EPJ Web of Conferences **127**, 00005 (2016) DOI: 10.1051/epjconf/201612700005
Connecting The Dots 2016

First Results of an "Artificial Retina" Processor Prototype

Riccardo **Cenci**[1,3,a], Franco **Bedeschi**[3], Pietro **Marino**[1,3], Michael J. **Morello**[1,3], Daniele **Ninci**[1,2], Alessio **Piucci**[1,2], Giovanni **Punzi**[1,2], Luciano **Ristori**[4], Franco **Spinella**[1], Simone **Stracka**[1,2], Diego **Tonelli**[5], and John **Walsh**[1]

[1] *INFN Sezione di Pisa, Italy*
[2] *Università di Pisa, Italy*
[3] *Scuola Normale Superiore, Pisa, Italy*
[4] *Fermi National Accelerator Laboratory, Batavia, IL, USA*
[5] *CERN, Geneva, Switzerland*

Abstract. We report on the performance of a specialized processor capable of reconstructing charged particle tracks in a realistic LHC silicon tracker detector, at the same speed of the readout and with sub-microsecond latency. The processor is based on an innovative pattern-recognition algorithm, called "artificial retina algorithm", inspired from the vision system of mammals. A prototype of the processor has been designed, simulated, and implemented on Tel62 boards equipped with high-bandwidth Altera Stratix III FPGA devices. The prototype is the first step towards a real-time track reconstruction device aimed at processing complex events of high-luminosity LHC experiments at 40 MHz crossing rate.

1 Introduction

With the luminosity increase expected at the Large Hadron Collider (LHC) in the next years, the available amount of collected data will scale by more than one order of magnitude. For many years, advancements in electronics technology (i.e. Moore's law) were able to cover for most of bandwidth increase due to higher luminosity and higher number of channels. However, this is not valid anymore, and today's technology advancements, when available, carry an additional development cost (compare a simple clock speed increase to software development for multi-core CPU), while physics complexity is still growing (luminosity, event pile-up, higher precision measurements). In some cases big steps forward in algorithms and architectures are already required, but it will be the default when moving to LHC Phase-2.

While in the past for one-collision events requirements on simple quantities (e.g. total calorimeter energy, number of muons, high transverse-momentum tracks) were sufficient to lower the bandwidth, multiple-collision events can contain interesting processes in almost any event, and different solutions, like real-time reconstruction, need to be explored. Real-time reconstruction of charged particle trajectories can help selecting events and reducing data size, but it will bring larger combinatorial problems, that require higher parallelization to be solved within typical latencies. In the '90s the Collider Detector at Fermilab (CDF) demonstrated the possibility to reconstruct in real-time two-dimensional tracks

[a] e-mail: riccardo.cenci@pi.infn.it

from clusters of aligned hits in a large drift-chamber [1]. The algorithm used by CDF was based on matching hits in the detector with precalculated patterns, and an updated version of the same approach is at the root of other current projects for LHC experiments [2, 3]. Performing a similar task at the LHC crossing rate is very problematic, due to the large combinatorial and size of the associated data flow that would require a massive network infrastructure and computing power. The ATLAS experiment is implementing an updated version of the CDF algorithm [2], called Fast TracKer (FTK). The typical number of clock cycles required by the FTK trigger to reconstruct an event is around 20,000, while the same number for a low-level trigger at the high-luminosity LHC is estimated to be around 25, so this solution cannot be used at the first-level trigger. An alternative approach developed for the CMS experiment is focused on reducing combinatorial locally at the detector level using pairs of tracking layers at short distance (1-4 mm) [4]. The local readout electronic searches for pairs of hits (called stubs), and sends out only the ones corresponding to tracks with transverse momentum greater than 2 GeV/c.

A new pattern-matching methodology has been recently proposed under the name "artificial retina" algorithm [5], inspired by the quick detection of edges in the visual cortex of mammals. Its aim is to further increase the parallelism of the pattern-matching process and decrease the number of stored patterns, to reduce latencies and hardware size. The purpose of the INFN-Retina project described below is to explore the possibility of using the artificial retina algorithm for a detailed tracking reconstruction at 40 MHz, compatible with the first step of online event selection at LHC, at a reasonable cost. A hardware prototype based on already-existing FPGA readout boards is under development, to explore the potential of this new approach in a realistic experimental environment, albeit at lower rates. The main purpose of the prototype is to show the possibility of reconstructing quality tracks using the same hardware resources that are normally required just for reading out the raw hits.

