UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

Master Thesis

Development of algorithms for real time track selection in the TOTEM experiment

Author:
Nicola Minafra

Supervisors:
Dr. F. S. Cafagna
Dr. E. Radicioni

September 20, 2012

"That's the strange world we live in, that all the advances and understanding are used only to continue the nonsense which has existed for 2000 years."
Richard P. Feynman

I would like to thank my supervisors Francesco Cafagna and Emilio Radicioni for their teaching and mentoring. Their knowledge, their expertise and their patience were essential for my growth as a student and as a researcher.

I would also like to thank Joachim Baechler and the CERN TOTEM group, Gabriella Catanesi and the University of Bari, the INFN and the CERN Technical Student program, for supporting my research and this thesis.

I would like also to thank Michele Quinto, Alessandro Mercadante, Adrian Fiergolski and Valentina Avati, Jan Kaspar, Gueorgui Antchev and all colleagues of the TOTEM group and its spokesman Simone Giani for their help and for their friendship.

Finally, I would like to thanks my parents Giovanni and Giovanna, my sisters Anna Rita and Maria Paola, my girlfriend Marilena and all friends and colleagues that constantly motivate my life and my work.

Contents

Introduction **IX**

1 The TOTEM Experiment at LHC **1**
1.1 The Large Hadron Collider 1
1.2 The TOTEM Experiment 2
1.2.1 Latest results 6
1.3 General overview of the TOTEM detector system 7
1.3.1 T1 telescope 8
1.3.2 T2 telescope 9
1.3.3 Roman Pots 11
1.4 TOTEM Electronics System 13
1.4.1 VFAT: a common front-end chip 15
1.4.2 TOTFed 16
1.4.3 OptoRx 18
1.4.4 Firmware and data frame 19
1.5 The off-line software 21

2 Cluster searching algorithm **25**
2.1 The cluster searching Algorithm 25
2.1.1 *cluster*: the main brick 26
2.1.2 *bitStat*: clusters in a bitstream 26
2.1.3 *byteStat*: from byte to clusters using a LUT 26
2.1.4 *wordStat*: clusters in a buffer of words of arbitrary length 27
2.1.5 Usage in the TOTEM framework 28
2.2 Cluster per plane filter 29
2.3 Implementation in the *OptoRx* firmware 29
2.3.1 *bitStat*: clusters in a bitstream 30
2.3.2 *gohStat*: cluster analysis for a single fiber 33
2.3.3 *optoRxStat*: cluster analysis for all the *OptoRx* 36

3 Track recognition algorithms **39**
3.1 Tracks in Roman Pot detectors 39
3.2 Histogram of the hits 40
3.2.1 The *DynamicHistogram* class 42
3.2.2 Track recognition 43
3.3 Simplified Hough transform 45
3.3.1 Track recognition with a simplified Hough transform 47

4 Data Analysis **49**
4.1 Data set 49
4.2 Data analysis using the off-line software 50
4.3 Cluster algorithm performance 51
4.3.1 Firmware place and route 51
4.3.2 Cluster per plane filter performance 52
4.4 Performances of the track recognition algorithms 55
4.4.1 Performances of the histogram based transform algorithm 56
4.4.2 Performances of the Hough transform algorithm 64

Conclusion **71**

A Real-time application for cluster analysis **III**

List of Figures **VII**

List of Acronyms **XI**

Bibliografy **XIII**

Introduction

The TOTEM experiment at the LHC has been designed to measure the total proton-proton cross-section with a luminosity independent method and to study elastic and diffractive scattering at energy up to 14 TeV in the center of mass.

TOTEM detector system allows an optimum forward coverage for charged particles emitted by the proton-proton interactions. Indeed, two telescopes, T1 and T2, are installed on both sides of the interaction point and together allow the detection of at least one charged particle in 99% of the diffractive and non-diffractive events with diffractive masses above $\sim 3.4\ \frac{GeV}{c^2}$ [1]. On the other hand, elastic scattered protons are detected by Roman Pot stations, placed at 147 m and 220 m along the two exiting beams. These detectors can be moved very close to the beam and, thanks to dedicated runs with special optics configurations (high $beta^*$), it is possible to extrapolate the elastic cross-section for values of the four-momentum transfer down to 0 with only 9% of non-visible zone [2]. TOTEM physic program and the detector system will be described in more detail in chapter 1.

At the present time, data acquired by the detectors are stored on disk without any data reduction by the data acquisition chain. Given the computational capability of the already installed read-out electronics, it should be possible to implement some zero-suppression or event filtering algorithms, based on track recognition, to improve the efficiency of the data acquisition and to optimize the disk usage. To achieve this goal, fast and smart algorithms have to be developed, implemented and tested.

In chapter 2 a fast algorithm for cluster searching will be proposed. This algorithm is able to find clusters in the hit maps of the detectors, stored in the TOTEM data frame. Moreover, it is also possible to use the number of clusters to filter data acquired by Roman Pot detectors. At first, the algorithm has been implemented using the C++ programming language to test its capabilities and performance. Then, an hardware implementation has been designed and implemented in the firmware of the FPGA present in one of the front-end cards.

In chapter 3 two algorithms for track recognition will be proposed. In the Roman Pot detectors a track can be seen as a line in the detector's hit map. Hence, a track recognition algorithm has to search for clusters in the detector hit map and then search for aligned patterns of clusters. Two different approaches have been implemented: the first uses less

computational resources than the second one that is, instead, more accurate.

Finally, to test these algorithms, a data set has been carefully chosen to be representative of all data acquired by the TOTEM experiment during 2011 and 2012 data taking periods. In chapter 4, the results of the TOTEM off-line reconstruction software on this data set will be used to test the performance of the proposed algorithms.

Chapter 1

The TOTEM Experiment at LHC

The LHC (Large Hadron Collider) is the world's largest and highest-energy particle accelerator, hosted at CERN (European Organization for Nuclear Research), the world's largest particle physics laboratory. TOTEM (TOTal cross section, Elastic scattering and diffraction dissociation Measurement at the LHC) is the experiment at the LHC spanning the largest distance. Indeed, TOTEM detectors are positioned more than 400 m far from each other.

The TOTEM experiment will measure the total proton-proton cross section and it will study elastic scattering and diffractive dissociation. These studies are usually referred to as *forward physics* because of their topology; precise measurements in this field are crucial for both high energy and cosmic ray physics.

The detectors and electronics used by the TOTEM collaboration will be introduced in this chapter.

1.1 The Large Hadron Collider

The LHC at CERN is a two-ring superconducting hadron accelerator and collider installed in a 26.7 km long tunnel buried between 45 m and 170 m below the surface near Geneva.

Nowadays the LHC is the most powerful particles accelerator; scientists and engineers from the 20 European Member States and from many non-Member Countries, representing 608 Universities and Institutes and 113 nationalities, are working on the LHC and other experiments at CERN. Using this amazing machine it is possible to accelerate protons or ions to speeds very close to the one of light, and to collide them in special locations, called IP (Interaction Point)s. Very active international collaborations designed, built and run their detectors placed close to these Impact Points to study the products of the collision. Among these there are CMS, ATLAS, ALICE, and LHCb.

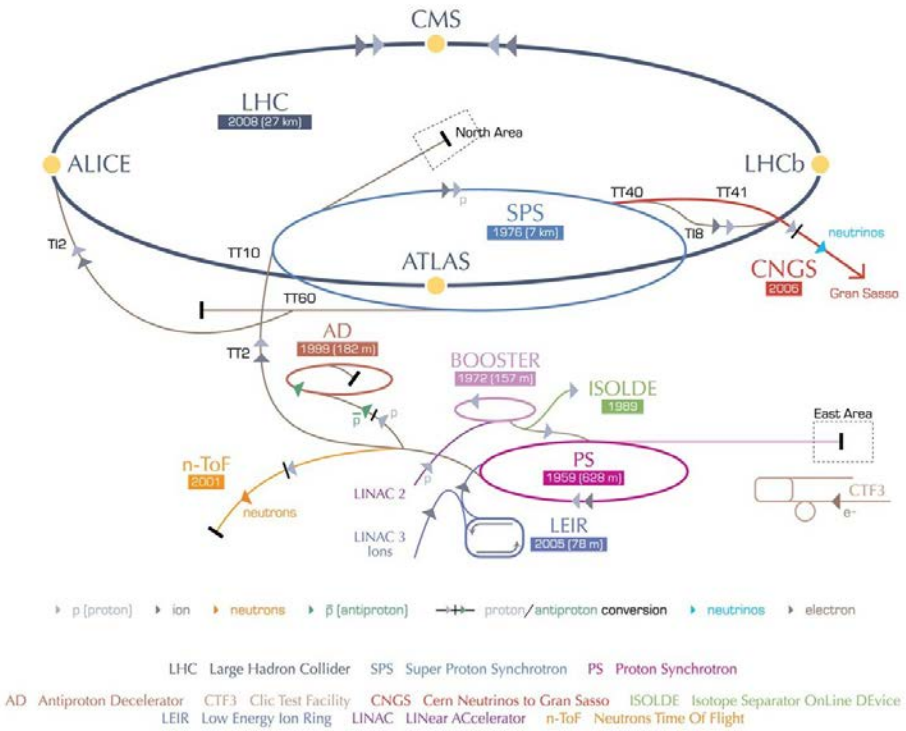


Figure 1.1: CERN's accelerator complex.

CMS (Compact Muon Solenoid) and ATLAS (A Toroidal LHC Apparatus) are general pourpose detectors. A huge part of their resources is focused on the Higgs boson hunting, the last missing particle forseen by the Standard Model. Recently, the two collaborations announced the discovery of a new boson [3][4] and its behavior is compatible with the Higgs particle. Moreover, CMS and ATLAS detectors are used also to look for signs of new physics, including extra dimensions and exotics interactions. ALICE (A Large Ion Collider Experiment) has been designed to study heavy ions interactions and to investigate a form of matter called *quark–gluon plasma* that is supposed to have existed shortly after the Big Bang. Finally, LHCb (Large Hadron Collider beauty) research is focused on the asymmetry between matter and anti-matter to answer open questions about the origin of our Universe.

1.2 The TOTEM Experiment

TOTEM (TOTal cross section, Elastic scattering and diffraction dissociation Measurement at the LHC) is an experiment whose detectors are located in the forward region of the IP (Interaction Point) shared with CMS and its main purpose is to measure the total

proton-proton cross-section using a luminosity-independent method.

The total cross section σ_{tot} can be thought as the effective area seen by two particles involved in a scattering process. Unfortunately, it is not possible to theoretically describe the behavior of σ_{tot} up to the LHC energy: some phenomenological models have been proposed, but a direct measurement is needed to confirm or reject them. Before the TOTEM experiment, direct measurement at these energies were performed only by observing the interactions of cosmic rays [5], as shown in Fig. 1.2; unfortunately, their uncertainties are too large to discriminate among the different models [6]. One more element underlining the importance of TOTEM results is that all LHC experiments need σ_{tot} to normalize the physical processes involved in their measurements.

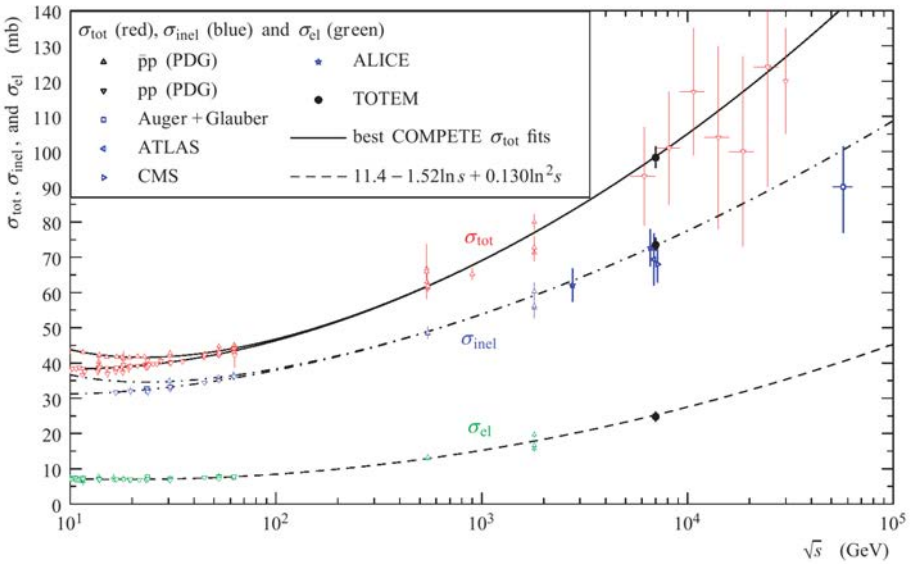


Figure 1.2: Compilation of total (σ_{tot}), inelastic (σ_{inel}) and elastic (σ_{el}) cross-section measurements [5].

In detail, the TOTEM measurement technique is based on the simultaneous estimate of the σ_{tot} and the luminosity $\mathcal{L}$. Thanks to the Optical Theorem it is possible to write:

$$\begin{aligned} \mathcal{L}\sigma_{tot}^2 &= \frac{16\pi}{1+\rho^2}\frac{dN_{el}}{dt}\bigg|_{t=0} \\ \mathcal{L}\sigma_{tot} &= N_{el} + N_{inel} \end{aligned} \tag{1.1}$$

where:

- N_{inel}: the inelastic rate;
- N_{el}: the total nuclear elastic rate;

- t: momentum transfer[1];
- $\frac{dN_{el}}{dt}\Big|_{t=0}$: the nuclear part of the elastic cross section;
- $\rho = \frac{\Re[f_{el}(0)]}{\Im[f_{el}(0)]}$;
- $f_{el}(0)$ is the forward nuclear elastic amplitude.

ρ can be estimated theoretically: $\rho \sim 0.14$, so the impact on $1 + \rho^2$ is small. The two equations set can be solved to find $\mathcal{L}$ and σ_{tot}:

$$\sigma_{tot} = \frac{16\pi}{1+\rho^2}\frac{\frac{dN_{el}}{dt}\Big|_{t=0}}{N_{el}+N_{inel}}\mathcal{L} = \frac{1+\rho^2}{16\pi}\frac{(N_{el}+N_{inel})^2}{\frac{dN_{el}}{dt}\Big|_{t=0}} \tag{1.3}$$

This method does not need a direct measurement of the luminosity $\mathcal{L}$; however, this will only be possible if all needed quantities can be computed.

To understand how to measure these quantities, in Fig. 1.3 the main event topologies for a proton-proton interaction are shown. The number of inelastic interactions N_{inel} includes all diffractive dissociation and, more generally, it includes all processes where a part of protons' kinetic energy "creates" new particles. These events can be detected in low rapidity regions. N_{el} is the number of elastic interactions, when kinetic energy is conserved. Usually, elastically scattered protons can be detected at very high rapidity (low t). Moreover, the elastic cross section can not be exactly calculated at $t = 0$. TOTEM will measure it down to $|t| = 10^{-3}\ GeV^2$ thanks to special runs with dedicated accelerator optics and it will be extrapolated to $t = 0$.

To perform these measurements TOTEM requires a unique coverage in pseudo-rapidity [2] on both sides of the interaction point to cover elastic and diffractive processes. To achieve this coverage, three different detectors have been chosen; all of them are tracking telescopes. A first telescope, close to the interaction point, is named T1 and it is made of CSC (Cathode

[1] In a two body scattering $a + b \to a + b$, defining the 4-momentums of in-going (p_1, p_2) and out-going (p_3, p_4) particles, the kinematics can be described using the Lorenz invariant Mandelstam Variables (s, t, u), that are defined as:

$$\begin{aligned} s &= (p_1 - p_2)^2 = (p_3 - p_4)^2 \\ t &= (p_1 - p_3)^2 = (p_2 - p_4)^2 \\ u &= (p_1 - p_4)^2 = (p_2 - p_3)^2 \end{aligned} \tag{1.2}$$

s represents the square of the cent re of mass energy, while t is the 4-momentum transfer squared.

[2] The pseudorapidity η is defined as $\eta = -ln(tan(\frac{\theta}{2}))$. Additionally, the rapidity y is defined as $y = \frac{1}{2}ln(\frac{E+p_z}{E-p_z})$ where E is the total energy and p_z is the momentum component parallel to the beam. For particle momentum $p \gg m$, the rapidity and pseudorapidity are approximately equal: $y \sim -ln(tan(\frac{\theta}{2})) \equiv \eta$, where m is the rest mass of the particle and θ is the angle between the beam and the scattered particle.

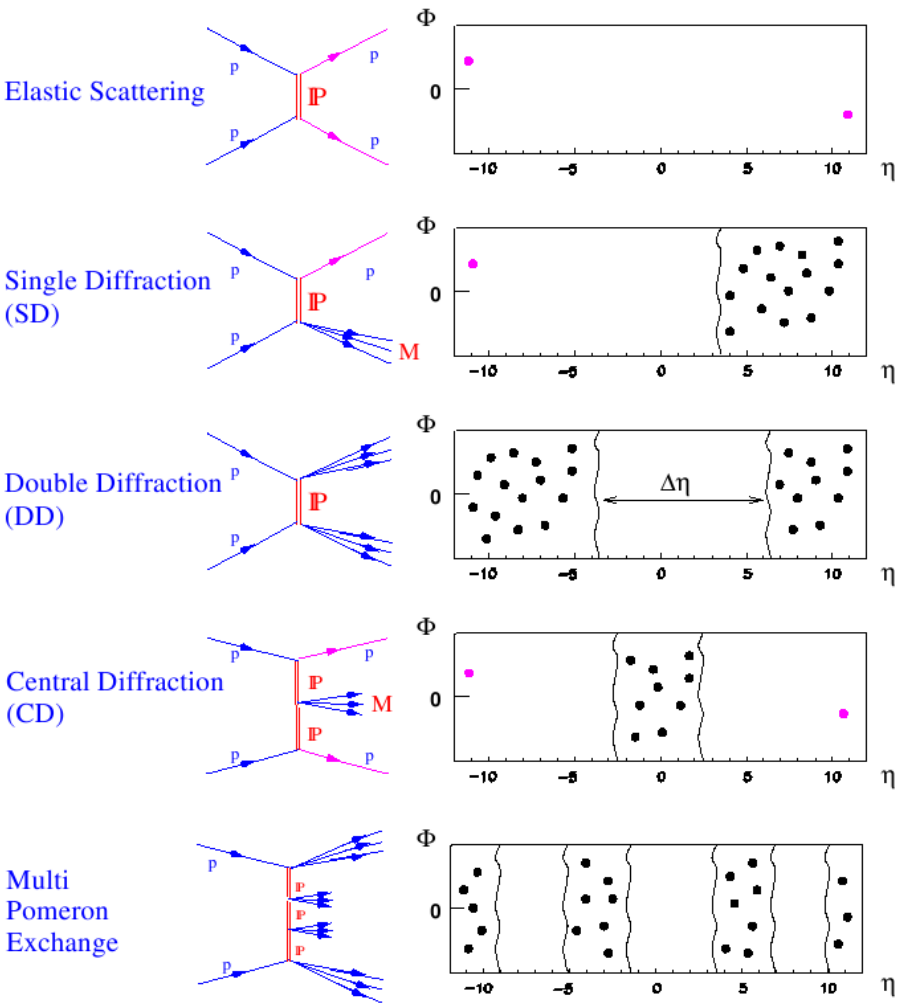


Figure 1.3: Graphical representation of the most common event types in $p-p$ collisions. The leftmost pictures are graphical representations of the processes and in the middle typical angular and pseudorapidity distributions are shown.

Strip Chambers). T2 is a second telescope a little further from the impact point and made of GEM (Gas Electron Multipliers). These telescopes are used to study charged particles produced inelastically. RP (Roman Pot)s are placed along the exiting beams, at 147 m and 220 m far from the interaction point. These detectors are based on silicon devices designed *ad hoc* for TOTEM.

The pseudo-rapidity coverage of the TOTEM apparatus is shown in Fig. 1.5.

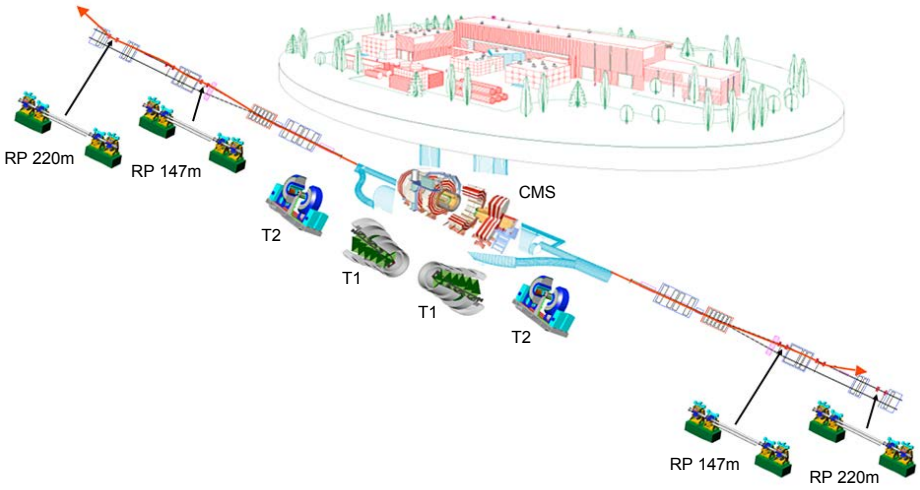


Figure 1.4: The LHC beam line, the TOTEM forward trackers T1 and T2 embedded in the CMS detector and the Roman Pots at 147 m (RP147) and 220 m (RP220).

