in terms of parallelization, timings, and resources usage. We used as testbed for our system a prototyping board with 2 Intel Stratix-V FPGAs, each carrying about 1M logic elements [8]. The maximum clock frequency is 650 MHz. The FPGA used for the test is comparable to the FPGA used on the LHCb DAQ board. The cluster finder logic needed to process one of the 52 readout boards of the VELO occupies less than 20% on one of the two Stratix-V FPGAs. This amount of logic easily fits within most modestly-sized FPGAs currently on the market; it might even be possible to fit it within the already-existing FPGAs performing the readout and formatting of VELO data.

A crucial quantity is the rate of events that our system can process, that must at least match the average LHC crossing rate, due to the aforementioned lack of an L0 preselection before data readout. This rate is inversely proportional to the average number of SPs in a event, which is different for each VELO module (Figure 4). As the throughput needs to be guaranteed over the whole detector, we performed our test with simulated data from the VELO module having the highest occupancy. We took a subset of 1000 events used for the physics performance study, load the SPs into RAM inside the chip, and feed them to the clustering processor in a continuous loop.

The system run comfortably without errors at a clock frequency of 350 MHz (out of a 650 MHz nominal maximum for our chip model), providing a measured event rate of 37.9 MHz (Figure 5), amply sufficient to sustain the target rate of 30 MHz readout.

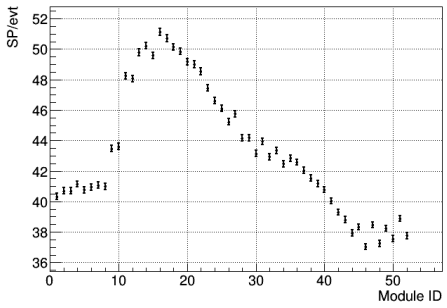

Figure 4. Average number of SPs per event for each VELO module, at upgrade luminosity $\mathcal{L} = 2 \cdot 10^{33}\,\text{cm}^{-2}\text{s}^{-1}$. Module 16 is the closest to the nominal interaction point.

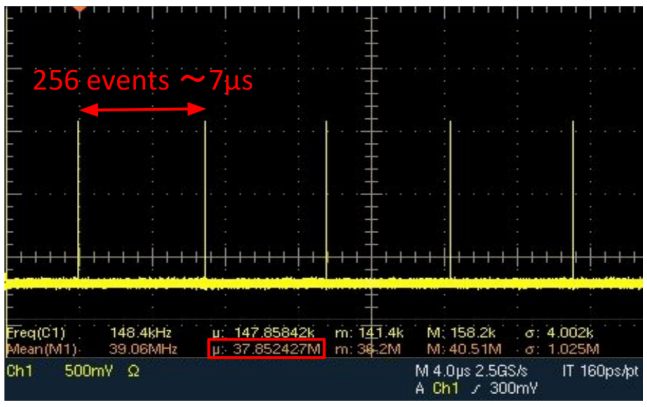

Figure 5. Oscilloscope screen shot showing the throughput test result. The FPGA board outputs a signal every 256 events processed.

6. Conclusion

In order to increase the trigger chain efficiency, the LHCb upgrade will have a trigger-less readout system, and the HLT software will process the full inelastic collision rate of 30 MHz.

The key challenges of a full-software trigger come from the limitations due to CPU budget. Assuming 1000 HLT nodes, the time budget for processing an event is 33 μs. VELO clustering on CPU consumes 6.1 μs, $\sim$18% of CPU time budget. So we designed a clustering algorithm for FPGA with physics performances essentially indistinguishable from CPU clustering. We measured a throughput of 37.9 MHz, thus the clustering on FPGA can run in real-time. Considering the time required by HLT to load clusters, this algorithm saves 4.5 μs of HLT execution time per event, which is equivalent to $\sim$14% of the total HLT time budget.

The main challenge for the trigger-less readout is to build a cost-effective system that can handle the sizable total bandwidth of 5 TB/s. Dropping SuperPixels we can also reduce VELO readout bandwidth by $\sim$24% with benefits to the data distribution system.

The clustering needs a relatively small amount of logic and memory, so even an FPGA with limited resources is well suited for this task.

References

[1] LHCb Collaboration 2008 *The LHCb Detector at the LHC* LHCb-DP-2008-001
[2] LHCb Collaboration 2014 *LHCb Detector Performance* CERN-LHCC-2014-002
[3] LHCb Collaboration 2011 *Letter of Intent for the LHCb Upgrade* CERN-LHCC-2011-001
[4] LHCb Collaboration 2014 *LHCb Trigger and Online Upgrade Technical Design Report* CERN-LHCC-2014-016
[5] LHCb Collaboration 2013 *LHCb VELO Upgrade Technical Design Report* CERN-LHCC-2013-021
[6] Cenci R *et al* 2017 *Proc., Topical Workshop on Electronics for Particle Physics: Santa Cruz, CA, USA, September 11-15, 2017* p 136
[7] LHCb Collaboration 2014 *LHCb Tracker Upgrade Technical Design Report* CERN-LHCC-2014-001
[8] Dini Group, `https://www.dinigroup.com/web/DNS5GX_F2.php`