2 The "artificial retina" algorithm

The "artificial retina" algorithm, elsewhere referred to as retina algorithm, was inspired by the first stage of mammal vision. Recent experimental studies show that specific neurons, called receptive fields, receive signals only from specific regions of the retina, in order to reduce the connectivity and save bandwidth. The neurons are tuned to recognize a specific shape and the response is proportional to how close are the stimulus shape and the shape for which the neuron is tuned to. Generated in parallel, the responses of neurons are then interpolated to create a preview of image edges in about 25 ms, corresponding to about 25 clock cycles and perfectly matching the requirement for processing an event mentioned above. Therefore those concepts of early vision can be used to realize a viable highly-parallel implementation of an "analog" pattern-matching system, where each pattern is assigned a continuous level of "matching", rather than a simple binary response. The mathematical aspects of the algorithm have some similarities with the "Hough transform" [6, 7], a method already applied for finding lines in image processing; however, the main challenge here is the design of the physical layout and the development of an implementation capable to sustain the event rate at high-luminosity LHC experiments [8].

For configuring the algorithm, the space of track parameters is divided into cells, which mimic the neurons connected to the receptive fields of the retina. The center of each cell corresponds to a specific track in the detector that intersects the layers in spatial points called *receptors*. A first mapping connects each cell with the receptors, as shown in Figure 1. For a group of contiguous cells, where variations of track parameters are small, the corresponding receptors in the detector layers would belong to a limited area. A second mapping, also shown in Figure 1, connects clusters of cells to

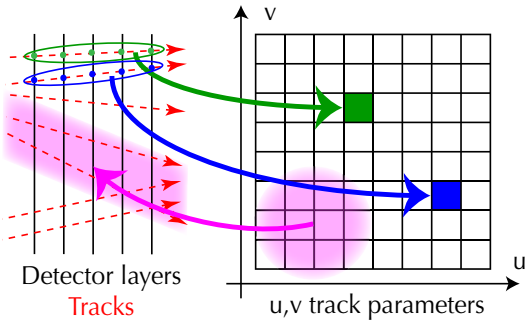
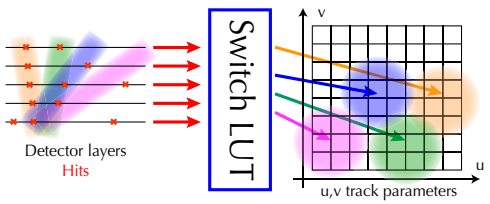
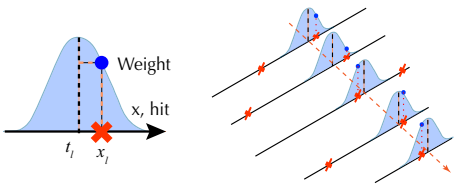


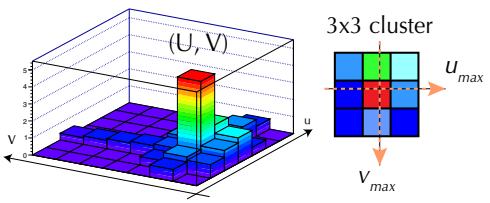
Figure 1: Retina mappings for tracks on a plane without magnetic field, where tracks can be described by two parameters u and v.



(a) Step 1



(b) Step 2



(c) Step 3

Figure 2: Processing steps of track reconstruction with the artificial retina algorithm.

areas of the detector, and this information is recorded in a look-up-table (LUT). A commercial PC is enough to generate the two mappings starting from simulated tracks.

The track reconstruction, to be implemented on high-speed FPGA devices, has three steps. During step 1, also referred to as the *switching* step, detector hits are distributed only to a reduced number of cells, according the LUT's created in the configuration phase, as shown in Figure 2a. In step 2, sketched in Figure 2b, for every incoming hit a Gaussian weight w is accumulated in each cell by a logic unit called "engine". The weight is proportional to the distance to the receptor and defines as follows:

$$w = \exp\left(-\frac{d_l^2}{2\sigma}\right)$$

where d_l is the distance, on the layer l, between the hit and the corresponding receptor, and σ is a parameter of the algorithm, that can be adjusted to optimize the sharpness of the response of the receptors. After all the hits from the same event have been processed, in step 3 tracks can be identified looking for local maxima of the accumulated weight over the cell's grid. For a track resolution similar to offline reconstruction, the grid does not require a high granularity, because significantly better precision can be obtained by computing the centroid of the accumulated weights for the cells surrounding each maximum, as shown in Figure 2c.