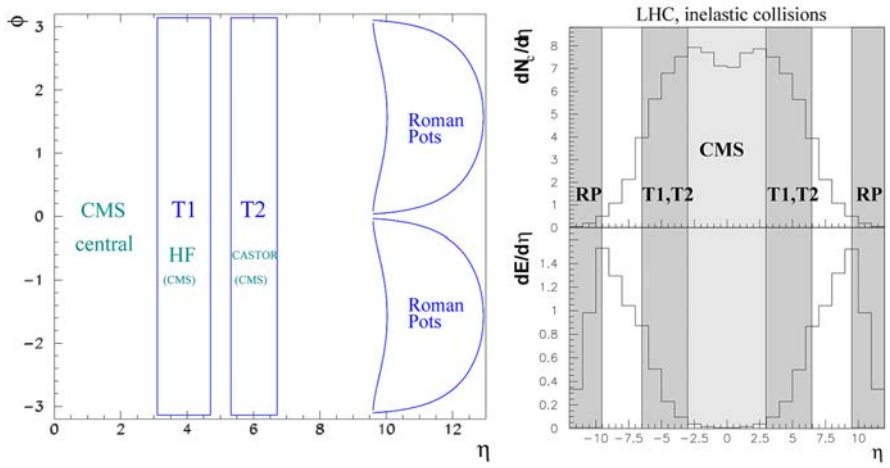


Figure 1.5: Left: Detector coverage in the pseudorapidity-azimuth plane. Right: pseudorapidity distribution of charged particle multiplicity and energy flow for generic inelastic collisions at $\sqrt{s} = 14TeV$.

1.2.1 Latest results

Using data taken during the year 2011 at the LHC energy of $\sqrt{s} = 7\,TeV$, TOTEM has measured the differential cross-section for proton-proton elastic scattering as a function of the four-momentum transfer t. Various measurements have been done under different beam and background conditions and, thanks to dedicated runs, $|t|$-values down to

$5 \times 10^{-3}\, GeV^2$ were reached. Thanks to these measurements, it was possible to extrapolate the value of the elastic cross-section to $t = 0$ with a non-visible region of only 9%.

The elastic cross-section has been determined to be $(25.4 \pm 1.1)\, mb$ and, using the luminosity probed by CMS in a first approximation, the total pp cross-section was indirectly estimated to be $(98.6 \pm 2.2)\, mb$ [2].

Moreover, during the same data taking, the proton-proton inelastic scattering cross-section was determined. Combined data from T1 and T2 allowed the measurement of the cross-section for inelastic events with at least one particle with a pseudo-rapidity $|\eta| \leq 6.5$ in the final state. This cross-section includes more than 99% of the non-diffractive and diffractive events with diffractive masses larger than $\sim 3.4\, GeV$.

On the base of these measurements, the total inelastic cross-section was deduced to be $\sigma_{inel} = (73.7 \pm 3.4)\, mb$, compatible with the previous indirect TOTEM measurements and with the direct measurement by the other LHC collaborations [1].

1.3 General overview of the TOTEM detector system

The TOTEM detectors are tracking devices that can be grouped in two families: gas and silicon detectors. All detectors are placed on both sides (arms) of the Impact Point.

T1 and T2 are dedicated to the measurement of the inelastic rate and are positioned to detect particles from almost all interactions.

T1 is made of 5 planes, each consisting of 6 trapezoidal CSC (Cathode Strip Chambers), and is installed inside the CMS End Caps. It is $3\, m$ long and its closer edge is 7.5 far from the Impact Point.

The much smaller T2, instead, is made up of 20 half circular sectors of GEM detectors per arm and it is installed at a distance of $13.5\, m$ from the Impact Point.

These detectors have to:

- provide a fully inclusive trigger for minimum bias and diffractive events;
- make possible to reconstruct the primary vertex of an event to reject tracks not crossing the Impact Point;
- be perfectly left-right symmetric with respect to the Impact Point, in order to have a better control on the systematic uncertainties.

Elastic events are, on the other hand, selected by RP (Roman Pot)s, that are movable enclosures for silicon detectors expressly designed for TOTEM; they are capable of tracking protons a few millimeters far from the beam. As for T1 and T2, also RPs have to provide both triggering and tracking capabilities.

1.3.1 T1 telescope

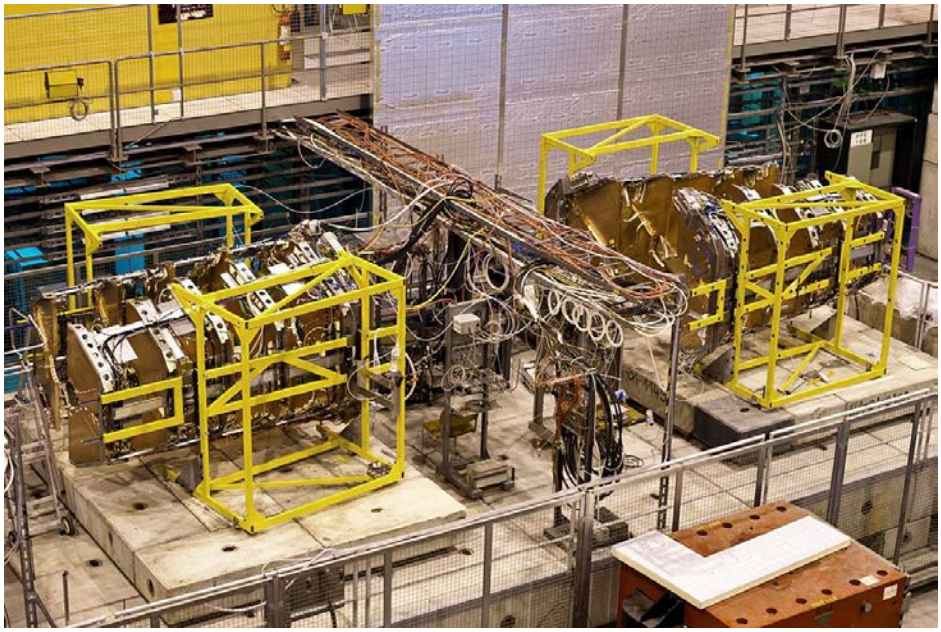


Figure 1.6: T1 telescope at the test beam facility. The five CSC planes are visible.

The closest telescope to the CMS impact point (7.5 m) is T1. This telescope covers a pseudorapidity range of $3.1 \leq |\eta| \leq 4.7$ and it is extremely important in the inelastic cross section estimate. The expected trigger rate for this detector is 1 kHz for a luminosity $\mathcal{L} = 10^{28} cm^{-2} s^{-1}$ and for each event ~ 40 charged particles are expected to be detected [7]. Moreover, T1 has to be able to provide a minimum bias trigger with a very high and well known efficiency and has to allow background (i.e. beam-beam or beam-beampipe) suppression after track reconstruction. These reasons led to the choice of CSC detectors, a widely used technology fast enough for TOTEM pourposes and lightweight enough to be positioned in front of the CMS forward calorimeters.

CSC chambers used in TOTEM are gas detectors with arrays of anode wires crossed with cathode strips on both sides. Anode wires are spaced 3 mm, while the cathode strips' pitch is 5 mm. Strips are $\pm 60^\circ$ tilted with respect to wires: on one side the angle is positive and on the other side it is negative. This design allows the detection of charged particles in three dimensions minimizing ghosts occurrence.

Each telescope (one per side) consists of five equally spaced CSC planes (Fig. 1.6). All planes are composed of six wire chambers, grouped in 2 halves, covering roughly one sixth of a circumference (60°) each. Moreover, planes are not perfectly aligned: this allows a better efficiency along the circular region and helps track reconstruction. The precision of the reconstructed position is of the order of 1 mm for the three coordinates [7], good

enough to reconstruct the primary collision vertex in the transverse plane within a few *mm* and to discriminate between beam-beam and beam-gas events.

Signals from the chambers are collected by a custom-designed ROC (Read-Out Card) module through AFEC (Anode Front-End Card)s and CFEC (Cathode Front-End Card)s. The ROC serializes and sends data to the DAQ (Data AcQuisition) system through an optical fiber, using a GOH (Gigabit Optical Hybrid) optical link developed by CMS. AFECs and CFECs are both based on VFAT (Very Forward Atlas and Totem) chips that provide the trigger capability (See section 1.4.1).

1.3.2 T2 telescope

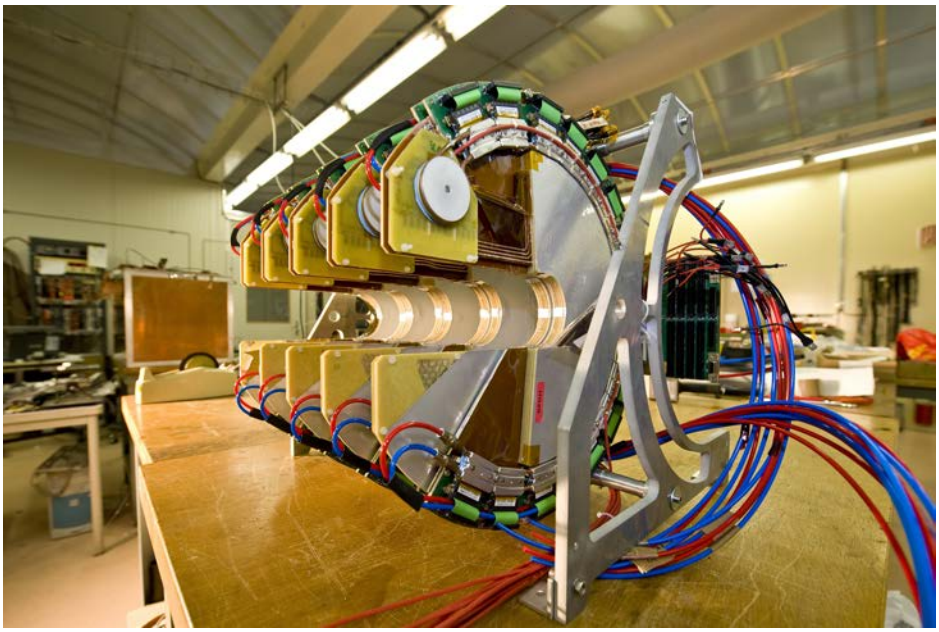


Figure 1.7: Picture of T2 during construction.

The T2 telescope is located at 13.5 m on both sides of impact point. It detects particles within the pseudorapidity range of $5.3 \leq |\eta| \leq 6.5$. As for T1, T2 has to provide a fully inclusive trigger for inelastic (mainly diffractive) events. Moreover, even if T2 is further from the interaction point than T1, it has to allow track reconstruction with almost the same rejection power of T1 to discriminate background.

These resolution requirements ($\sim 110\,\mu m$ and $\sim 1^\circ$ on the radial and azimuthal coordinates), together with the high rate capabilities, drove the choice on GEM detectors. These detectors were proposed for the first time in 1997 by Fabio Sauli [8]; thanks to their relatively easy and cheap design and to their diffusion in high energy physics, GEM can be considered a mature technology for the LHC environment.

The idea behind the GEM is to multiply electrons produced by ionizing particles inside small holes, where an high electrical potential is applied ($\sim 3\,kV/cm$). This is achieved using tiny resistive foils (the thickness of T2 GEM is 50 μm) with metal cladding ($\sim 5\,\mu m$) on both sides. This sandwich is etched to create holes from one side to the other (Fig: 1.8).

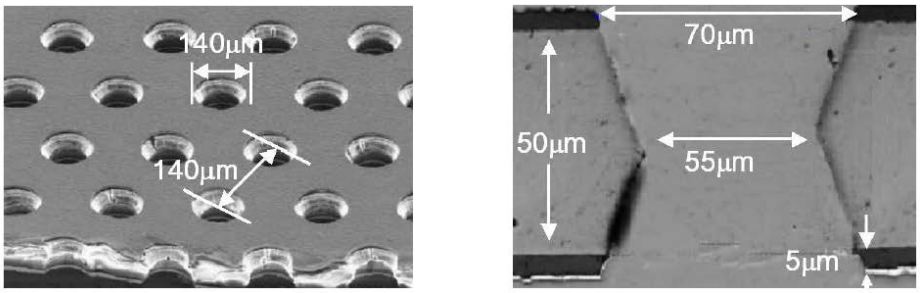


Figure 1.8: In T2's GEM, holes are conical and their diameter goes from 55 μm to 70 μm.

The hole dimensions are important for the quality of the electrical field inside them, avoiding the use of too high potentials between the two metal claddings. Usually, GEM foils can be cascaded: T2 has been made with the triple GEM configuration, based on the COMPASS detector [9] design.

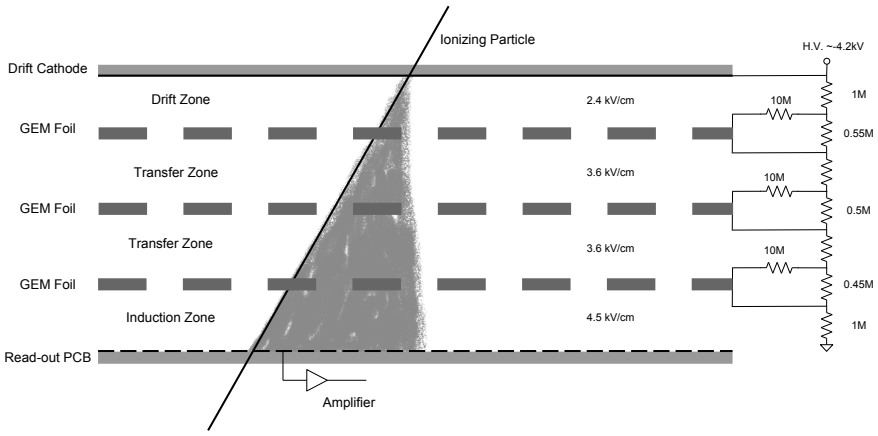


Figure 1.9: Schematic view of the triple GEM detector used in TOTEM experiment. Electric fields inside T2 chambers are provided by a resistive divider, useful to maintain the right voltage proportions between all the electrodes during power on/off operations. Each foil is also connected with a series resistor to the divider to limit the current in case of discharge.

The structure of this detector is schematized in Fig. 1.9. Charged particles ionize gas

molecules in the *drift zone*; ionization electrons drift towards the GEM foil stack where they are multiplied (*multiplication and transfer zones*) and finally they reach the *induction zone* where an electrical signal is induced on the readout foil. The main advantage of this design is that a large gain can be achieved without using extremely high voltages on the single GEM foil, decreasing the probability to have discharges inside the detector.

Moreover, the charge collection process, on the read-out PCB, is totally decoupled by the multiplication process, near GEM foils, simplifying the design the read-out pads.

Without GEM foils it would be impossible to design and build detectors with the geometry and the low density of T2. Indeed, GEM foils allow to build high space and time resolution detector with huge sensitive areas that are, at the same time, relatively cheap and easy to build.

The T2 detector uses this triple GEM design to achieve a gain of about 8000. The gas mixture used is Ar(70%) and CO_2(30%) and the applied voltages are shown in Fig. 1.9 [10].

The front-end chip is the VFAT, as for all other TOTEM detectors; furthermore, the readout and control systems are the same for all the experiment (See section 1.4.1).

1.3.3 Roman Pots

RP (Roman Pot) usually names movable box-shaped detectors used for detecting particles very close to the beam. They were used for the first time at the ISR [11] in early 70's. Indeed, the name "Roman" comes from the group of scientist from Rome that developed their main principles. These detectors are placed inside a secondary vacuum vessels (where the primary one is that of the beam pipe), called "Pots" because of their shape; the vessel is moved into the primary vacuum of the machine through vacuum bellows: the detectors are physically separated from the beam to prevent an uncontrolled out-gassing from the detector's materials. The challenging constraints of the LHC environment, such as the high luminosity and the UHV (Ultra High Vacuum), led to a massive improvement over the ISR prototypes.

TOTEM RPs are grouped in *unit*s. Each *unit* is made of three RPs, one approaching the beam from above, one from below and one from the side. Only one horizontal RP is needed because the LHC magnetic field deviates protons according to their momenta, and basic conservation rules imply that scattered protons can only have lower momentum with respect to the original one of the beam [3]. Pairs of these *unit*s, called *station*s, are placed at 149.6 m and 217.3 m far from the interaction point. The TOTEM's RP system is symmetric with respect to the interaction point allowing to tag surviving protons in elastic, single and double diffractive events.

A single RP is equipped with a stack of 10 tracking planes. These planes are silicon

[3]Considering the LHC lattice, to detect protons which have lost momentum, horizontal RPs are positioned on the external side of the ring.

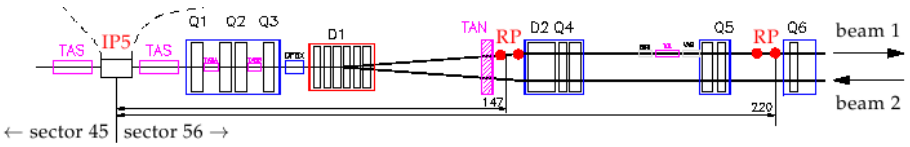


Figure 1.10: Position of RP detectors with respect to the impact point.

microstrip detectors and, in order to maximize the acceptance, Roman Pot systems have to detect particles as close as possible to the beam. A specific research project has been done to reduce the non active zone at the edge facing the beam.

The working principle of a silicon microstrip detector is that inside the silicon, when a depletion region[4] has been created by an applied electric field, an ionizing particle crossing it releases energy generating electron-hole pairs. Holes are collected by p^+ strips inducing a signal in the readout strips (Fig. 1.11). The sensitivity of these detector depends on the biasing voltage. Indeed, the width of the depletion layer is proportional to the square root of the biasing voltage and also the efficiency of strips at collecting holes is strongly correlated with the biasing voltage.

TOTEM Roman Pots are equipped with 300 μm thick silicon planes, with a strip pitch of 66 μm and each plane has 512 parallel strips.

In fact, silicon devices are cutted out from big silicon disks (wafers) and this procedure affects the behavior of the devices on the edge. The most common technique to cope with the distortion of the electric field in the vicinity of the cut edge is called *Voltage Terminating Structure* and it consists of a sequence of floating guardrings surrounding the sensitive part of the device. However this technique does not permit an insensible zone less than 1 mm wide. A new design, called *Current Terminating Structure*, has been developed to reach full sensitivity within $\sim 50\ \mu m$ from the edge [12].

The planes inside the RPs have been arranged in such a way that strips are oriented at an angle of $+45°$ and $-45°$ with respect to the edge facing the beam. These two orientations are called u and v projections. Planes are coupled back to back. A picture of the planes inside each RP is shown in Fig. 1.12.

The advantage of this arrangement is that u and v planes can be analyzed independently: a particle detected by all the planes will have aligned hits in both orientations. On the other hand, a big disadvantage is that it is not possible to reconstruct particle tracks in high multiplicity events because of wrongly reconstructed tracks, named *ghosts* (Fig. 1.13).

The VFAT is used as front-end chip and the rest of the DAQ chain is the same of the other detectors.

[4] A region (or layer) is said to be depleted when there are no free charges inside it.

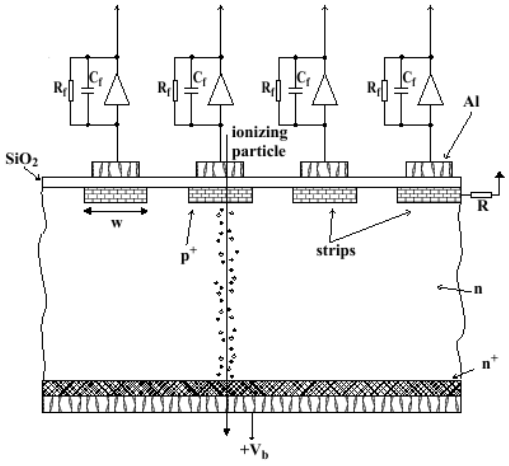


Figure 1.11: Cross section of a strip detector. Ionizing particles generate electron-hole pairs that drift to the strips thanks to the applied bias voltage. Strips are AC coupled to the readout electronics by the thin insulating silicon dioxide layer.

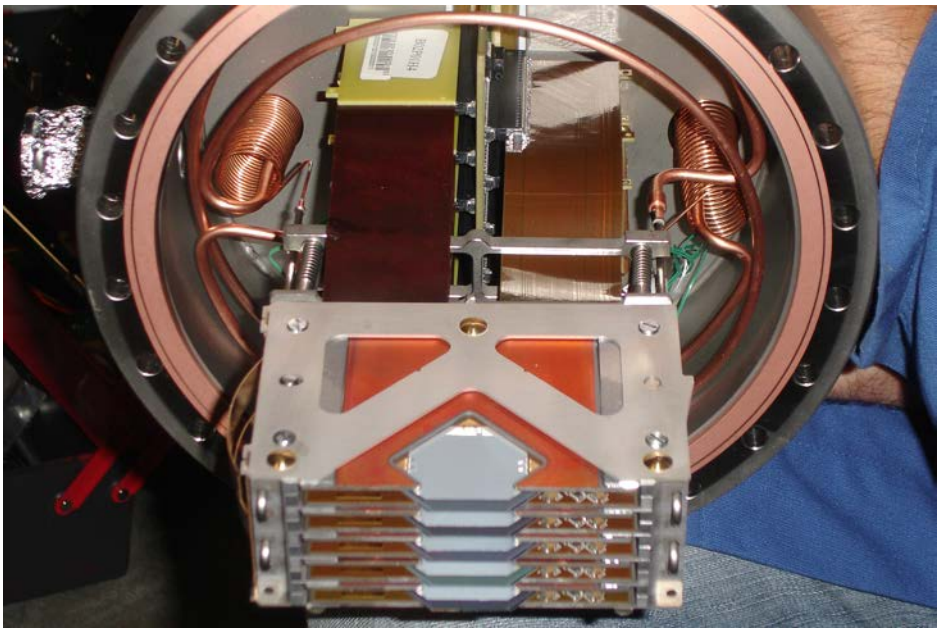


Figure 1.12: Silicon detectors inside each Roman Pot.

1.4 TOTEM Electronics System

As described in the previous sections, TOTEM has three separate and distinct detectors that use completely different technologies. Despite this, a big effort has been done by the

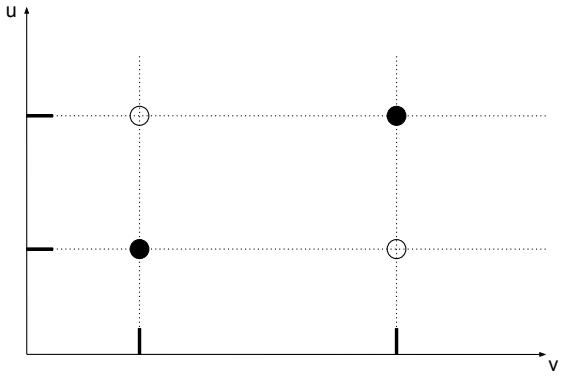


Figure 1.13: When two (or more) hits have been recorded for each projections, it is not possible to uniquely reconstruct the tracks. Indeed, there is no way to choose which pair of circles (empty or full) corresponds to the correct pair of tracks.

collaboration to have a common electronics architecture. A milestone in this direction has been the design of a common front-end chip, the VFAT, capable of providing common data format and common control and readout chains.

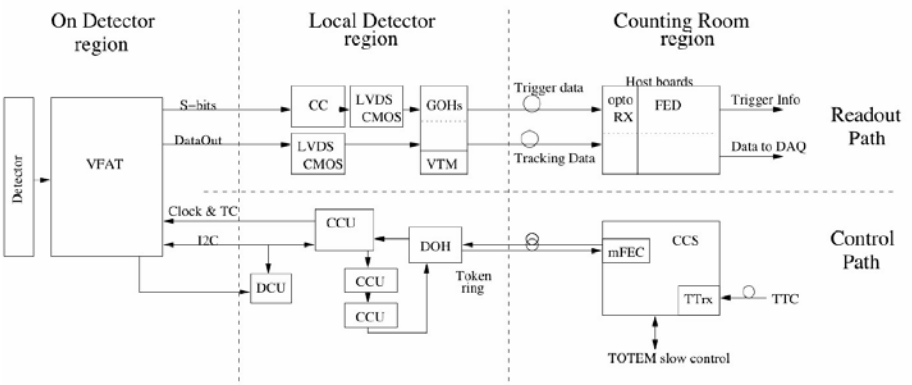


Figure 1.14: Block diagram of the TOTEM electronics system [7].

The TTC (Timing, Trigger and Control) signals provide the necessary reference LHC machine clock, TOTEM standalone trigger and control commands to the VFAT chips through the *control* path, as shown in Fig. 1.14. The configuration of the VFATs is done using a low speed protocol (I2C) to encode commands broadcasted by the control system (FEC).

Furthermore, the data readout chain ensures data to be read and stored. Its main component is the *TOTFed* hostboard that allows readout PCs to access data sent by the front-end electronics. The VFAT electronics transmit *trigger* primitives and *tracking* data.

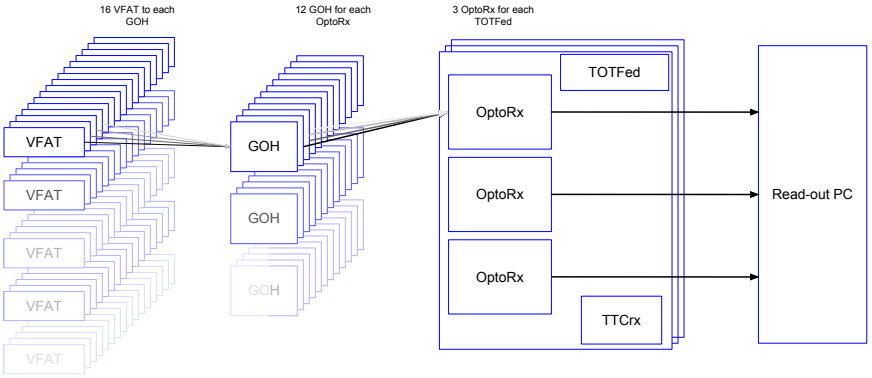


Figure 1.15: Schema of the TOTEM data readout chain.

The *trigger* primitives are processed as fast as possible by the trigger electronics to generate the TOTEM LV1A (LeVel One trigger Accept) trigger signal. Upon LV1A assertion, tracking data are collected and stored by the DAQ system. A deep knowledge of the data acquisition system was required for this thesis. The next sections will be focused on the description of the overall structure of the data acquisition system and of its main components.

1.4.1 VFAT: a common front-end chip

TOTEM collaboration decided to develop a common front end chip to simplify the design of the detectors' front-end: all detectors will use identical control, trigger and readout systems. This chip, named VFAT [13], has to store detector hit maps and to provide fast regional information to aid the creation of a first level trigger.

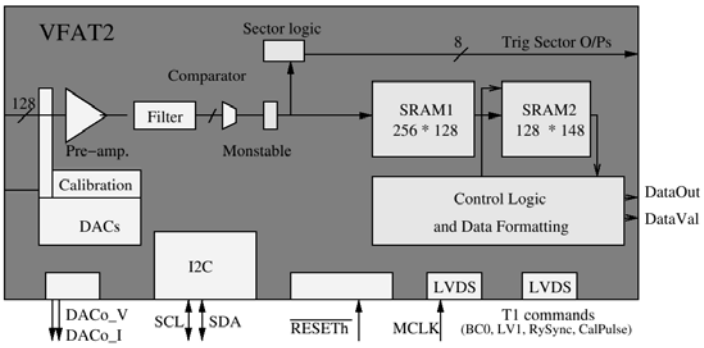


Figure 1.16: Block diagram of the VFAT chip. [13]

The VFAT chip is driven by the LHC clock frequency (40.08 *MHz*) and has 128 channels.

Each of these consists of a preamplifier followed by a shaper and an asynchronous comparator and with a programmable threshold. If a signal exceeds a given threshold, a monostable produces a logic 1 for n clock cycles, where n is programmable and can be in the range $1, \ldots, 8$. All monostable outputs are buffered to a circular SRAM and, at the same time, are used to set a trigger flag using a fast OR. These flags are collected and processed by the trigger system to generate the LV1A signal that is broadcasted back to the VFATs. Upon the receiving the LV1A signal, data are transferred to a second SRAM that can be read-out by a DAQ chain. Otherwise, data not corresponding to a LV1A are overwritten in the first circular SRAM (Static Random Acess Memory).

1010	BC <11:0>	
1100	EC <7:0>	Flags <3:0>
1110	ID <11:0>	
Channel Data <127:0>		
CRC 16 checksum <15:0>		

Table 1.1: Format of the VFAT data, which are serialized and streamed without any compression. [13]

Every detected hit is stored as a logic 1 in the so called "Channel Data" of the VFAT data frame. The first three words of this data frame are used to identify the VFAT producing the data and to synchronize events. Indeed, these words contain:

- BC (Bunch Crossing), a counter that is incremented on every clock cycle;
- EN (Event Number), a counter incremented on every LV1A;
- ChipID, an unique identification number.

Using this format, shown in Table 1.1, data are streamed to the counting room using optcal fibers; up to 16 VFAT can share the same fiber.

1.4.2 TOTFed

The *TOTFed* hostboard is the main part of the on-line Data Acquisition System of TOTEM.

All the devices work synchronously with the 40.08 *MHz* clock provided by the *TTCrx* QPLL; this ASIC (Application Specific Integrated Circuit) makes sure that the system is latched to the main LHC clock and allows also the decoding of commands from the TTC (Timing, Trigger and Control) system [14].

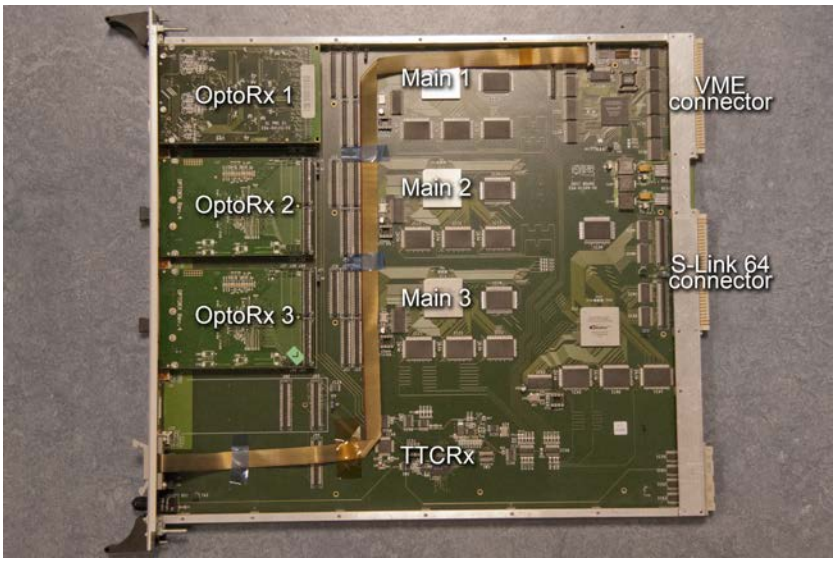


Figure 1.17: Picture of the *TOTFed* hostboard with three *OptoRx*s.

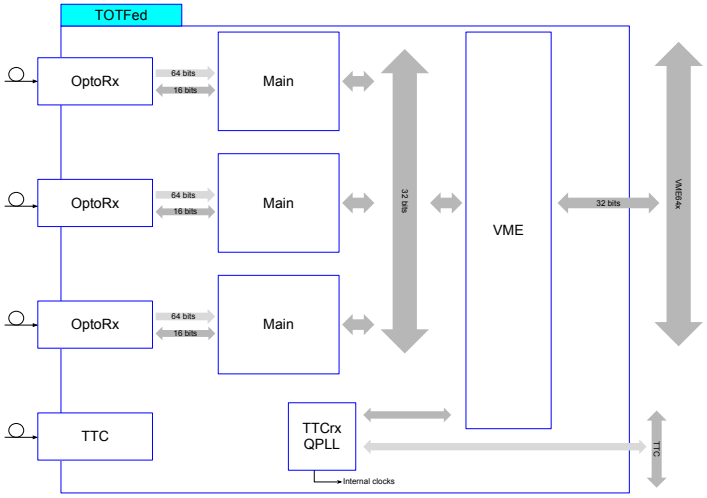


Figure 1.18: Schema of the *TOTFed* hostboard architecture.

Data from detectors are acquired through *Optical Receiver*s, named *OptoRx*s. Up to three of them are plugged onto the *TOTFed*. Each *OptoRx* is connected via two buses to an FPGA (Field Programmable Gate Array), named *Main*: one bus is 192 bits wide and the other 16. The first one is used for data, while the other is used to configure the devices and to read their status. Data read-out is allowed by a VME (VERSABUS Module Eurocard) bus, interfaced to a dedicated FPGA, that provides also I2C and JTag interfaces to control the *TTCrx* and to program devices plugged on the board. The architecture of the *TOTFed*

is shown in Fig. 1.18.

1.4.3 OptoRx

Figure 1.19: Picture of the *OptoRx* card.

Data from detector electronics are transmitted to the counting room (Fig. 1.14) via optical fibers that are connected to optical receivers plugged into the *TOTFed*. These receivers are called *OptoRx*; they are equipped with the most powerfull FPGA[5] used in the TOTEM electronics: *EP2SGX60EF1152C5*. Indeed, this FPGA, manufactured by Altera, belongs to *Stratix II GX* family and it is equipped with 12 high-speed serial transceivers used to receive data from the front-end electronics. This allows the connection of up to 12 optical fibers. Once data are received, the *OptoRx* has to synchronize data coming from different fibers and to buffer them. Furthermore, it is also possible to simulate a data flow to test and debug the DAQ chain that follows the *OptoRx*.

The *OptoRx* is connected to the *TOTFed* via a 64 bits bus to read buffered data and via a 16 bit bus to control and configure the card. Also TTC and TTS (Trigger Throttling System) signals are respectively received and sent by the *OptoRx* through dedicated lines. The first are used to synchronize acquired events, while TTS signals are used to suspend LV1A generation when internal buffer is approaching a full condition.

Moreover, the possibility is forseen to read data directly from the *OptoRx* connecting it directly to the CMS DAQ chain using an *S-link* bus.

[5]The name *OptoRx* is used for both the FPGA that equips the optical receiver and the receiver itself.

1.4.4 Firmware and data frame

The *OptoRx* architecture can be divided in two main blocks: the *Synchronization block* and the *Packet preparation block*, as schematized in Fig. 1.20.

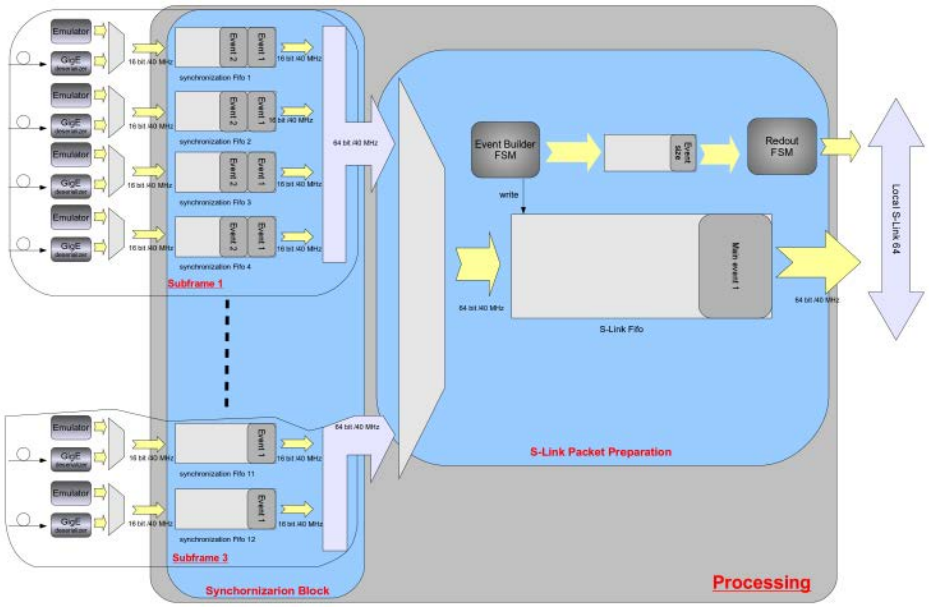


Figure 1.20: Schema of the *Optical Receiver* (*OptoRx*) architecture.

Data collected by up to twelve fibers are synchronized by the *Synchronization block*. After all acquired fibers have been synchronized, data has to be associated to the LV1A that triggered their acquisition, and independently collected by the TTCrx receiver.

In the *Packet preparation block*, data are organized in the frame shown in Fig. 1.4.3, compatible with the CMS data format. It is possible to highlight three main frames and, as for the VFAT, each data frame has an header, a trailer and a payload to store the data themselves:

- *OptoRx* frame, made of 64 bit words. Fig. 1.4.3 shows header and trailer in blue;
- 12 GOH frames, made of 16 bit words, highlighted in yellow;
- 192 VFAT frames, made of a single bitstream, colored in green.

OptoRx frame is organized in up to three subframes that contain four GOH frames. When less than twelve fibers are connected to the *OptoRx*, the data frame will be modified accordingly. If all fibers that belong to a subframe are not present, the subframe itself will not be present. Otherwise, if at least one fiber is connected, missing GOH frames will be filled in with zeros.

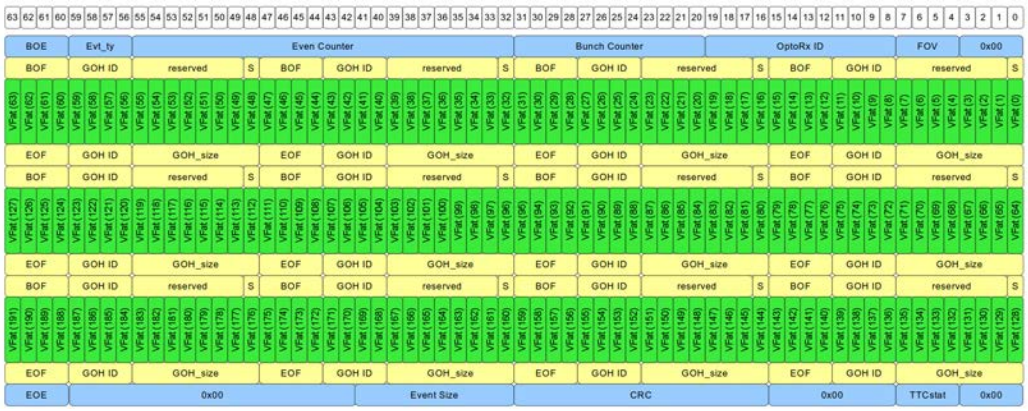


Figure 1.21: Data format used in TOTEM raw data.

In the RP system, four planes are grouped in each fiber in such a way that their VFATs are consecutive in the bitstream, as shown in Fig. 1.22. Furthermore, each *OptoRx* is connected to one *unit*. A RP detector consists of ten planes and all data acquired by a single detector are stored in forty consecutive VFAT frames. These frames are grouped in three GOH frames; hence, only nine consecutive GOH frames are used for each *OptoRx* of the TOTEM RP system, as shown in Fig. 1.23.

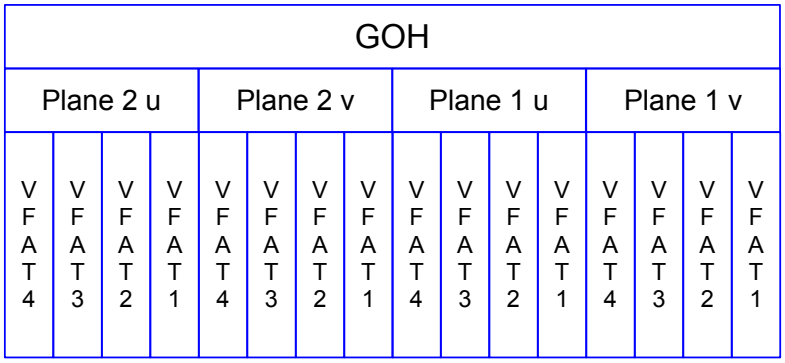


Figure 1.22: Schema of the arrangement of the VFAT frames in the TOTEM RP system.

OptoRx			
RP 2 GOH 1	RP 1 GOH 3	RP 1 GOH 2	RP 1 GOH 1
RP 3 GOH 2	RP 3 GOH 1	RP 2 GOH 3	RP 2 GOH 2
			RP 3 GOH 3

Figure 1.23: Schema of the arrangement of the GOH frames in the TOTEM RP system.

1.5 The off-line software

Once data have been acquired and stored, an off-line software takes care of reconstructing the physical processes occurred to the scattered protons at the IP. To accomplish this task, the off-line software [15] checks the data integrity and synchronization. Then, it transforms the data using calibration and alignment parameters that are computed using Monte Carlo simulations and preliminary analysis on real data. Indeed, an important part of the software implements Monte Carlo simulations.