Compared to other algorithms, the retina method takes advantage of two levels of parallelization. First, each cell processes in parallel hits from a limited region of the detector, reducing the required input bandwidth for the single cell. Moreover, if time information is associated to every hit and weights are accumulated separately for a small number of consecutive events, events can be processed simultaneously. The latter allows to feed data continuously to each cell, because cells are not receiving the same number of hits for every event.

Another important feature is the total system bandwidth: the bandwidth increases significantly during the hit distribution, because multiple copies of the same hit can be produced, but shrinks down when only the information about local maxima is kept in the last step. The larger bandwidth can be physically managed only because it is limited to one stage of the system. Curiously, a similar behavior of the bandwidth is found in the first stage of mammal vision as well.

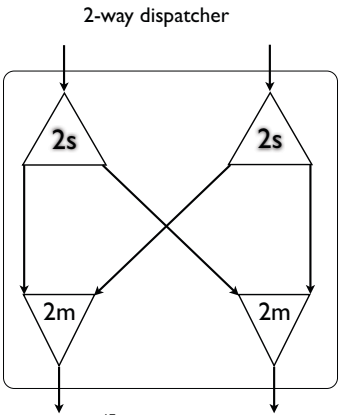
3 Architecture of the track processing unit

The implementation of the retina algorithm on a real device is called track processing unit (TPU). The first step of the algorithm is achieved using a switching network able to redirect hits from the readout module to any engine of the system. The network increases the global bandwidth making multiple copies of hits, but it never sends a hit to all the engines, reducing the input bandwidth for them. The network is assembled using basic 2-way dispatchers (2d) with two inputs and two outputs: data on any input can be redirected to any output using two splitters and two mergers, as shown in Figure 3a. To implement a dispatcher with 2^n inputs/outputs, shown in Figs. 3b and 3c, we need N 2-way dispatchers connected together, where

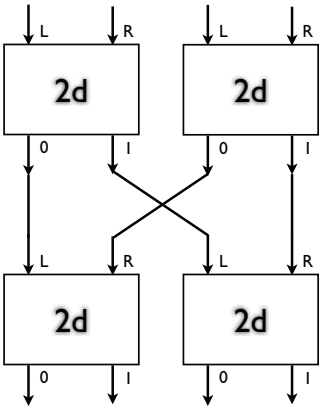
$$N(n) = 2N(n-1) + 2^{n-1}.$$

The dispatchers work as a pipeline and the latency is proportional to n. The space parameters grid is divided into matrices of engines that fit inside a single device. Maxima can be found on any cell of the grid keeping a one-cell overlap between the matrices. The switching network can be easily scaled adding enough inputs to receive data from the readout modules and outputs to transfer data to all the devices used for the engines.

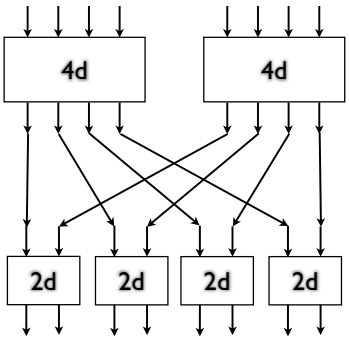
The best device to implement a prototype of the TPU system is FPGA, due to their high flexibility, and shorter time for designing and test compared to standard ASIC devices. Moreover, FPGA performances are still increasing at a steady pace, taking advantage of new silicon technology (14-16 nm), increasing the number of logic elements (up to 5.5 millions), and including faster connection technology (up to 30 Gbit/s on a single serial link). After testing the system performances on FPGA, the firmware can be always transferred on ASIC devices for mass production. Preliminary studies shows that a TPU system for a modern detector can be implemented using 128 FPGA devices [9], and detailed numbers for the estimated resources and cost are reported on Table 1. All the required devices are already available on the market, and devices available in few years can improve the performances by a factor ten.



(a) 2d Dispatcher



(b) 4d Dispatcher



(c) 8d Dispatcher

Figure 3: Schemes of dispatchers with 2, 4, and 8 inputs/outputs.

4 First prototype design

4.1 Geometry details and simulation studies

To validate the algorithm and measure its performances, a TPU prototype was designed to reconstruct tracks in a small tracker. A telescope with six single-coordinate layers (50 cm long, 20 cm wide) and no magnetic field [10], shown in Figure 4, was used as model. Any track going through the telescope has been parametrized with two variables: the x coordinates on the first and last layer, below referred to as u and v. The GEANT4 simulation of the tracker [10] shows that real tracks are likely to be along z-axis, corresponding to the region around the diagonal in the u-v-plane.