Briefly, a Monte Carlo simulation consists of using random number generator to simulate a process that involves a large number of elements, i.e. particles, that are too complicated to be studied analytically. In the TOTEM off-line software they are used to simulate the transport of particles that exit the interaction point through the LHC lattice till their detection. Every step of the simulation is regulated using probability distributions

When a particle reaches the detector, also the interaction with the detector's materials and the response of the electronics have to be simulated. This process can be iterated millions of times to produce data similar to that expected from the real detector. Such data are useful to align and calibrate the detector and - by comparison to real data - to check if there are some unexpected phenomena that where not taken into account.

The TOTEM data offline software is both capable of generating these simulated events and of transforming simulated and real data into a common data format so that they can be analyzed in the same way. This common data format has been called *DIGI*: since the VFAT chips are digital, their output is a boolean hitmap of the detector.

The off-line software can be subdivided in some main blocks, as shown in Fig. 1.24. In this work the attention will be focused on the RP's reconstruction block.

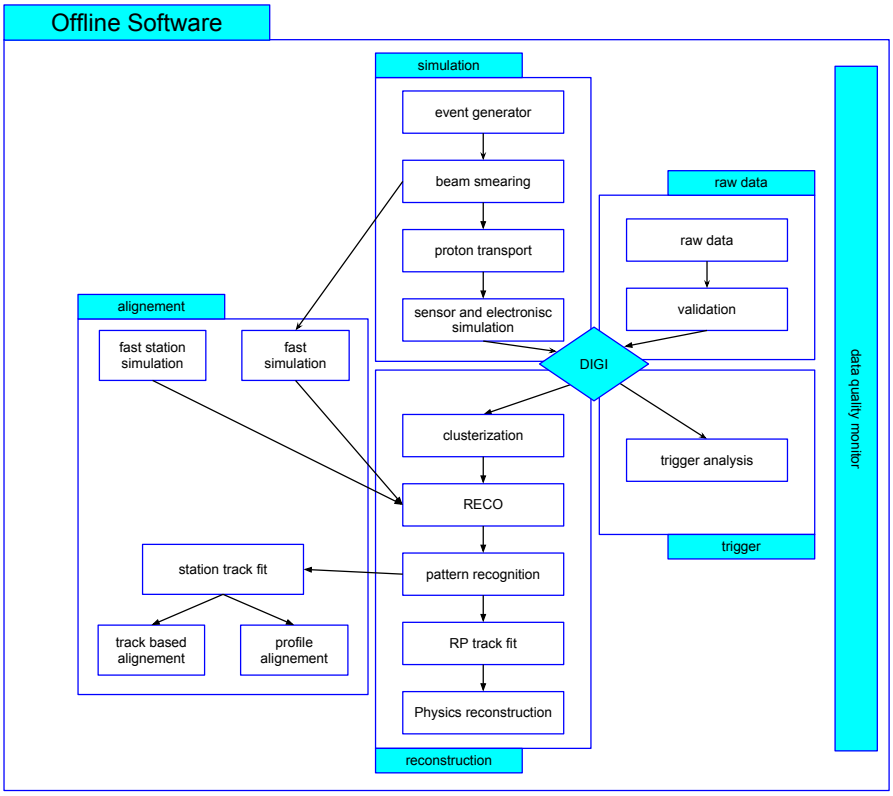


Figure 1.24: The structure of the RP simulation and reconstruction software [16].

The first step of the track reconstruction is clusterization. Indeed, when a charged particle goes through a detector, it could fire a group of neighboring strips, but these strips have to be viewed as a single block of information: the cluster. After clusterization, clusters are converted into positions in a defined reference system, i.e. their center of mass is expressed using the distance from the center of the detector. This is done by the *RECO* block in Fig. 1.24. In the next step, the *pattern recognition*, hits from each projection (see section 1.3.3) that doesn't belong to a track are rejected and, if possible, a track is reconstructed.

Two method have been proposed to recognize tracks: a quicker method and a more precise one. The first [17] is based on the fact that elastically scattered protons have longitudinal angles lower than 1 *mrad*, while tilted tracks are probably due to noise or interaction in the beam pipe and should be disregarded. In these conditions, a particle hits consecutive detector planes at about the same position: if the hit positions are histogrammed, the number of entries in the bins corresponding to a straight track will be higher with respect to bins filled with hits due to a non parallel track (see Fig. 1.25).

However, during an alignment procedure or for efficiency measurements it can be

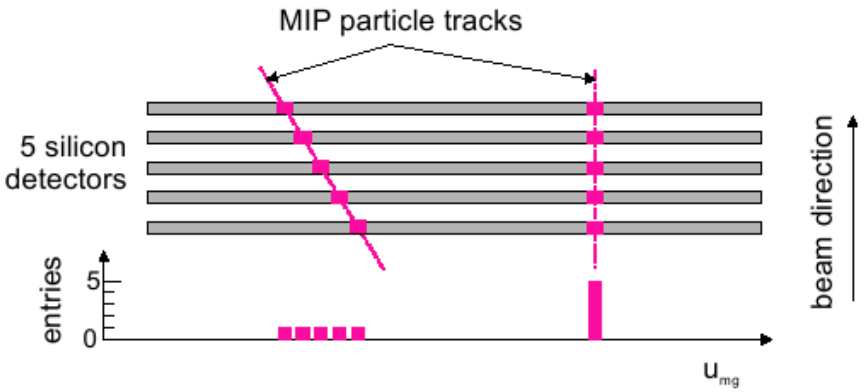


Figure 1.25: For a track parallel to the beam, and so perpendicular to the detector, hits through the 5 silicon planes fall in the same bin of the positions' histogram.

important to consider also tracks that are not parallel to the beam. To select these tracks, a second algorithm, based on a Hough transform, has been implemented [16]. Both algorithms will be described in more detail in section 3.

If tracks have been found in both projections, they are merged in the *one-RP track fit* block.

Finally, a correlation between tracks found in different RP is used to classify events, i.e. an ideal elastic scattering event needs left and right scattering angles to be identical. This process needs to carefully consider the effect of all LHC magnets between the impact point and each RP that can be up to 220 m far.

Chapter 2

Cluster searching algorithm

When protons interact with a silicon plane in the Roman Pot system, because of charge-sharing effects, it is possible that more than one strip is fired. In this case, adjacent hits have to be considered as belonging to a single track. For this reason the very first step for any kind of track reconstruction is to search for clusters. The goal of the algorithm proposed here is to find clusters in the VFAT frames (see 1.4.1). However, the modularity and the generality of the algorithm make possible to easily adapt it to different data frames that contain several bitstreams with an hitmap.

The algorithm has been developed using the *C++* programming language, to test its capabilities and performance. In this way it is possible to use it in applications capable of working on real data using the framework implemented by the DAQ team to both simulate and read data in the TOTEM format.

Eventually, the algorithm has been revised and implemented on an FPGA and it has been tested using simulation, RP (Roman Pot) test setup in the laboratory and using the real TOTEM Roman Pot system installed at the Impact Point.

2.1 The cluster searching Algorithm

Being the VFAT frame a bitstream, the algorithm consists of searching for consecutive "1"s in a bitstream. The problem we face is to efficiently extract the bitstream from the *OptoRx* frame (see 1.4.4 and Fig. 1.21). The idea is to use an highly specialized class for each step of the algorithm. In particular, a class will find clusters in a bitstream, an other one will manage a bytestream and finally, the last class will work on streams of words of arbitrary size. Such an architecture will allow the algorithm to work not only with the TOTEM data frame, but with any bitstream holding an hit map.

2.1.1 *cluster*: the main brick

The first step is to provide a precise definition of cluster: a group of consecutive hits in an hitmap. In the VFAT stream, this is equivalent to a set of consecutive "1"s.

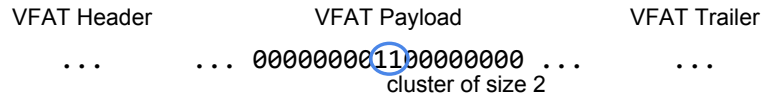


Figure 2.1: Example of a cluster of size 2 in the VFAT payload.

A cluster is fully defined by two properties: the position of its appearance and the number of consecutive "1"s (its size). This definition has been used to implement a class, *cluster*, which stores the position and the size of the cluster and has a method to increment its size.

2.1.2 *bitStat*: clusters in a bitstream

Using the definition of *cluster*, a class called *bitStat* has been designed to create and manage them. This class creates a collection of *cluster*s and has methods to access this collection. It is possible to assign an identifier to correlate each *bitStat* object with a bitstream, or a VFAT in the TOTEM case, where the clusters have been found. To fill the collection of *cluster*s the class has two methods:

- *FoundZero*: enables the creation of a new *cluster*;
- *FoundOne*: if the creation is enabled, creates a new *cluster* and stores it in the collection, else it increments the size of the last *cluster* in the collection.

These methods will be called according to the value found in each bit read from a bitstream. In addition this class has methods to read, print and clear the collection of *cluster*s.

It is important to stress that *FoundOne* and *FoundZero* have the same interface: they return /emphvoid and take no parameters; hence, it is possible to create a pointer to both methods using the same function pointer type. This property will be used in the next proposed class, *byteStat*.

2.1.3 *byteStat*: from byte to clusters using a LUT

The purpose of the *byteStat* class is to manage 8 *bitStat*s and call their appropriate *FoundOne* or *FoundZero* methods according to the bit pattern in a given byte.

The simplest way to do this task is to read each bit of the byte and call the corresponding *bitStat*'s method. However, this approach requires 8 bit shifts for each read byte: a more efficient way is to create a Look Up Table (LUT).

Thanks to the fact that the pointer to both *FoundOne* and *FoundZero* has the same type, it is possible to create a collection of these pointers. Thus, in the LUT each byte pattern corresponds to the address of a list of the above mentioned pointers. For each byte the list of pointers corresponding to the bit pattern is retrieved from the LUT and applied to the *bitStat*s managed by the class. This approach is faster because all the 8 bits are computed at the same time, without using any bit shift.

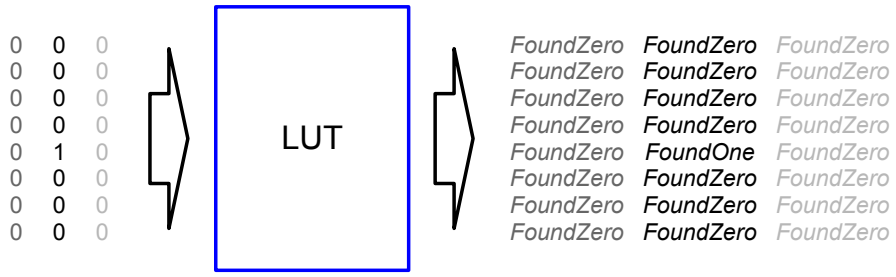


Figure 2.2: Working principle of the Look Up Table.

As an example, if the input byte is 1000 0001 (129), *byteStat* will point in LUT location 129 and find a collection of pointers. This collection will contain a pointer to *FoundOne*, followed by 6 *FoundZero* and one *FoundOne*.

The LUT has been implemented as a *singleton*[1]; this ensures that all *byteStat* objects access the same LUT which is initialized only once.

2.1.4 *wordStat*: clusters in a buffer of words of arbitrary length

As said before, the algorithm has to be able to find clusters in an arbitrary buffer of words and not be linked to any particular architecture. To achieve this goal a class was designed which, given a buffer of a certain type and a stream of words of a different type in a fixed position in the buffer, searches for clusters using the right number of *byteStat*s, according to the word size. Every word of the stream is in the same position of each word of the buffer.

[1] A *singleton* is a creational pattern that guarantees the existence of one and only one instance of a class and provides an access point to it (pointer or reference) [18].

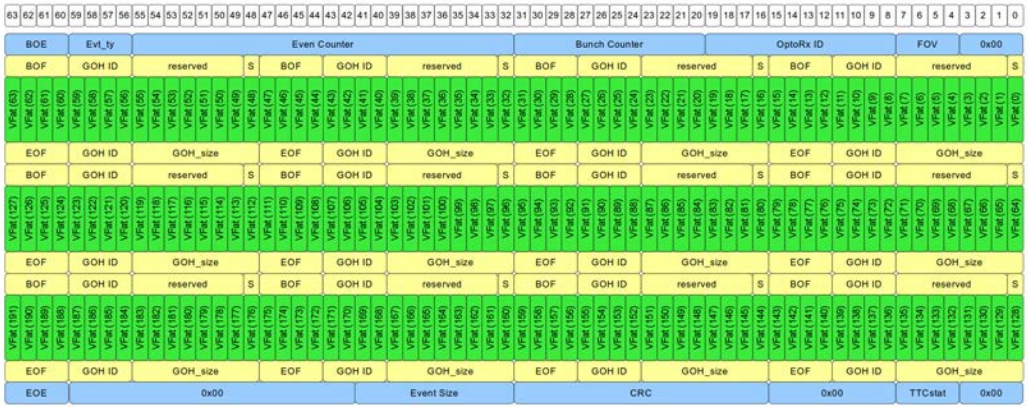


Figure 2.3: Data format used in TOTEM raw data (See section 1.4.3).

As an example, it is possible to consider the data frame shown in Fig. 2.3 (See section 1.4.3). The GOH's frame is a stream of 16 bit words stored in the *OptoRx* frame, a buffer of 64 bit words. It is worth noting that every GOH word has the same position in the *OptoRx* word; so, to jump from a GOH's word to its next, it is only necessary to know this position. Two *byteStat* objects can search for clusters in the GOH frame: it is necessary to design a class which creates 2 *byteStat*s and gives to each one the corresponding 8 bit word to analyze.

This class has been called *wordStat*. To achieve the proposed goal, *wordStat* is a template class with 2 parameters: the type of the buffer's words (*B-word*s) and the type of the words of the stream (*T-word*s). Moreover, *wordStat* has to be initialized also with the number of words to be read from the buffer and has methods to read, print and clear the collected clusters.

It is useful at this point to introduce two additional classes.

bytePointer, which allows to access single bytes of the given stream words (*T-word*s) and knows how to jump to the next corresponding byte in the buffer (*B-word*s).

word2byte is a class that takes a pointer to a *T-word* and the position of the stream in this word and returns the opportune *bytePointer*.

2.1.5 Usage in the TOTEM framework

A steering class has been designed to specialize the algorithm to the TOTEM data format. This class, *optoRxStat*, knows the data frame structure and initializes the algorithm. Its main task is to read an *OptoRx* frame from a buffer and create the appropriate *wordStat* objects, passing to them the corresponding pointers and the size of the buffer.

To read the *OptoRx* frame, *optoRxStat* uses an object from the DAQ library, *OptoRx-*

DataFrame, which has a method that, for each GOH frame (Fig. 1.4.3), returns a pointer to the beginning of the buffer which contains VFAT data. This pointer is used, together with the size of the buffer, to initialize the corresponding *wordStat*, managed by *optoRxStat*. After the initialization, each *wordStat* (one per GOH) does its analysis independently.

2.2 Cluster per plane filter

After the design of an algorithm to search clusters inside the TOTEM data frame, it is useful to implement a data filter to reduce disk usage and, most importantly, improving the data acquisition rate. On the base of the knowledge achieved by the off-line software developers (section 1.5) we can safely exclude from the analysis all planes with more then 4 clusters, or less then 1; in addition, if more than 2 planes per projection are excluded, the whole RP will be rejected. Excluding these parts of the data from the analysis will lead to performance improvements without penalties in the results.

For this reason, counting the number of clusters per plane could be a very powerful tool to select track candidates and to reduce data to be written on disk, improving DAQ efficiency. In some of the data taking runs the occurrence of empty frames is very high; as an example, if the event has been triggered by T1, the probability to have hits in Roman Pots is very low. Moreover, if a detector registered too many hits, it would be very likely rejected by the reconstruction algorithm, so recorded data will be useless.

However, by filtering information with an on-line analysis, rejected data will not be available for future investigation with improved techniques. Therefore, it is of great relevance to design a filter which rejects only data that contains useless information and nothing else.

This is the case of an algorithm which rejects *empty RP*s. An *empty RP* is a RP with at least 4 planes with no clusters for each projections. It should be noted that a single plane is useless for any reconstruction, so this kind of filter does not reject any data suitable for the reconstruction.

A more aggressive approach is to reject also RP detectors with a number of clusters above a certain threshold.

The performance of these filters will be studied in the chapter 4.

2.3 Implementation in the *OptoRx* firmware

The proposed algorithm can be very powerful to reduce data size at hardware level, rejecting useless events or applying data reduction. However, to fully exploit its capability, it needs to be implemented as close as possible to the front-end electronics. It has been chosen to implement the filter inside the *OptoRx*. Indeed, the FPGA of the *OptoRx* has enough resources to implement a cluster finding algorithm; moreover, this

card communicates with both VME and *S-link* bus, making it the perfect location for implementing selection algorithms.

The FPGA version of this algorithm has been slightly modified. In fact, inside an FPGA is possible to access a single bit; the use of a LUT to access 8 bits at the same time can be avoided. Hence, each VFAT bitstream is analyzed by an independent *bitStat* block. Then, sixteen *bitStat* blocks are managed by a *gohStat* block and, finally, twelve *gohStat* blocks are managed by one *optoRxStat* module that can be inserted in the *OptoRx* firmware.

All these blocks will be described in details in the following sections.

2.3.1 *bitStat*: clusters in a bitstream

The *bitStat* block corresponds to the hardware implementation of the *bitStat* class: it searches for cluster in a bitstream and stores them in an internal memory, i.e. a FIFO.

Moreover, to debug the implementation, a special effort has been done to read all the information stored in the FIFOs. For these reasons an interface has been designed to transfer all the information collected by the module to the VME bus. To avoid interferences with data streams, it has been chosen to read FIFOs via the 16 bit bus that connects the *OptoRx* to the *TOTFed* (see section 1.4.3). This approach will allow to have an independent read-out block for cluster analysis that can be added or removed from the *OptoRx* firmware. To synchronize FIFO's content with event's data, at the occurrence of every new event an event counter is stored in each FIFO.

The *bitStat* architecture is implemented with several components as shown in Fig. 2.4.

The inputs to the *bitStat* block are:

- *newEvent*: it goes high for one clock cycle when a new VFAT payload is beginning;
- *vfat*: it is the VFAT bitstream;
- *vfatPayload*: it is high when VFAT payload data is streaming;
- *evCounter*: it is an 8 bit signal used for data synchronization;
- *clock*: the main 40.08 *MHz TOTFed* clock that drives all previous signals;
- *clockFifo*: 80.16 *MHz* clock signal to read the FIFO;
- *rdAck*: read acknowledge signal.

The outputs are:

- *beginOfCluster*: it goes high for one clock cycle when a new cluster begins. It is synchronyzed with *clock*;
- *data*: the 16 bit output from the FIFO. It is synchronyzed with *clockFifo*.

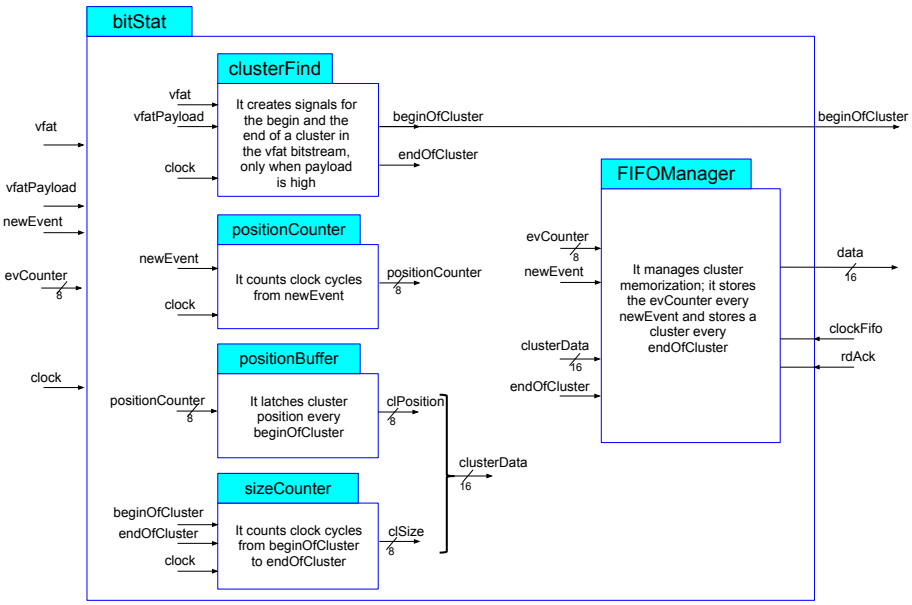


Figure 2.4: Block diagram of *bitStat* module.

A *clusterFind* block analyzes *vfat* when *vfatPayload* is high and assert *beginOfCluster* and *endOfCluster*. These signals are asserted respectively when a cluster begins and when it ends. Their duration is one clock cycle. To generate *beginOfCluster* and *endOfCluster* signals, this module uses a Mealy FSM (Finite State Machine) with two states:

- *IDLE*: when no cluster have been found in the input bitstream;
- *CLUSTER*: while a cluster is streaming.

The transitions are:

- from *IDLE* to *CLUSTER*: when both *vfat* and *vfatPayload* are high; on this transition *beginOfCluster* is set to high;
- from *CLUSTER* to *IDLE*: when at least one of *vfat* and *vfatPayload* is low; on this transition *endOfCluster* is set to high.

In all other cases, the state stays the same and both *beginOfCluster* and *endOfCluster* are set to low. The *newEvent* signal resets the FSM to the *IDLE* state.

The FSM transition chart is shown in Fig. 2.5.

A *positionCounter* block counts the number of clock cycles, starting from 0, every time a new VFAT payload begins (*newEvent* goes high for one clock cycle).

When *clusterFind* asserts a *beginOfCluster*, the value of the *positionCounter* is latched into the *positionBuffer* register. Meanwhile, *sizeCounter* starts counting from 1. When

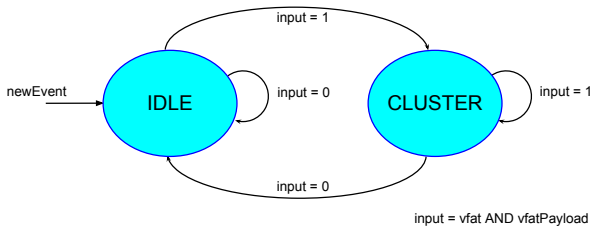


Figure 2.5: State transition chart of the FSM that generates *beginOfCluster* and *endOfCluster* signals.

endOfCluster goes high, *sizeCounter* outputs size informations that are merged in a 16 bit word with the cluster position stored in *positionBuffer*. This word is stored into the FIFO: the most significant 8 bits are used for the size of the cluster, while the remaining 8 bits for the position.

Moreover, every time a *newEvent* is asserted, *evCounter* is stored into the FIFO, marking the beginning of a new event.

A *FIFOManager* block handles the FIFO read and write procedures. During the writing procedures, it transfers the *evCounter* in the FIFO when *newEvent* is asserted and transfers *clusterData* when a *endOfCluster* is received. Read-out procedures are more complex, indeed some control word has to be added:

- an *header* with an identification number for the *bitStat*;
- a *fifoSize* with the number of word written into the FIFO;
- a *trailer* to end the read-out process.

It is worth noting that the cluster size can be a positive value different from 0 and less than 128 (the VFAT payload is 128 bit long) and cluster position can be from 0 to 127. All words beginning with *0xF* or *0x0* can be used as control words:

- *0xFF* has been chosen as the most significant part of the *header*, the remaining 8 bits are used to identify the *bitStat*;
- *0xFE* has been chosen as the most significant part of the *fifoSize*, the remaining 8 bits are used to say how many words are stored in the FIFO;
- *0xFFFF* is the *trailer*.

Moreover, as mentioned before, the *evCounter* is an 8 bit word. The fact that FIFO words have 16 bits allows to set the 8 most significant bits to 0. In this way, it is possible to discriminate between the *evCounter* and the cluster information. At the same time, it will be possible to interpret *evCounter* either as a 8 bit or a 16 bit word without issues.

To handle this read-out procedure a four states Mealy FSM has been implemented; its states are:

- *HEADER*: *data* output is the *header*;
- *SIZE*: *data* output is the *fifoSize*;
- *PAYLOAD*: *data* output is a word popped from the FIFO;
- *TRAILER*: *data* output is the *trailer*.

The transitions are:

- from *HEADER* to *SIZE*: when the *rdAck* signal is high; on the transition the *header* with the *bitStat* identification is set to the *data* output;
- from *SIZE* to *PAYLOAD*: when the *rdAck* signal is high and the FIFO is not empty; on the transition the *fifoSize* is set to the *data* output;
- from *SIZE* to *TRAILER*: when the *rdAck* signal is high and the FIFO is empty; on the transition the *trailer* is set to the *data* output;
- from *PAYLOAD* to *PAYLOAD*: when the *rdAck* signal is high and the FIFO is not empty; on the transition a word is popped from the FIFO and set to the *data* output;
- from *PAYLOAD* to *TRAILER*: when the *rdAck* signal is high and the FIFO is empty; on the transition the *trailer* is set to the *data* output;
- from *TRAILER* to *HEADER*: when the *rdAck* signal is high; on the transition the *header* with the *bitStat* identification is set to the *data* output.

When the *rdAck* signal is low, the state remains the same and, in case the state is *PAYLOAD*, no word is popped from the FIFO. The *newEvent* signal resets the FSM to the *HEADER* state. It should be noted that if the FIFO is empty, i.e. read-out is done without acquiring any event, the output data frame will contain only header, *fifoSize* (*0xFE00*) and trailer.

The FSM transition chart is shown in Fig. 2.6.

2.3.2 *gohStat*: cluster analysis for a single fiber

OptoRx receives VFAT data from up to twelve optical fibers and each fiber has sixteen independent data sources (see section 1.4.3). It is usefull to create a block that analyzes VFATs' data carried by one fibers: if a fiber is not present or not enabled, the correspondent block can be easily disabled.

The architecture of this block is shown in Fig. 2.7.

The inputs to the *gohStat* block are:

- *vfat*: it holds 16 VFAT bitstream;

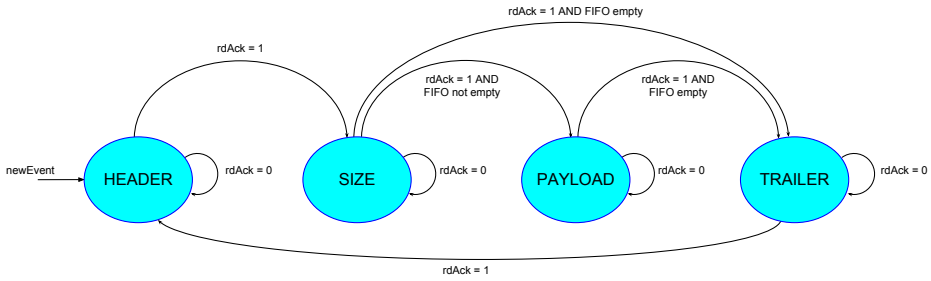


Figure 2.6: State transition chart of the FSM that manages reading procedure of FIFOs.

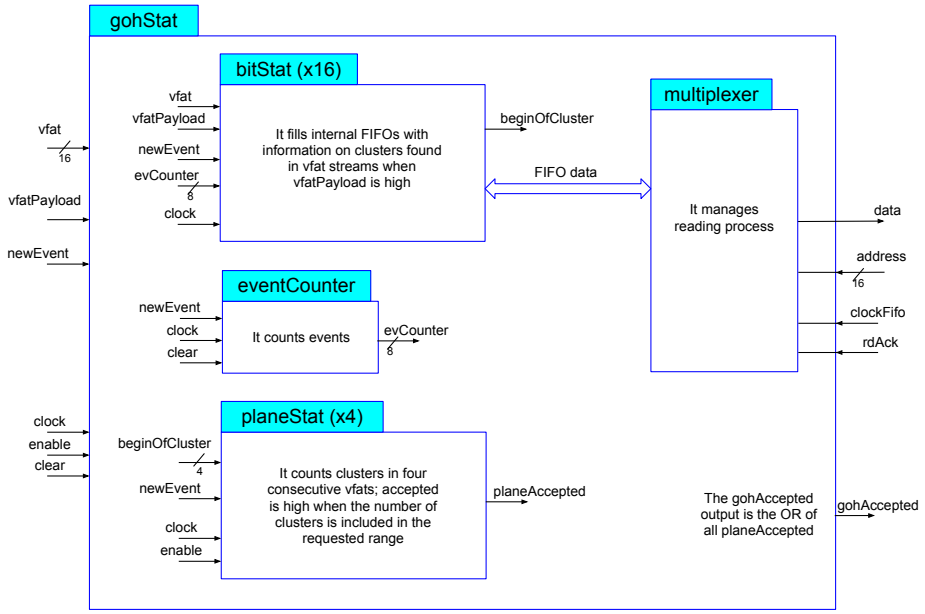


Figure 2.7: Block diagram of GohStat module.

- *vfatPayload*: it is high when VFAT payload data is streaming;
- *newEvent*: it goes high for one clock cycle when a new VFAT payload is beginning;
- *enable*: it enables *planeStat* block;
- *clock*: the main 40.08 *MHz TOTFed* clock that drives all previous signals;
- *clockFifo*: 80.16 *MHz* clock signal to read the FIFO;
- *rdAck*: read acknowledge signal;
- *address*: it is the address used by the *multiplexer* to read data from a *bitStat*.

The outputs are:

- *gohAccepted*: it goes high when at least one plane has a number of cluster compatible with the chosen range. It is synchronyzed with *clock*;
- *data*: the 16 bit signal controlled by the read-out FSM. It is synchronyzed with *clockFifo*.

Each VFAT bitstream in the fiber is sent to a *bitStat* block, together with the *vfatPayload* and *newEvent* signals.

To synchronize collected data, the *eventCounter* block counts the number of *newEvent* signals received since the reset of the *gohStat*. The output of this block, *evCounter*, is used by the *bitStat* block (see section 2.3.1).

The pourpose of the algorithm is to count the number of clusters per plane. Hence, data from VFATs that belong to the same plane are analyzed by dedicated blocks, called *planeStat*. Each plane has four VFATs and these are consecutive bitstreams inside the GOH data frame (see section 1.4.4). Each fiber can transmit data from up to 4 planes. Indeed, four *planeStat* blocks count clusters found in planes and assert an *planeAccepted* signal if the number of cluster per plane is included in a given interval.

The architecture of *planeStat* block is shown in Fig. 2.8.

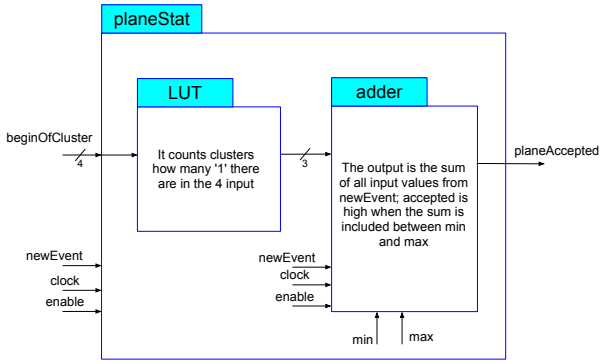


Figure 2.8: Block diagram of *planeStat* module.

The inputs to the *planeStat* block are:

- *beginOfCluster*: *beginOfCluster* collected from four VFATs of the same plane;
- *newEvent*: it is high for one clock cycle when a new VFAT payload begins;
- *enable*: it enables *planeStat* block;
- *clock*: the main 40.08 *MHz TOTFed* clock that drives all previous signals;

Moreover, this block needs two parameters that define the acceptance interval for the number of cluster per plane: *min* and *max*.

Its outputs is:

- *planeAccepted*: it is high while the number of cluster found in the plane is greater or equal than *min* and less or equal than *max*.

In more details, the *planeStat* block counts how many clusters have been found in the four inputs using a *LUT*. Every clock cycle this value is added to the previous one and it is compared with the given parameters. The *planeAccepted* is asserted accordingly to this last operation.

The outputs of these four *planeStat* blocks are merged in one *gohAccepted* signal. In the proposed implementation this signal is the logic OR of the *planeAccepted* signals, but more advanced criteria can be implemented.

The sixteen *bitStat* outputs can not be read using separate outputs; to handle the read-out procedure using a single 16 bit output, *data*, a *multiplexer* block has been designed. When an address is written on the *address* input, the *multiplexer* internal logic connects the *rdAck* input of the *gohStat* to the *rdAck* input of the corresponding *bitStat* block. Then, also the *data* output of the appropriate *bitStat* block is connected to the *data* output of the *gohStat*.

2.3.3 *optoRxStat*: cluster analysis for all the *OptoRx*

optoRxStat is the top level block that, once in the *OptoRx* firmware, manages the cluster analysis for all the the *OptoRx*.

The architecture of the *optoRxStat* block is shown in Fig. 2.9.

The inputs to the *optoRxStat* block are:

- *vfat*: it holds 12 fibers streams;
- *dataValid*: it holds 12 data valid signal: one for each fiber;
- *enable*: it enables *planeStat* block;
- *clear*: it resets *planeStat* block;
- *clock*: the main 40.08 *MHz TOTFed* clock that drives all previous signals;
- *clockFifo*: 80.16 *MHz* clock signal to read the FIFO;
- *rdAck*: read acknowledge signal;
- *address*: it is the address used by the *multiplexer* to read data.

The output is:

- *data*: the 16 bit signal controlled by the read-out FSM. It is synchronyzed with *clockFifo*.

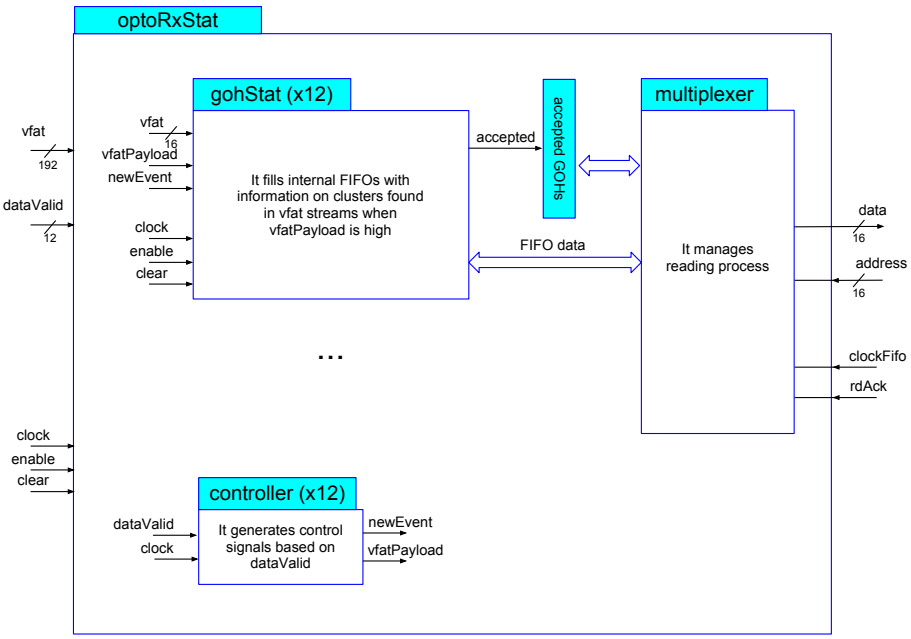


Figure 2.9: Block diagram of OptoRxStat module.

The *optoRxStat* block groups 12 *gohStat* objects, each one with a 16 bit output to read the internal FIFO. The access to all these outputs would require 12 memory addresses on the local bus of the *TOTFed*. To limit the memory space usage, *optoRxStat* has been designed to use an indirect addressing memory access using an internal *multiplexer* that manages the reading process, as seen for the *gohStat* block. When an address is written in the *address* input, the requested data will be available in the *data* output.

The available addresses have been paged: the page with the most significant 8 bits set to *0x00* is used to address *gohStat* and *bitStat*. The remaining 8 bits of the address are divided into two 4 bit words: the most significant is used for the *gohStat* address (from *0x0* to *0xA*) and the other one for the *bitStat* address inside the *gohStat* (from *0x0* to *0xB*). As an example, to read the FIFO of the third *bitStat* of the second *gohStat* the address to use is: *0x0023*. It is worth to note that the first word read by the *bitStat* data output contains the *bitStat* identification, see section 2.3.1. This will allow a cross check of the reading procedure.

An internal 16 bit register, called *accepted GOHs*, is used to merge all *gohAccepted* signals to allow their simultaneous read-out. Each bit of this register will correspond to a *gohStat* and the most significant 4 bits (*gohStat*s are twelve) are set to 0. This register will be readable from the *data* output when the *address* is set to *0xACFF*.

Thanks to the indirect addressing schema used, *data* and *address* are using only two

registers connected to the local bus of the *TOTFed*.

Inside the *optoRxStat* block, twelve *gohStat* blocks are instantiated. These blocks need *newEvent* and *vfatPayload* signals that are generated by twelve independent Mealy FSMs. The states of the FSMs are based on the VFAT data frame, see section 1.4.1:

- *IDLE*: the VFAT data are not streaming;
- *HEADER*: the VFAT header is streaming;
- *PAYLOAD*: the VFAT payload is streaming;
- *TRAILER*: the VFAT trailer is streaming.

The transitions are:

- from *IDLE* to *HEADER*: when the *dv* signal is high; on the transition an internal counter is set to 0;
- from *HEADER* to *HEADER*: when the *dv* signal is high and the internal counter is different from 46; on the transition the internal counter is incremented;
- from *HEADER* to *PAYLOAD*: when the *dv* signal is high and the internal counter equals 46; on the transition the *vfatPayload* is set to high;
- from *PAYLOAD* to *PAYLOAD*: when the *dv* signal is high and the internal counter is different from 174; on the transition the internal counter is incremented;
- from *PAYLOAD* to *TRAILER*: when the *dv* signal is high and the internal counter equals 174; on the transition the *vfatPayload* is set to low;

When the *dv* signal is low, the state remains the same and the internal counter is not incremented. The *newEvent* signal resets the FSM to the *IDLE* state. It should be noted that the values of the counter that trigger the state transition are computed according to the VFAT data frame.

The FSM transition chart is shown in Fig. 2.10.

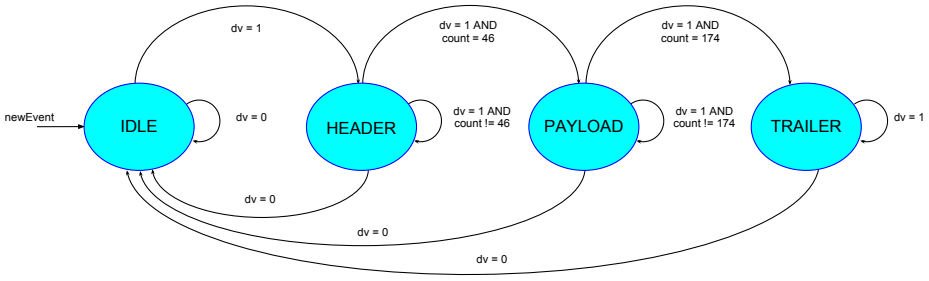


Figure 2.10: State transition chart of the FSM that generates *newEvent* and *vfatPayload* signals.

Chapter 3

Track recognition algorithms

When a charged particle interacts with the silicon planes of a Roman Pot detector it creates a signal that is collected by the front-end electronics and is stored by the DAQ system. Since the signal is subject to fluctuations, even for the same detector in the same conditions, the stochastic nature of this process makes the recognition of the trail of signals, called track, a complex task.

In this chapter two different approaches in recognizing tracks will be presented. The first is fast and can work on a stream of data, but it can be applied only to a particular typology of tracks (see section 3.2). The second proposal is based on a simplified Hough transform and allows the detection of all tracks, but its computational complexity is higher (see section 3.3).

The performance of these algorithms will be studied in chapter 4.

3.1 Tracks in Roman Pot detectors

In the TOTEM Roman Pot detectors, a track is a series of aligned hits in the silicon planes. The planes are grouped into two projections, named u and v, that have to be treated separately.

Thanks to the geometry of the experimental setup and to the LHC magnets configuration, tracks of elastically scattered protons have longitudinal angles lower than 1 μrad, while strongly tilted tracks are probably due to noise or close-by beam-pipe interactions and should be discarded. These good tracks will be called *elastic* tracks. It is possible to choose a coordinate system in which an *elastic* track in the RP is seen as a vertical line in a histogram. A possibility is to use the plane number, on both u and v projections, on the Y axes and the strip number per plane as X coordinate, as shown in Fig. 3.1.

Because of the charge-sharing effects, the track recognition should consider clusters, instead of hits, produced by the charged particle in RP detectors. Hence, the proposed methods will start finding clusters with the algorithm described in chapter 2.