Fundamental design parameters of the system are: the minimum number of cells needed to cover the space parameters and to obtain track resolution; the size and inter-connectivity of the system. The retina algorithm for the telescope was simulated using a C++ emulator [9], showing that resolution and efficiency similar to offline tracking systems can be achieved using only 3,000 cells to cover the diagonal region mentioned above. The emulator was also used to compute the switching network configuration, and as debugging tool, providing data at different stages of processing. The prototype

Table 1: Estimation of resources and cost for a track processing unit that can reconstruct tracks in a modern HEP detector at LHC.

crossing frequency	40 MHz	HL-LHC design
# of layers	10	2 in fringe B field
number of hits per layer per crossing	300	Lumi = 10^{33}-$10^{34} cm^{-2} s^{-1}$
number of bits per hit	40 bits	2D pixels + time
total hit bandwidth	5 Tbit/s	(readout)
number of engines	50k	1/20 of precalculated patterns used by CDF
engines per FPGA	800	
number of FPGAs (1M logic elements)	64+64	switch + engine boards
links per FPGA	64	@10 Gbit/s (standard)
max total bandwidth	40 Tbit/s	max 8 copies per hit
clock speed	500 MHz	
cycles to process one event	150+350	switching + engine steps
latency	1 µs	
number of events in pipeline	16	
cost	1 M€	5 k€ switch FPGA, 10 k€ engine FPGA

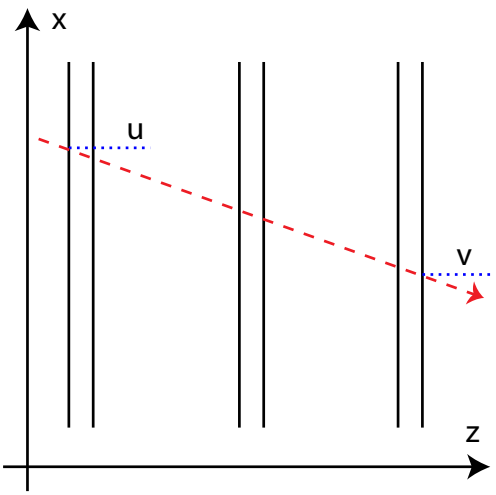


Figure 4: Sketch of the 6-layer telescope used for the prototype, showing also the two parameters u and v used to represent a track.

was designed to be implemented on a currently-available electronic board (see description below) that includes several large and high-bandwidth FPGA devices. The diagonal region in the u-v-plane was covered using matrices of cells, each one of them fitting inside a single device of the chosen board. The matrices in different devices overlap by one cell to be able to find local maxima in all the cells.

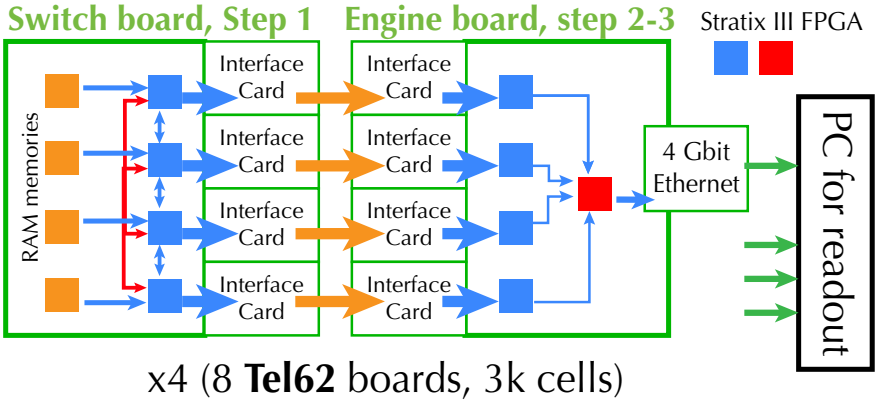


Figure 5: Architecture of the first prototype using Tel62 boards.

4.2 Board details

The board Tel62, developed by INFN Pisa for the NA62 experiment [11], was chosen to implement the TPU prototype because it carries multiple large and high-bandwidth FPGA chips and provides fast interconnection between them. The board includes four Stratix III chip for data processing, each with approximately 200k logic elements. Each chip is connected through a high-speed link (10 Gbit/s) to a master chip (another Stratix III) that collects the processed data and controls the other FPGAs. The main clock is 40 MHz, while the processing internal to the FPGAs is done at 160 MHz. Each processing chip is connected to a 2 Gbyte DDR2 RAM through a high-speed bus (40 Gbit/s), to a mezzanine connector that can host various interface cards for I/O (5 Gbit/s), and to the neighbouring FPGAs (2.5 Gbit/s). The master FPGA is also connected to a mezzanine card with four Ethernet ports (2.5 Gbit/s). Implemented on an embedded PC that sits on the board, a slow-control interface can be used to monitor all the devices and to perform various standalone tests.