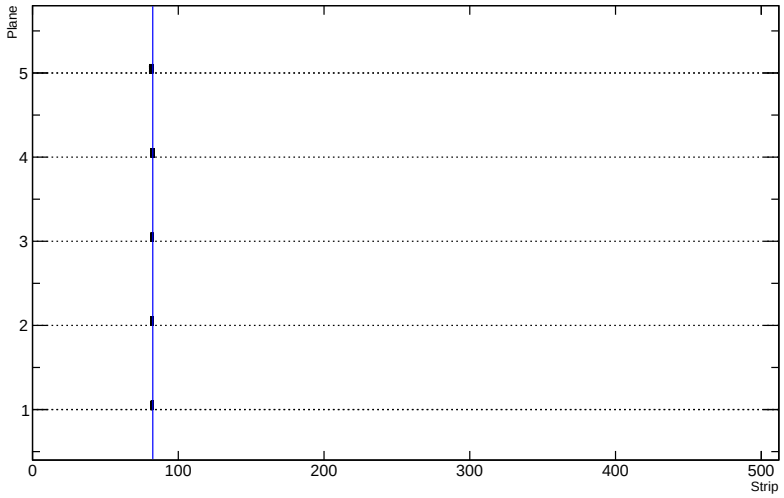


Figure 3.1: Coordinate system for rod search algorithm

Furthermore, track recognition algorithms should work on a single projections: in any case, tracks will be treated separately for u and v. A filtering algorithm based on track reconstruction can be configured to be as inclusive as possible, to avoid loss of useful data; for instance, an event could be accepted when a track is found either in u or v projections. However, to compute the position of the track, it is necessary to have only one track in both u and v projections, see section 1.3.3.

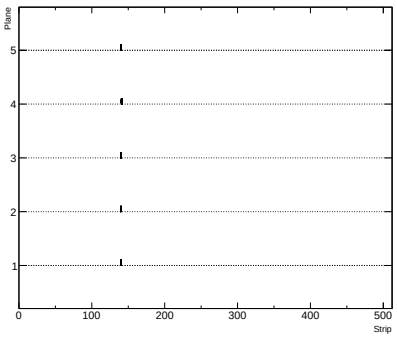
3.2 Histogram of the hits

It is possible to note that an *elastic* track is almost straight, and therefore made by a collection of hits on different planes with almost the same strip number, as shown in Fig. 3.1. However, the planes inside the RP detector can be not perfectly aligned and the whole detector can be not perfectly aligned with the beam. This misalignment explains the small "displacement" in the strip hit by the particle from a plane to the next.

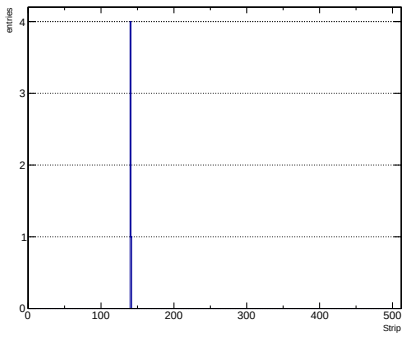
To highlight the difference between an *elastic* track and a *non-elastic* one, an histogram of the total number of hits for all five planes in the same projection can be filled. Each bin of the histogram corresponds to a strip (x axis) and its number of entries is the number of planes with an hit in that strip (y axis).

A preliminary analysis of real data confirmed that, in the case of an *elastic* track, the histogram has a large part of the entries in one single bin. In the opposite case, the histogram corresponding to an oblique track has no bins filled with more than a single

entry. Some examples are shown in Fig. 3.2(b) and Fig. 3.3(b).

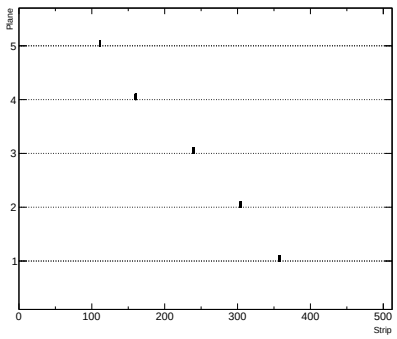


(a) Hit distributions in the five planes.

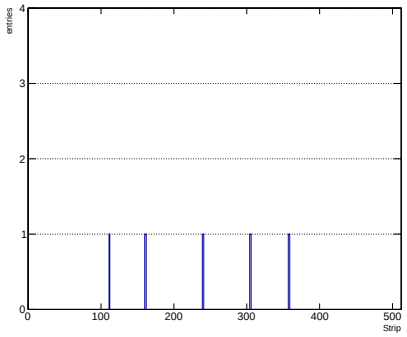


(b) Histogram of the hits of all five planes.

Figure 3.2: Example of an *elastic* track. In Fig. 3.2(a) is shown the hit distributions for the five planes; In Fig. 3.2(b) is shown the histogram filled with hits acquired by all five planes together. The histogram has a bin with 4 entries.



(a) Hit distributions in the five planes.



(b) Histogram of the hits of all five planes.

Figure 3.3: Example of an *non-elastic* track. In Fig. 3.3(a) is shown the hit distributions for the five planes; In Fig. 3.3(b) is shown the histogram filled with hits acquired by all five planes together. The histogram has no bin with more than one entry.

This analysis suggests that *elastic* tracks can be quickly found building such an histogram and searching for bin entries above a threshold. As mentioned before, because of the charge-sharing effects, the histogram should be filled with the position of the center of mass of the clusters, instead of simple hit positions.

It is also possible to account for the above mentioned misalignment problems by re-binning the histogram. However, this introduces a further problem: a track could be exactly on the edge between two consecutive bins; this case will result in two half filled bins and maybe none of them will be above the threshold.

This problem can be avoided using a sort of dynamic binning. The histogram does not have a fixed number of bins and their positions are not known *a priori*. All positions without clusters will be skipped, while, if a cluster is found, a bin that begins in that position will be created. The width of the bins can be configured, depending on the misalignment between planes.

As well as in the previous case, if a bin has entries over a threshold, it is an indication of an *elastic* track.

Some minor improvement can be also implemented. Instead of using the cluster size to compute the cluster center of mass, it is possible to use the begin of the cluster to fill the histogram. An additional modification might consist of using the total number of hits per planes to assign a weight to the clusters. In this way the effects of noisy planes are reduced. However, in a future hardware implementation, this last modification will lead to a large usage of memory resources. Indeed, the computation has to wait until all cluster have been found and - moreover - they need memory to be stored. In chapter 4 all these algorithms will be tested on real data to understand if their recognition performance justifies their resource cost.

3.2.1 The *DynamicHistogram* class

In more detail, the histogram is implemented as a *struct*, called *SingleBinHisto*, that stores a *vector*[1] of integer values. The *struct* constructor requires only one parameter: the number of bins of the histogram[2]. It has been kept as simple as possible to achieve the maximum speed performance. It should be noted that we have chosen to have an histogram with bins only corresponding to positive values. The increment of the entries corresponding to a given bin can be done using a method of *SingleBinHisto*.

A *struct*, called *DynamicBin*, is used to manage a single dynamic bin of the histogram. Cycling through *SingleBinHisto*'s bins, when an entry is found, the corresponding strip number and the number of entries are stored in a *DynamicBin* object. This object is initialized with a parameter, called *binWidth*, that indicates the maximum width the *DynamicBin* can have, or, in other words, the minimum separation between a *DynamicBin* and the next. Moreover, *DynamicBin* has a method, called *add*, to increment its entries. It

[1] A *vector* is one of the container class templates in the standard library of the *C++* programming language that implement storage of data elements.

[2] In TOTEM RP system, the number of bins is fixed and corresponds to the number of strips in each silicon plane, that is 512.

checks whether the bin includes a given position[3] or not: if yes it increments entries and returns *true*, otherwise it returns *false*.

A *DynamicHistogram* class takes care of collecting *DynamicBin* objects and to manage them. It is initialized with two parameters and it has an internal counter. One parameter is the *binWidth* and it is used to initialize *DynamicBin*s. The other is used as a threshold to say if a *DynamicBin* is filled enough to indicate a track. When a bin is filled above this threshold, the internal counter is incremented.

The class has a method, called *insert*, that calls the above mentioned *add* method of the last created *DynamicBin* to increment its entries if the bin position is compatible. Indeed, if the *add* method returns *true*, the number of entries of the *DynamicBin* is incremented. Otherwise, the number of entries of the *DynamicBin* is compared with the threshold; if it is above the threshold, the internal counter is incremented.

Hence, the internal counter is incremented every time a track candidate is found.

This algorithm is showed in Fig. 3.4.

3.2.2 Track recognition

A steering class, *TrackRecognition*, is needed to start the cluster search, using the algorithm described in chapter 2, and to compute the track recognition. This class uses an *OptoRxDataFrame* class form the TOTEM DAQ library to access *OptoRx* data, as already explained for the *optoRxStat* class in section 2.1.5. The *TrackRecognition* class manages both *SingleBinHisto* and *DynamicHistogram* and, after the cluster research, takes care of extracting cluster positions from corresponding *bitStat*s to fill the *SingleBinHisto* and to compute the number of track candidates using *DynamicHistogram*.

An important task of the *TrackRecognition* class is to associate *bitStat* objects to planes using their position in the *OptoRx* frame (see section 1.4.3). Indeed, a method, named *setPlanes*, creates a collection of pointers to *bitStat* objects grouped by planes of the same projection and the same RP detector. This method needs two parameters to choose the RP detector and the projection, u or v, on which to run the track recognition:

- *configuration*: the position of the chosen detector in the *OptoRx* frame;
- *uProjection*: *true* to choose the u projection; *false* to chose the v projection.

The track recognition itself is performed by the *Compute* method. It checks if the collection created using *setPlanes* is not empty and, if it has not been already done, it runs the cluster research for the whole *OptoRx* data frame. Then it starts the track recognition in the indicated detector and returns how many tracks have been found. This method fills the above described histogram using an object, a *functor*, implemented with an operator *()* that reads the information stored in the collection created by *setPlanes*.

[3] A position is included in a *DynamicBin* if $pos_{bin} - pos_{DynamicBin} \leq binWidth$

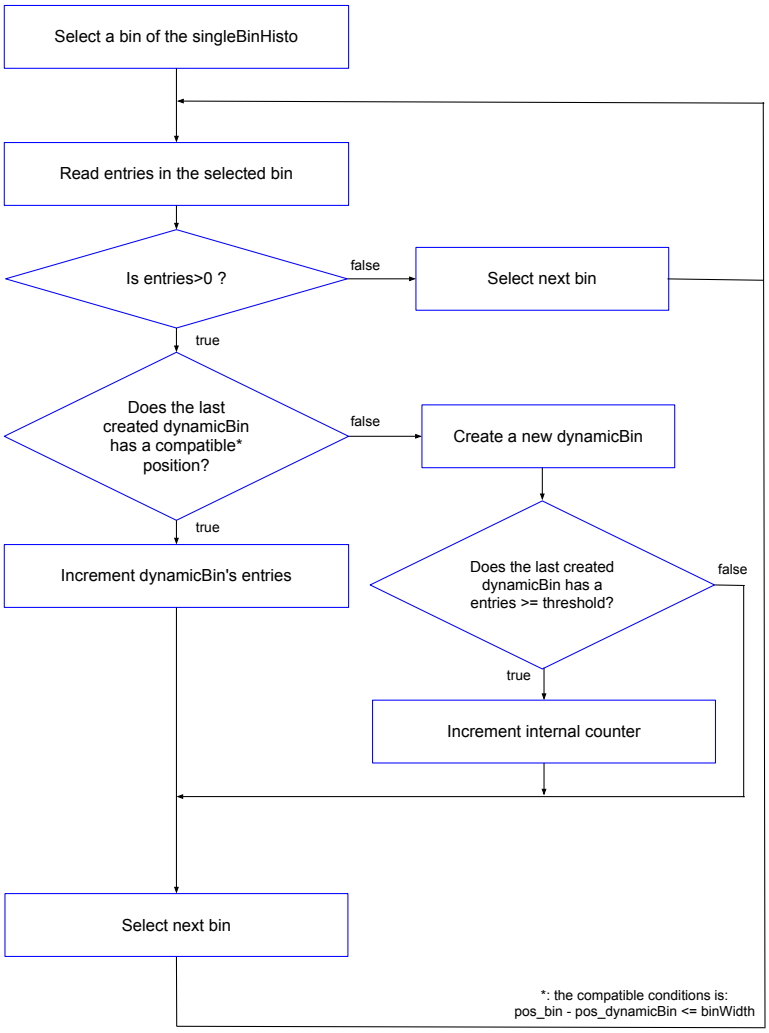


Figure 3.4: Flow chart of the dynamic binning algorithm.

In this way, the *TrackRecognition* class can be generalized using a template to choose the *functor* object that fills the histogram.

TrackRecognition class has been implemented as a template and the parameter must have an operator *()*. For the purpose of the track recognition, the template will be specialized to the *functor* that will be used to fill the *SingleBinHisto*. This implementation will allow to easily change the algorithm to fill the histogram, without modification of the steering class.

To implement the described track recognition algorithm, the *functor* has to scan the collection of bitStats and fill the histogram using the information in each bitStat.

TrackRecognition needs two parameters for the configuration, that can be set directly at the creation of the class:

- the *DynamicBin* maximum width;
- the threshold on the *DynamicBin* entries.

Performance and tuning of algorithm's parameters will be discussed in chapter 4.

3.3 Simplified Hough transform

The Hough transform is a smart method first proposed by Hough in 1962 [19] to improve and speed up track recognition in a bubble chambers. The Hough transform is at the base of many rod-searching algorithms, very useful in image analysis, computer vision and digital image processing.

The idea is to transform each point in the data, the *source space* with coordinate x and y, in a line in a *parameter space*, with coordinate m and q. If the points of the *source space* lay on a line, all lines in the *parameter space* will intersect in a point. The coordinates of this point corresponds to the parameters of the line.

In more details, the equation which describes a straight line in the *source space* is:

$$y = mx + q \tag{3.1}$$

This can be written in the *parameter space* as:

$$q = -xm + y \tag{3.2}$$

Hence, there is a correspondence between points in the *source space* and lines in the *parameter space*.

If two points (x_1, y_1) and (x_2, y_2) belong to the same line

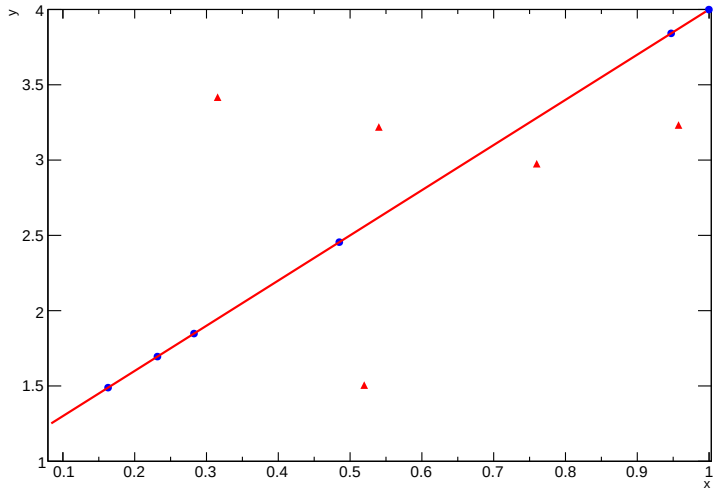
$$y = Mx + Q \tag{3.3}$$

the two Hough-transformed lines will intersect in the point (M, Q):

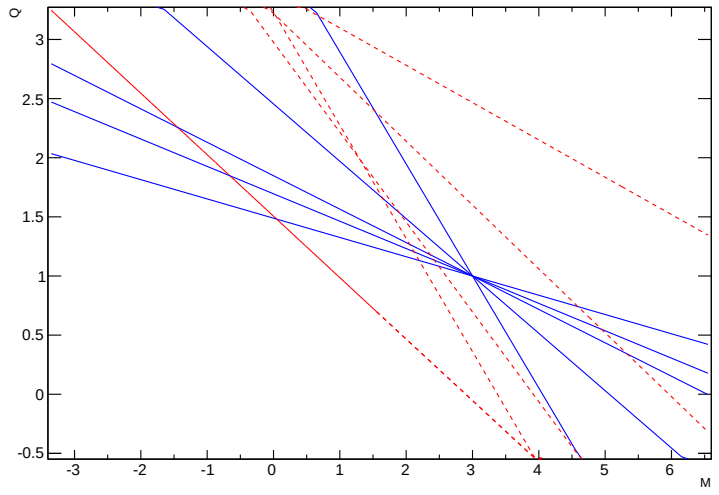
$$\begin{cases} Q = -x_1 M + y_1 \\ Q = -x_2 M + y_2 \end{cases}$$

$$\begin{cases} M = -\frac{y_2 - y_1}{x_2 - x_1} \\ Q = -\frac{x_2 y_1 - x_1 y_2}{x_2 - x_1} \end{cases}$$

If different points of the same line are used, all the Hough-transformed lines will intersect in the same point. All other points, e.g. due to noise, will be transformed in lines which do not intersect in the same point, as shown in Fig. 3.5.



(a) *Source space*. Blue points satisfy $y = 1 + 3x$; red triangles are random.



(b) *Parameter space*. Blue solid lines intersect at point (3,1).

Figure 3.5: If the points in the *source space* lay on a straight line (blue points in Fig. 3.5(a)), their Hough transform in the *parameter space* will show intersecting lines (solid blue lines in Fig. 3.5(b)). If the points in the *source space* are randomly chosen (red triangles in Fig. 3.5(a)), their Hough transform in the *parameter space* will show lines that do not intersect in the same point (dashed red lines in Fig. 3.5(b)).

If all the source points belong to the same line, a bi-dimensional histogram filled with M and Q computed for each pair of intersecting lines in the *parameter space* will have entries in only one bin. If some of the source points are due to noise, or the line is not perfectly sharp, the histogram will show a cluster around a certain bin. The coordinates of this cluster will give the parameters of a line; the smaller the cluster size the smaller the errors on the parameters.

In a simplified way, it is possible to consider only pairs of points belonging to two successive planes of the same projection. In the proposed coordinate system (Fig. 3.1), the difference $y_2 - y_1$ always equals to 1. Defining

$$\bar{m} = |\frac{1}{M}| \tag{3.4}$$

for each pair of points in the *source space*, $\bar{m} = \frac{x_2 - x_1}{y_2 - y_1} = x_2 - x_1$ can be computed and an histogram of $\bar{m}$ can be easily filled. A peak in this histogram will suggest the presence of a track.

Finally, it is possible to further simplify the algorithm. For some applications it could be useless to know both parameters of the line. Indeed, avoiding the computation of q and using only a mono-dimensional histogram of m it is possible to understand if there is at least one track without knowing the exact number of tracks.

An improvement could be to consider not only adjacent planes but also interleaved planes of the same projection; indeed, their distance $y_2 - y_1$ is 2 and the division by two is easily done in an FPGA. However, this approach will increase the number of hit pairs to be analyzed.

The complexity of this algorithm is $o(n^2)$ instead of $o(n)$ like the method proposed in section 3.2, where n is the number of clusters found.

3.3.1 Track recognition with a simplified Hough transform

The track recognition using the Hough transform is based on the searching of a peak in an histogram. This is exactly the same procedure of the algorithm seen in section 3.2. Thanks to the fact that the *TrackRecognition* class has been implemented as a template, see section 3.2.2, it is possible to use the same template class, specialized to a custom *functor* that implements the Hough transform.

This *functor* will cycle trough all pairs of clusters found in consecutive or interleaved planes and will fill the histogram with the calculated $\bar{m}$.

It should be noted that $\bar{m}$ values are positive numbers included between 0 and 511; so, the range of the histogram is exactly the same as the one used in the previous method.

Chapter 4

Data Analysis

In chapter 2 an algorithm for cluster analysis has been proposed. This algorithm has been implemented both via software, using C++, and in firmware, using the VHDL language. Both implementations need to be tested to be sure that all and only actual clusters are found.

In chapter 3, two different approach to the track recognition were proposed. Also these algorithms have been implemented and their performances have to be benchmarked.

In this chapter these algorithms will be tested using real data and their results will be compared with the off-line reconstructing software.

To investigate the possibility to reduce acquired data, the performance of the algorithms for data reduction will be computed filtering at the level of a single RP detector, a single *OptoRx* and for whole events.

4.1 Data set

To benchmark the performances of the proposed algorithms, a sub-set of the data acquired by TOTEM during the year 2011 will be analyzed. The chosen sub-set is representative of different run typologies; these typologies differ mainly for the luminosity and for the optics used before near the Interaction Point.

A parameter to quickly understand the behavior of the optics is the β^* that can be seen as the focal length of an optical lens. Most of the time the LHC is configured to run with a low β^* optics, to achieve the highest luminosity possible; however, high β^* runs are important for TOTEM to study the elastic scattering. During TOTEM dedicated runs, an high β^* optics is used and, to have a lower pileup, the luminosity is lower than usual. From the point of view of track recognition for the RP detectors, have a lower pileup means that is easier to distinguish single tracks.

An other important parameter that drove the choice of the data set is the trigger schema. In fact, TOTEM trigger system foresees the possibility to trigger on both T2 and

RP detectors. Often, due to the topology of the detectors, an event triggered by T2 has no tracks in RP detectors. Moreover, if the pileup is low, it is very likely that only a part of the RP detectors will record tracks. For these reason data acquired with two different trigger configuration have been included in the data set.

These remarks will be highlighted in section 4.2.

Some useful information about the chosen data set are available in table 4.1.

Run	Date-time	RP distance	Trigger schema	Energy per beam	β^*
5608	18.05.2011 20 : 22 : 40	$5.0\,\sigma$	RP_{45} AND RP_{56}	$3.5\,TeV$	$1.5m$
5657	29.06.2011 04 : 44 : 45	$10.0\,\sigma$	T2 AND (RP_{45} OR RP_{56})	$3.5\,TeV$	$90.0m$
6945	20.10.2011 21 : 48 : 55	$4.8\,\sigma$	T2 AND BX AND (RP_{45} AND RP_{56})	$3.5\,TeV$	$90.0m$

Table 4.1: Data sample used for tests and benchmarks.

4.2 Data analysis using the off-line software

The TOTEM off-line software, described in section 1.5, is used for track recognition in RP detectors and for event reconstruction. For this reason, it is useful to compare the performances of the proposed algorithms with the results of the off-line analysis.

The off-line software has been modified to extract, for each RP detector, the number of found clusters and the angle of the recognized track, if any, in both u and v projections. The selected data set has been analyzed using the customized version of the software. Moreover, the percentage of RP detectors, *OptoRx*s and events with at least one track have been computed.

The results of this analysis are summarized in table 4.2.

Run	Number of candidate events	Total number of tracks	Percentage of RPs with tracks	Percentage of *OptoRx*s with tracks	Percentage of events with tracks
5608	17646	38657	18.3	48.6	78.2
5657	17646	15929	7.5	22.2	36.6
6945	17604	15366	7.3	21.3	29.1

Table 4.2: Performance of the off-line software.

As expected, not all acquired events contains recognized tracks and the percentage of events without tracks (in RP detectors) is lower for runs where the trigger includes T2

(runs 5657 and 6945). Furthermore, it should be noted that, on average, tracks are found in one quarter of the RP detectors.

It should be noted that, especially for runs with the trigger configured to include T2, the percentage of the data from RPs that contains at least one usable track is very low. This suggest that a good data reduction algorithm could produce a great improvement on the DAQ performances (up to 10 times).

4.3 Cluster algorithm performance

To check the accuracy of the software cluster searching algorithm proposed in chapter 2, the first step has been a one to one comparison between found clusters and a small subset of the raw data. Then, the results of the algorithm has been automatically compared with the number of cluster found by the off-line software for all the above mentioned data set. A 100% correspondence has been found.

After that the accuracy of the software implementation of the cluster searching algorithm has been proved, the hardware implementation (see section 2.3) can be tested comparing the results with the software implementation, without using the off-line software.

4.3.1 Firmware place and route

The hardware oriented algorithm has been implemented in the firmware of the *OptoRx*.

Before to test the firmware on the *OptoRx* FPGA, the design has been simulated using an *ad hoc* simulation library developed by the TOTEM DAQ group. This library provides some powerful tools, developed in VHDL and System Verilog [20], to simulate the *OptoRx* firmware as a black box. Indeed, it is possible to inject fibers signal and to use a simulated local bus to configure the firmware and to read status register. Thanks to these tools, a sub set of raw data has been used to inject data and simulate the cluster analysis. Then the content of the FIFOs has been read using the simulated local bus and it has been compared with the results of the software implementation, with a full correspondence.

After the preliminary test using the simulation, the firmware has been programmed into the FPGA and tested using the RP test setup in the TOTEM laboratory. The data acquisition software has been modified to collect, together with the acquired raw data, the information stored into the *bitStat*s' FIFO. These information, after the acquisition, have been used to check the consistency of the found clusters.

Finally, following the same procedure, the firmware has been tested using the RP detectors installed at the Interaction Point during a dedicated data taking. Particular attention was given to the trigger configuration. Indeed, the generation of a LV1A signal concurrent with the reading process of the FIFOs could lead to a corruption of read data.

Hence, the trigger has been prescaled to have a very slow ($\sim 10\,Hz$) trigger rate. Also in this case, a 100% correspondence has been found between the cluster found by the firmware and that found by the software.

4.3.2 Cluster per plane filter performance

In section 2.2, a filter on the number of cluster per plane has been proposed. Its performances have been benchmarked using the off-line software.

The filter can be configured with two parameters: the *min* and the *max* number of cluster can have to be accepted. If more than two planes per RP are accepted, the whole RP is accepted. A first stage is to set the *max* to a virtually infinite value [1] to reject only *empty RP*s, i.e. RPs with at least four planes with no clusters for each projections.

Given the DAQ architecture, it is worth considering the percentage of *empty RP*s as well as the percentage of *empty OptoRx*s and *empty event*s. An *empty OptoRx* is an *OptoRx* where all connected RPs are *empty*. In the same way, an *empty event* is an event where all RPs are *empty*.

The performances of this filter are summarized in table 4.3.2.

Run	Number of candidate events	Percentage of RPs with clusters	Percentage of *OptoRx*s with clusters	Percentage of events with clusters
5608	17646	50.0	89.2	100
5657	17646	10.1	25.1	40.8
6945	17604	9.7	24.9	32.0

Table 4.3: Performance of the filtering on Pots with at least 4 planes with no clusters for each projections.

The run 5608 is a collection of events triggered only by the Roman Pot detectors; so, it is very unlikely to find an event without clusters, but still it is possible to reject a relevant percentage of *OptoRx*s. In the other two runs, for almost all events triggered by T2, Roman Pot detectors have no clusters. Thus, the use of this filter could led to a clear optimization of DAQ bandwidth and disk usage.

The other proposal was to reject also RP detectors with a number of clusters above a certain threshold. In more details, a RP will be accepted if $0 < cl \leq th_{cl}$ for at least 2 planes per projection, where cl is the number of cluster. For values of th_{cl} greater than 4, the filter is more conservative than the off-line software, in the sense that a detectors with $0 < cl \leq 4$ for at least 3 planes per projection is rejected by the off-line software.

[1] Virtually infinite, in this case, means bigger than 256 that is the maximum number of clusters that can be found in a single plane (512 strips.

The choice of th_{cl} can be inferred from the histograms shown in Figs. 4.1, 4.2, 4.3, where the percentage of accepted RP, *OptoRx*s and events is shown for different values of the threshold. The results are compared with the one found by the off-line software. It is worth noting that the off-line software has only one value corresponding to the configuration currently used for the TOTEM data analysis.

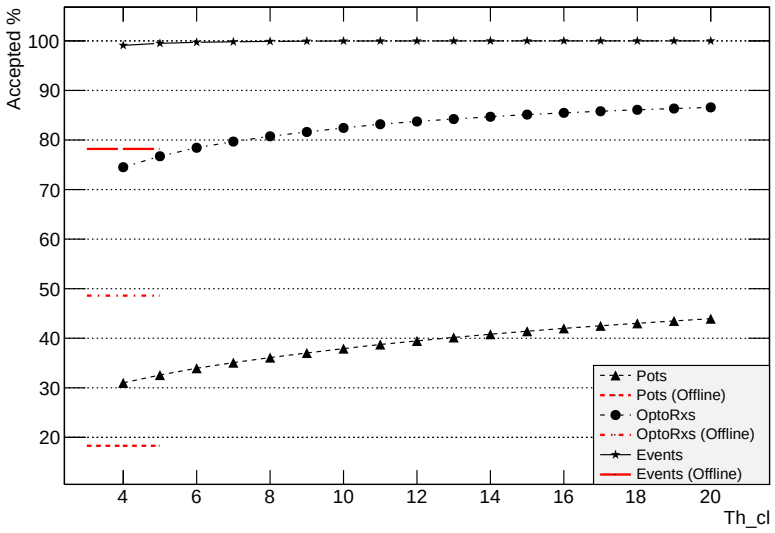


Figure 4.1: Percentage of accepted RP, *OptoRx*s and events. Red lines shows the percentage of RP, *OptoRx*s and events where the off-line software found a track. Data-set: run 5608.

On the base of the results shown in table 4.3.2 and in Figs. 4.1, 4.2, 4.3, it is possible to state that such filter on the number of clusters per plane found is a very powerful tool to reduce data and to improve the DAQ efficiency. Furthermore, this filter should be seen as a *zero suppression* algorithm. Indeed, it doesn't involves any patter recognition and, depending only on two threshold, it is relatively easy to configure and debug. Finally, it should be noted that, especially for low luminosity and high β^* runs, only a few percent of data is not filtered by the proposed method and rejected by the off-line track recognition. For example, analyzing data from run 6945, only $\sim 2.5\%$ of the RPs that have not been filtered, have been rejected by the off-line track recognition. Because of the high luminosity, for some typology of runs, a bigger percentage of RP, that do not contain reconstructible tracks, is accepted.

This means that, for low luminosity runs, using more complex algorithms for track recognition will not lead to a data reduction more than few percent better. However, it would be interesting to investigate the performance of the proposed track recognition

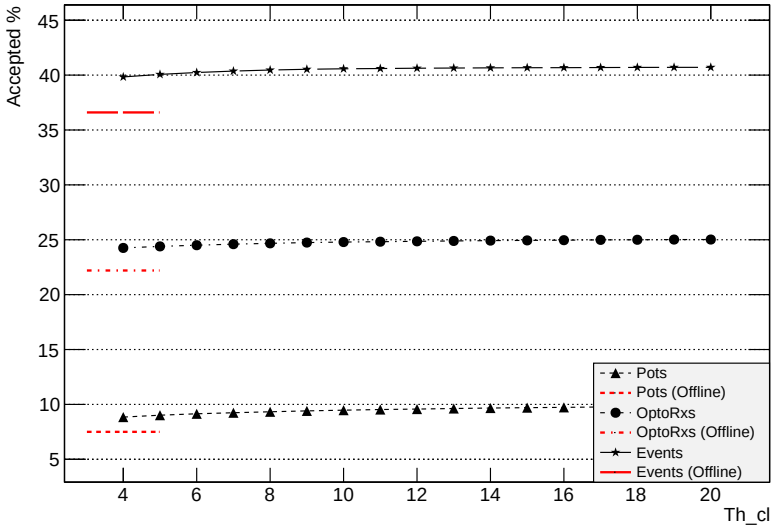


Figure 4.2: Percentage of accepted RP, *OptoRx*s and events. Red lines shows the percentage of RP, *OptoRx*s and events where the off-line software found a track. Data-set: run 5657.

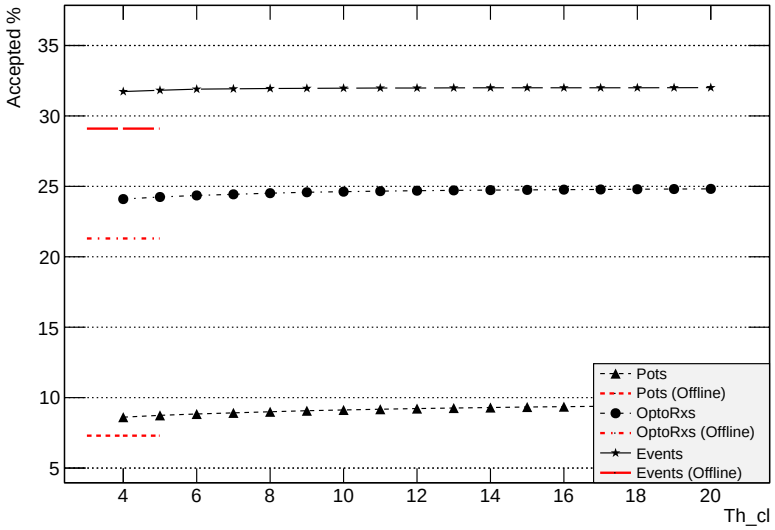


Figure 4.3: Percentage of accepted RP, *OptoRx*s and events. Red lines shows the percentage of RP, *OptoRx*s and events where the off-line software found a track. Data-set: run 6945.

algorithms for future development of some high level trigger algorithms and for data reduction during high luminosity runs.

4.4 Performances of the track recognition algorithms

The configuration of the track recognition algorithms proposed in chapter 3 needs two parameters. These parameters change the behavior of the track recognition and depends on the application. Indeed, the algorithms can be tuned either to be more accurate, rejecting a RP where a tracks is not perfectly recognizable, or to be more inclusive, rejecting only data where is not possible to reconstruct a track.

In particular, these parameters are:

- *Bin Width*: the maximum width a dynamic bin can have;
- *Threshold*: the minimum number of entries in the same dynamic bin that indicates the presence of a track.

A simplified view of the impact of the parameters on the algorithms is shown in Fig. 4.4.

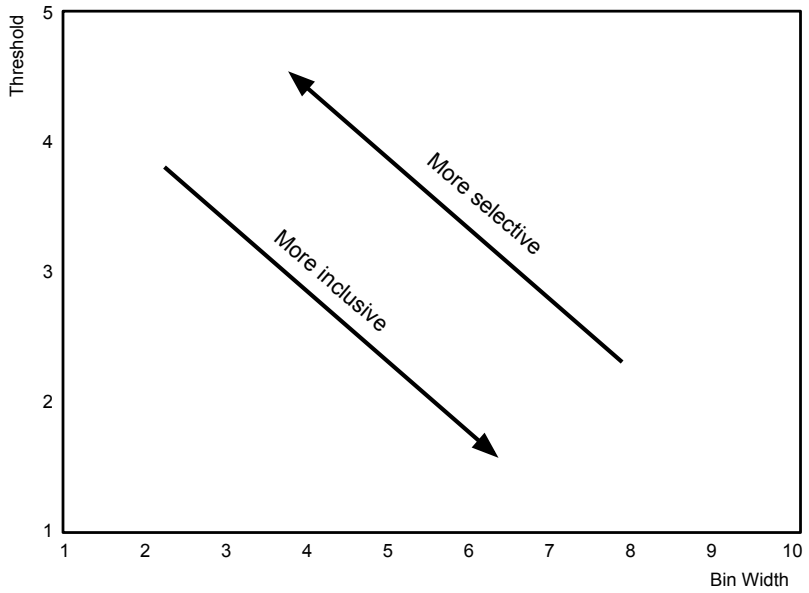


Figure 4.4: Simplified view of the impact of the parameters on the algorithms.

To produce the following data analysis, both algorithms have been set to work using the beginning of the cluster, ignoring their size, to search for tracks. Then, the same algorithms have been tested using the center of mass of the clusters. Furthermore, the off-line software is set to ignore all planes that have more than 4 clusters. Hence, this choice has been adopted in the proposed algorithms too.

Track recognition algorithms have been run independently on each projection of each RP detector. To benchmark them, some counter were used:

- *True Positive*: tracks recognized by the off-line and recognized by the algorithm;
- *True Negative*: tracks not recognized by the off-line and not recognized by the algorithm;
- *False Positive*: tracks not recognized by the off-line, but recognized by the algorithm;
- *False Negative*: tracks recognized by the off-line, but not recognized by the algorithm.

To understand the filtering capability of the algorithm, the important parameter is *False Positive*. If it is small, it means that only *real*[2] tracks have been accepted. On the other hand, to understand the efficiency of the algorithm, the important parameter is *False Negative*. Indeed, *False Negative* counts *real* tracks that are not recognized.

False Positive and *False Negative* are shown in the following histograms, varying the above discussed parameters.

4.4.1 Performances of the histogram based transform algorithm

Comparing the following histograms, for instance between Fig. 4.6 with Fig. 4.7, it is clear that the performances of track recognition are not affected by the choice of using the begin of clusters or their centroid. This results can be explained by the fact that almost all clusters that belongs to recognized tracks are small, as shown in Fig. 4.5.

[2]Recognized by the off-line software.

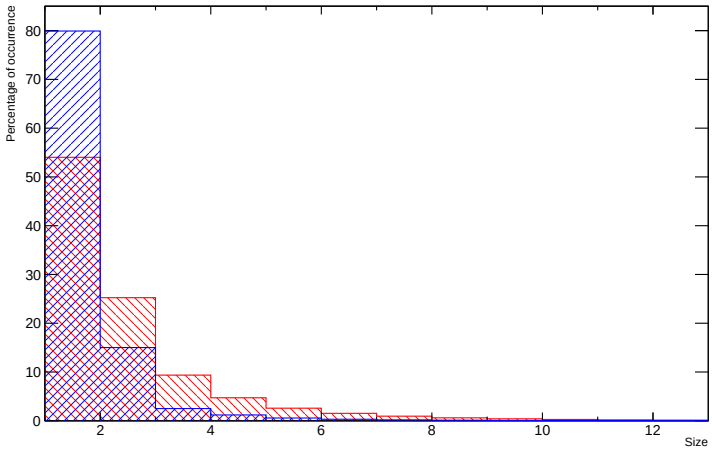
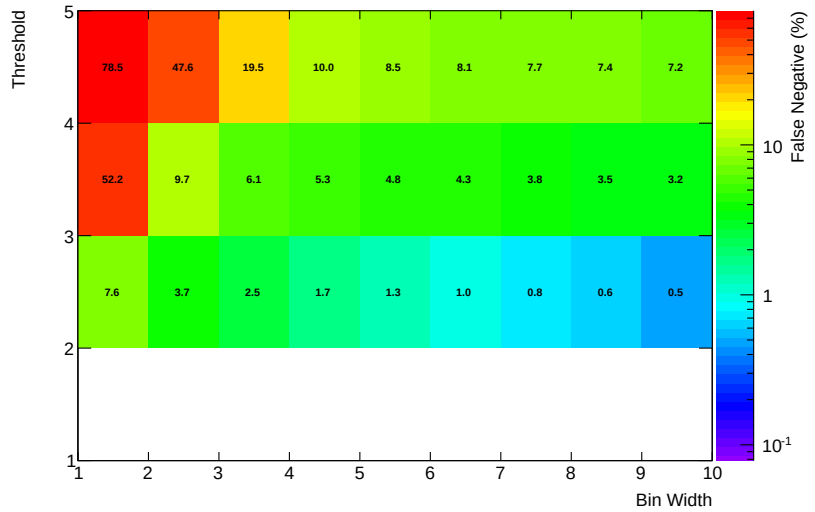
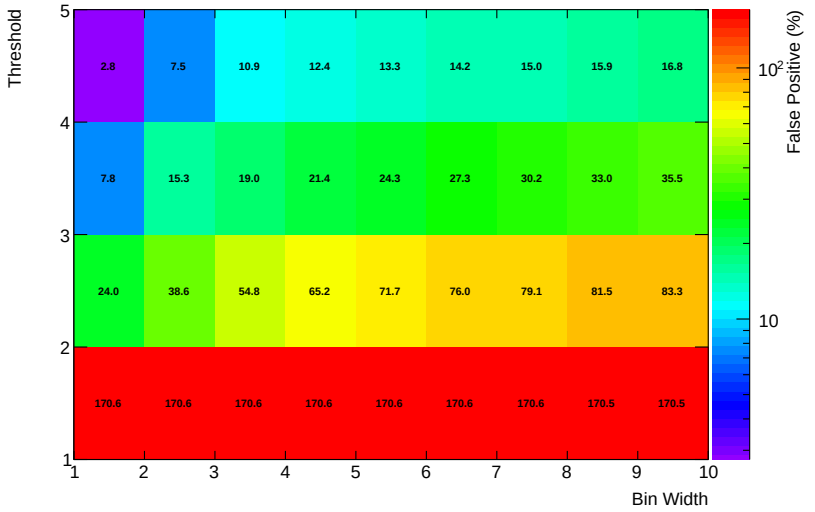


Figure 4.5: Size of clusters found in all data set (red) and only in detectors where a track has been recognized by the off-line software (blue). Data-set: run 5608.