4.3 Algorithm implementation

The Tel62 board is designed for a standard DAQ system, where bandwidth typically reduces following the data-flow (only selected data move to the next step) and data streams do not have many connections between all of them. In any TPU system, however, the bandwidth increases in the first step and data have to be exchanged also laterally within the switching network. Therefore, two boards were connected together: the data-flow was reversed in the first one, where the switching network was implemented (switch board), and a newly designed interface card was used to connect a second board, where the processing engines were implemented (Engine board), as shown in Figure 5. This configuration is fully consistent with the implementation of the system proposed in Ref. [9], so the prototype results are meaningful for future applications to real detectors.

The processing engine is implemented in a fully pipelined mode, so each engine can receive one hit for each clock cycle [12]. All the engines in a chip receive the same hit sequence and, after a special end-event (EE) hit is received, engine values are copied to the following stage where the search for local maxima starts, while the next hit sequence enters. The latency between the input of EE hit and the output of maximum data is fixed. As explained in more detail below, we fit approximately 200

engines on each Stratix III device (EP3SL200F1152), therefore 16 chips, corresponding to four pairs of Tel62 boards, are enough to implement the whole system.

4.4 Hardware development details

The firmware for the different FPGA chips on the boards was developed using the software "HDL Designer" by Mentor Graphics and the Altera specific software "Quartus II". Most of the firmware is written in VHDL using generic components configurable through parameters, so they can easily be reused for systems on larger devices. The same firmware is loaded into all the four chips on the boards, so specific values for engine receptors and switching network LUT's have to be loaded through the slow control and stored on internal RAM's.

The firmware for the switch board chips is based on a 4d dispatcher block with four inputs and four outputs already described above. Each splitter in the basic 2d dispatcher copies the input data to zero, one, or both outputs, according to a 2-bit LUT value based on the hit coordinate and layer. Additional logic is also required to implement the correct propagation of EE hits: generated by the readout, they arrive separately on each input, and, after the switching step, hit sequences on each output have to contain all the appropriate hits from the event, tailed by a single EE hit. Each of the four switching block outputs corresponds to a different chip on the engine board. Because each engine chip is connected only to one switch chip, all the switch outputs between chips are properly redirected, using the lateral connections or through the master chip. Then, the data streams are merged from the redirected switch outputs and transferred to the engine board through the interface card, as shown in Figure 5.

The firmware for the processing chips of the engine boards is based on a matrix of engines. Each engine completely surrounded by other ones has additional logic to check if the accumulated value is greater than the first neighbors one. The maximum search requires a large number of interconnections between engines, limiting the number of engines that can be fit in our Stratix III FPGA to a 16x15 matrix[1]. A priority encoder manages the serial transfer of resulting maxima to a single output FIFO. Due to the reduced bandwidth achieved at this stage, maxima from all the processing chips are moved to the master chip, and then sent out to a PC through the Ethernet connectors.

5 First results

The prototype is an advanced stage of development and some tests were already performed on a switch board connected to an engine board. The basic firmware for all the system devices has been written and the logic behavior verified using the "ModelSim" software by Mentor Graphics. Simple hit sequences were loaded on internal FIFO's of the switch board chips, and the hits were correctly dispatched to the proper output channels, together with the EE flags. During these tests, all the logic for the switching, internal and between chips, was running properly at the maximum clock rate of the board (160 MHz). The data transmission between the switch and engine boards was successfully run at 80 MHz, using two data channels for each interface card pair. Hit sequences were also loaded on the input FIFO of the engine chip: the hits were processed and the correct maxima data were received by the readout PC through Ethernet connection. Also here the internal logic for the engine is running properly at 160 MHz clock rate. Because the engine input latency is proportional to the number of hits in the sequence and having hit sequences for realistic events with an average of 70 hits per layer, the maximum rate sustainable by the engine board is 1.8 MHz with a latency smaller than 1 µs. The final integration and tests of the firmware are currently ongoing.

[1]The place and route tools from Quartus II report 90% of resources used on our specific device.

6 Future plans

After validating the algorithm and measuring its performances for a small tracker with available boards, another prototype will be built in the next months to test the TPU performances with data from a 3D tracker partially inserted in a magnetic field. For this geometry the algorithm requires three additional parameters to describe a track, but considering a parameter space with five dimensions will increase too much the number of cells. If variations for some parameters can be approximated to small perturbations, we can factorize the parameter space into the product of two subspaces [9]. Therefore we perform the full pattern recognition using only a two-dimensional grid for two parameters, and then extract the other ones using lateral cells (six of them for each cell of the main grid) that represent a track with the same values for the first two parameters and slightly different ones for the remaining three. The lateral cells and the computation of 3D distances require additional logic, so the engine implementation has to be extended and the maximum number of engines fitting in one chip has to be updated. The fundamental parameters of the prototype are the minimum clock speed to reconstruct tracks at LHC Run3 rate, and the size and throughput of the FPGA device needed to implement the system. The prototype will consist only of a basic unit of the system implemented on an ASIC prototyping board with two very large and very high-bandwidth FPGA's. Multiple connections between the two devices will allow to test different configurations, e.g. a standalone switching network connected to a standalone engine matrix, both receiving data at full speed prepared using the algorithm emulator.

7 Conclusions

Computing and storage available for future experiments at high-luminosity LHC will not be able to cope with the increase of data rate, therefore more processing will have to be performed "online" to reduce event rate and size. The methods already employed for data triggering may not scale well and alternative advanced solutions should be explored, like the "artificial retina" algorithm, that exploits higher degrees of parallelization and provides analog response. The INFN-Retina project aims to demonstrate the feasibility at reasonable cost of a system based on this algorithm able to reconstruct tracks at rates expected for LHC Run3. A first, sizable hardware prototype of a retina tracking system with 3,000 patterns is under advanced development, based on already existing FPGA readout boards. First results show the track-processing system based on our algorithm is feasible, and essential steps are successfully implemented on the real board at the nominal clock speed. The system is capable of reconstructing tracks at a 1.8 MHz event rate, using boards that had originally been designed for 1 MHz readout-only functionality. Moreover, the requested additional hardware to implement the tracking functionality is also comparable with respect to what is needed for the readout-only function. Performances mentioned above are expected to be easily scalable to higher speeds, as much larger and faster FPGA devices are already available today on the commercial market, and another prototype to perform speed test is expected to be built in the next months. Further developments and synergies with fast and smart tracking detector may lead to future experiments with detector-embedded data reconstruction.

References

[1] G.W. Foster, J. Freeman, C. Newman-Holmes, J. Patrick, Nucl. Inst. & Meth. A **269**, 93 (1988)
[2] M. Shochet, L. Tompkins, V. Cavaliere, P. Giannetti, A. Annovi, G. Volpi, Tech. Rep. CERN-LHCC-2013-007, ATLAS-TDR-021, CERN, Geneva (2013), https://cds.cern.ch/record/1552953

[3] A. Tapper, D. Acosta (CMS collaboration), Tech. Rep. CERN-LHCC-2013-011. CMS-TDR-12, CERN (2013), https://cds.cern.ch/record/1556311
[4] G. Hall, Nucl. Inst. & Meth. A **824**, 292 (2016)
[5] L. Ristori, Nucl. Inst. & Meth. A **453**, 425 (2000)
[6] P. Hough, Proc. Int. Conf. High Energy Accelerators and Instrumentation **C590914** (1959)
[7] P. Hough, US Patent **3069654** (1962)
[8] W.H. Smith, *Trigger and Data Acquisition for hadron colliders at the Energy Frontier* (2013), arXiv:1307.0706
[9] A. Abba, F. Bedeschi, M. Citterio, F. Caponio, A. Cusimano, A. Geraci, F. Lionetto, P. Marino, M.J. Morello, N. Neri et al., Tech. Rep. LHCb-PUB-2014-026, CERN-LHCb-PUB-2014-026, CERN, Geneva (2014), https://cds.cern.ch/record/1667587
[10] A. Piucci, Master's thesis, University of Pisa, Pisa, Italy (2014), https://etd.adm.unipi.it/theses/available/etd-06242014-055001/
[11] B. Angelucci, E. Pedreschi, M. Sozzi, F. Spinella, JINST **7**, C02046 (2012), http://iopscience.iop.org/article/10.1088/1748-0221/7/02/C02046
[12] D. Ninci, Master's thesis, University of Pisa, Pisa, Italy (2014), https://etd.adm.unipi.it/theses/available/etd-11302014-212637/