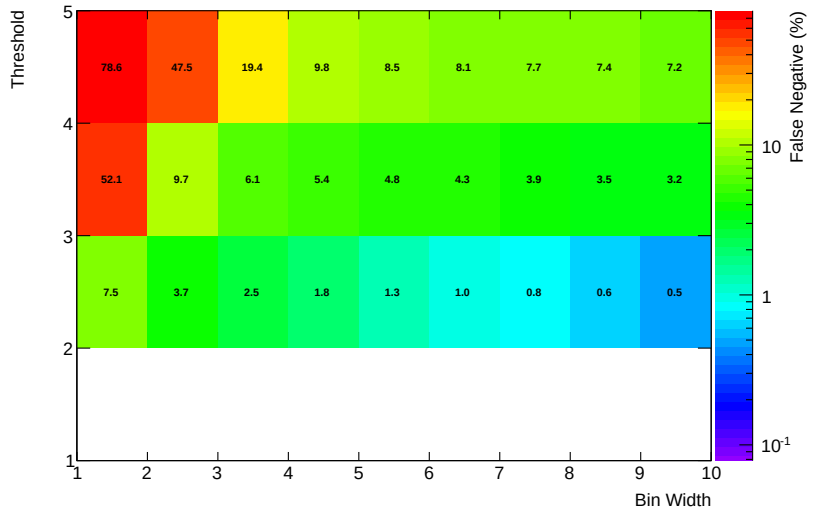


(a) Percentage of *False Negative*s

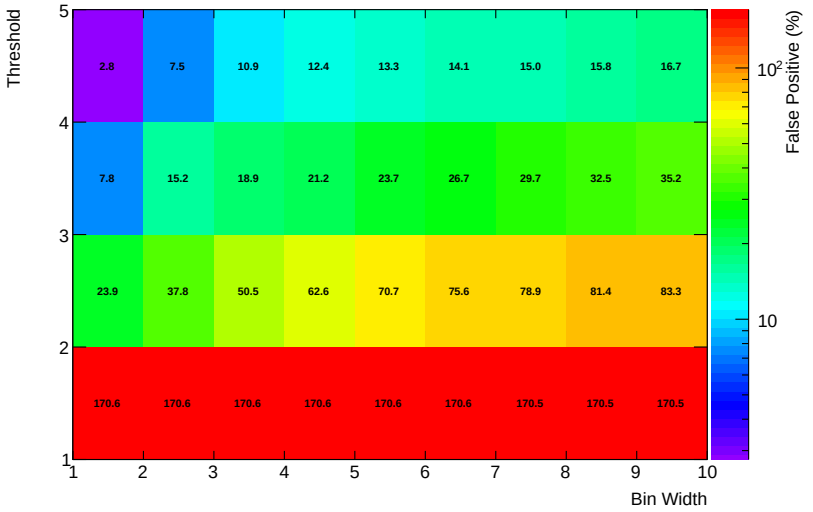


(b) Percentage of *False Positive*s

Figure 4.6: Performances of the histogram based track recognition algorithm, using only the position of the begin of the clusters. Data-set: run 5608.

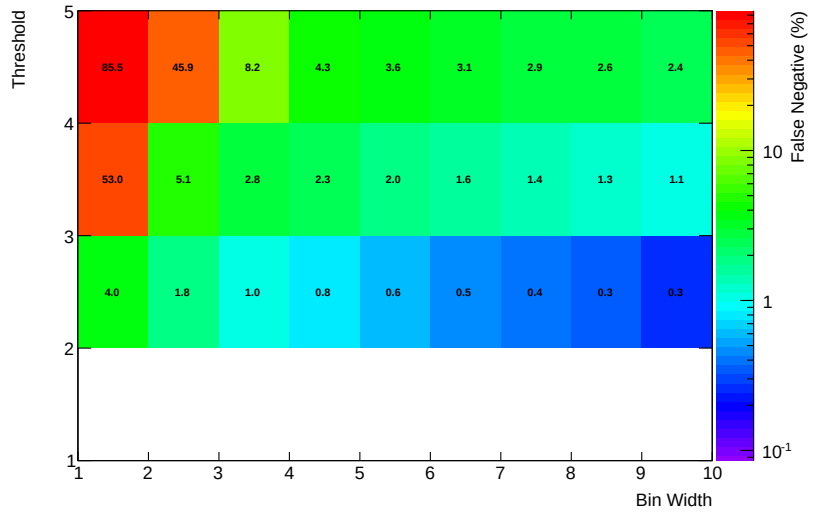


(a) Percentage of *False Negative*s

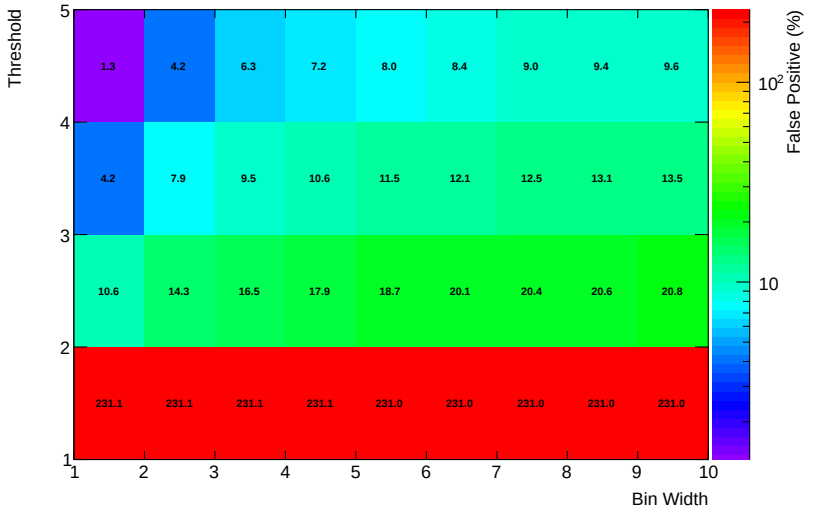


(b) Percentage of *False Positive*s

Figure 4.7: Performances of the histogram based track recognition algorithm, using the position of the centroid of the clusters. Data-set: run 5608.

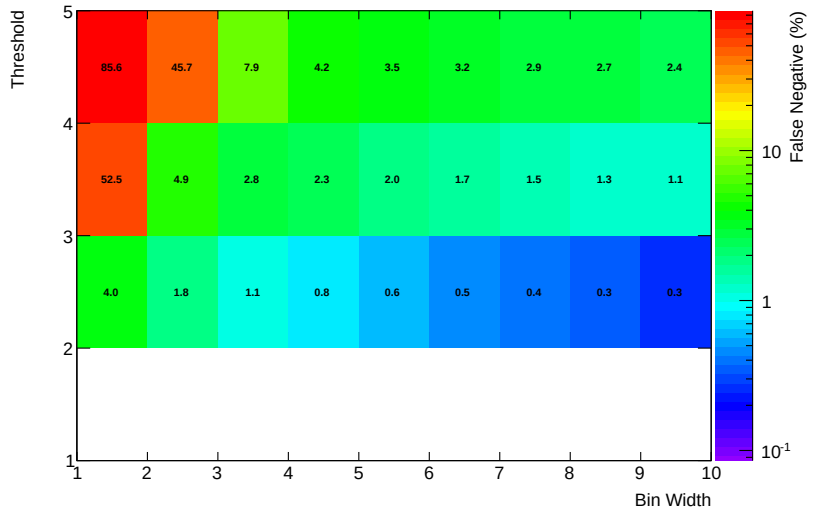


(a) Percentage of *False Negative*s

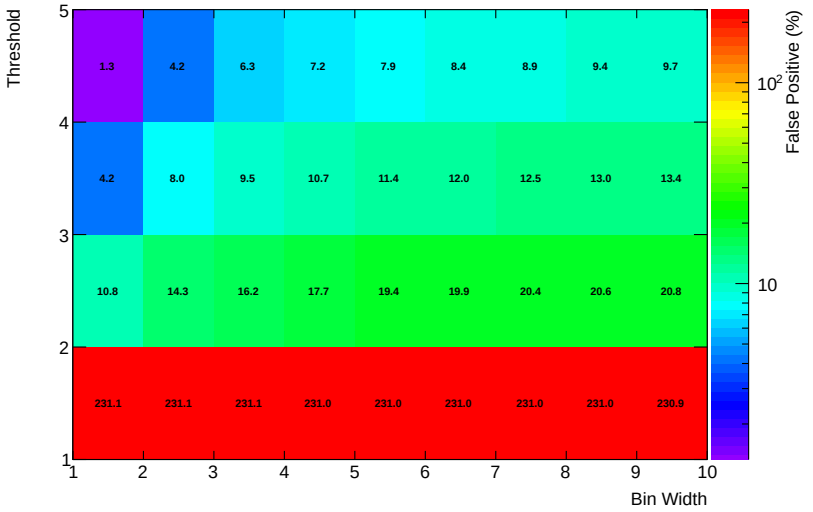


(b) Percentage of *False Positive*s

Figure 4.8: Performances of the histogram based track recognition algorithm, using only the position of the begin of the clusters. Data-set: run 5657.

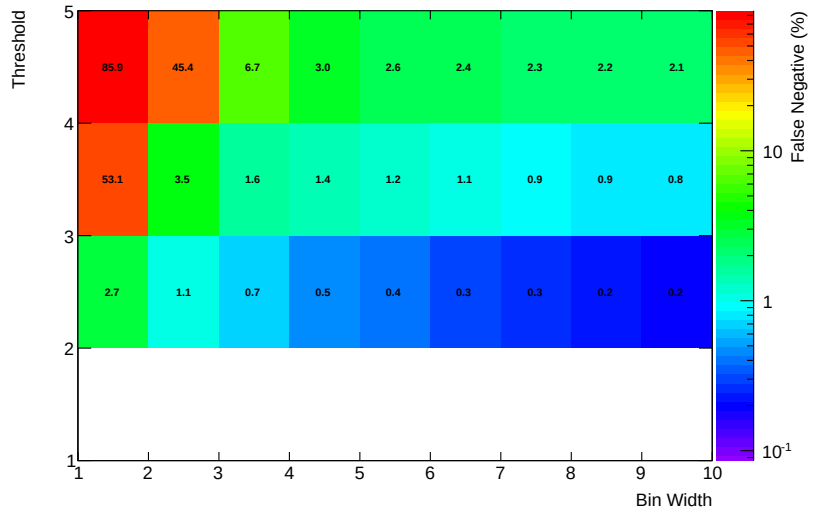


(a) Percentage of *False Negative*s

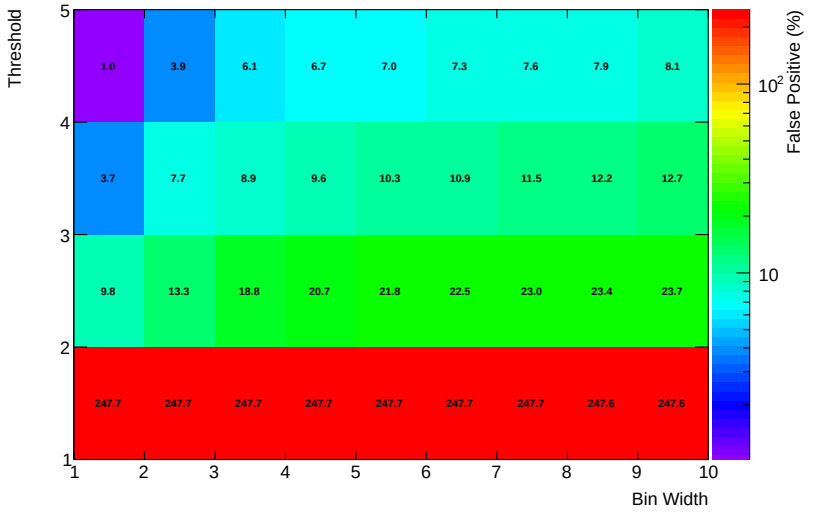


(b) Percentage of *False Positive*s

Figure 4.9: Performances of the histogram based track recognition algorithm, using the position of the centroid of the clusters. Data-set: run 5657.

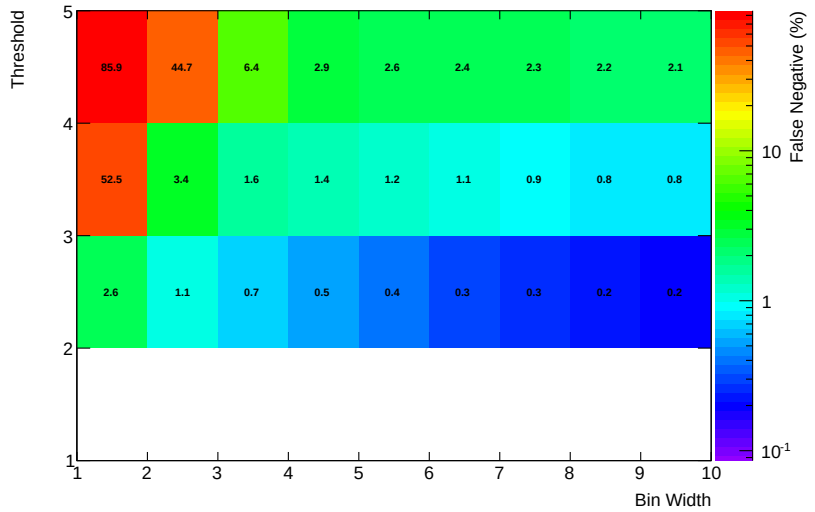


(a) Percentage of *False Negative*s

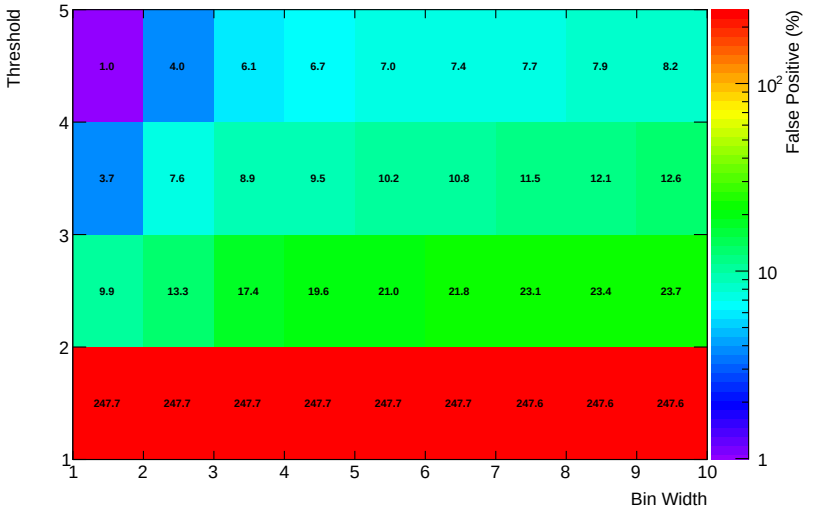


(b) Percentage of *False Positive*s

Figure 4.10: Performances of the histogram based track recognition algorithm, using only the position of the begin of the clusters. Data-set: run 6945.



(a) Percentage of *False Negative*s



(b) Percentage of *False Positive*s

Figure 4.11: Performances of the histogram based track recognition algorithm, using the position of the centroid of the clusters. Data-set: run 6945.

This algorithm is very powerful when tracks have small angles with the beam. Hence, can be useful to benchmark the algorithm ignoring all tracks recognized by the off-line software that have an angle bigger than 1 *mrad* in both u and v projections. This limit is big enough to include all tracks due to elastic scattered protons ($\sim 1\ \mu rad$) even considering alignment problems ($\sim 1\ mrad$). At the same time, the limit of 1 *mrad* is small enough to make tracks appear as hits with the same position in all RP planes.

With this limitation the number of *False Negative* is drastically lowered. This can be seen comparing the histogram in Fig. 4.7(a), made using all tracks,.with the one in Fig. 4.12, made using only tracks with an angle lower than 1 *mrad*.

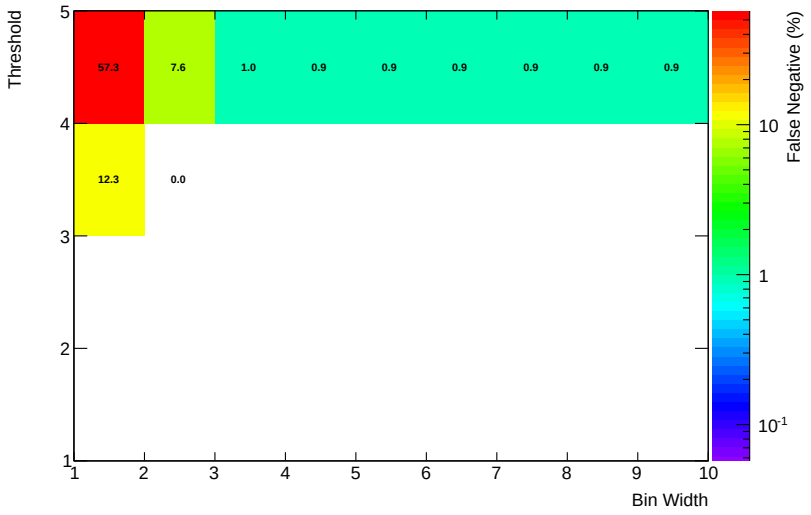
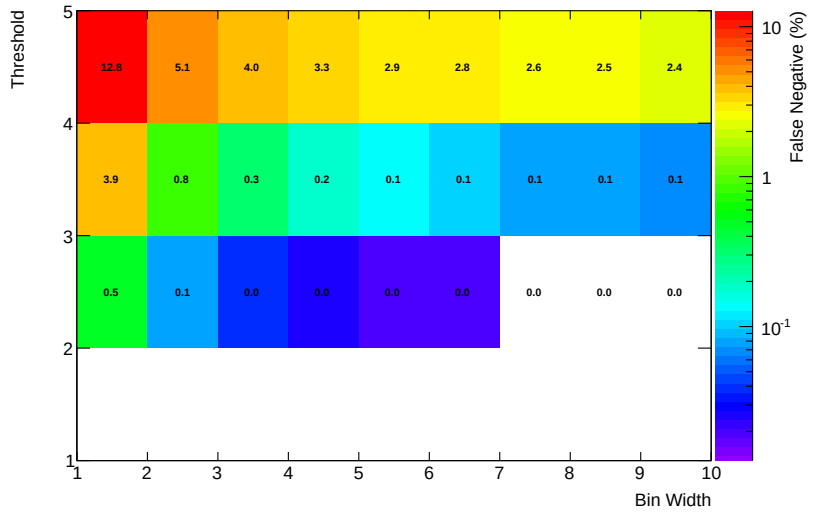


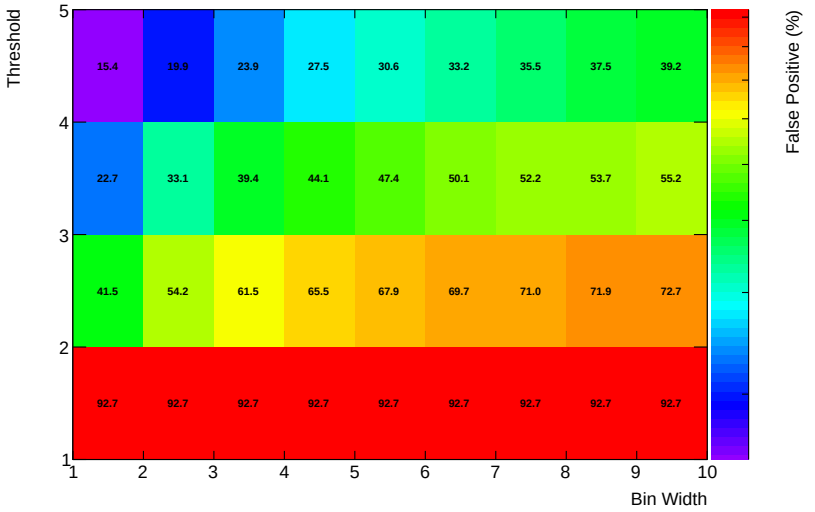
Figure 4.12: Percentage of *False Negative*s using the histogram based track recognition algorithm, using the position of the centroid of the clusters. Only tracks with angles less than 1 *mrad* in both u and v projections have been considered. Data-set: run 5608.

4.4.2 Performances of the Hough transform algorithm

The Hough based algorithm is more inclusive than the histogram based one, i.e. a lower number of *False Negative*. For instance this can be seen comparing the histogram in Fig. 4.15(a) with the one in Fig. 4.8(a) or the histogram in Fig. 4.13(a) with the one in Fig. 4.6(a). Indeed, it is possible (parameters *Bin Width* = 5 and *Threshold* = 3, in the histogram shown in Fig. 4.12) to not reject any track having only a $\sim$ 15% of *False Positive*s.

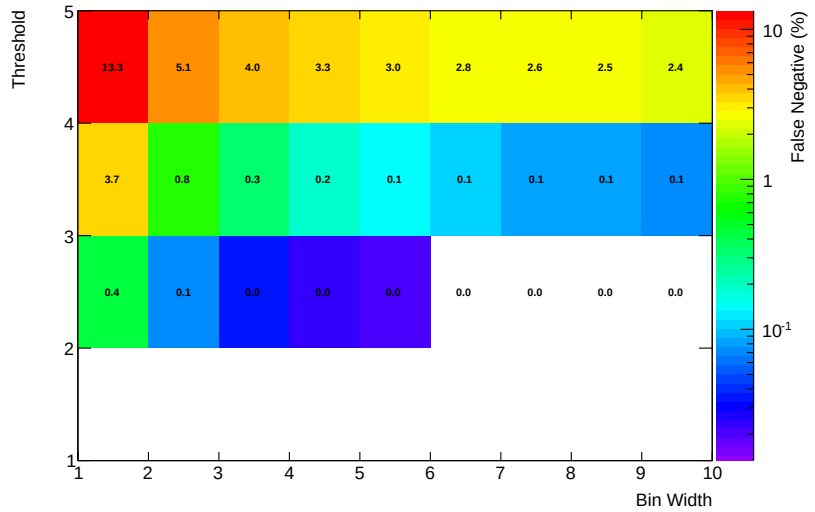


(a) Percentage of *False Negatives*

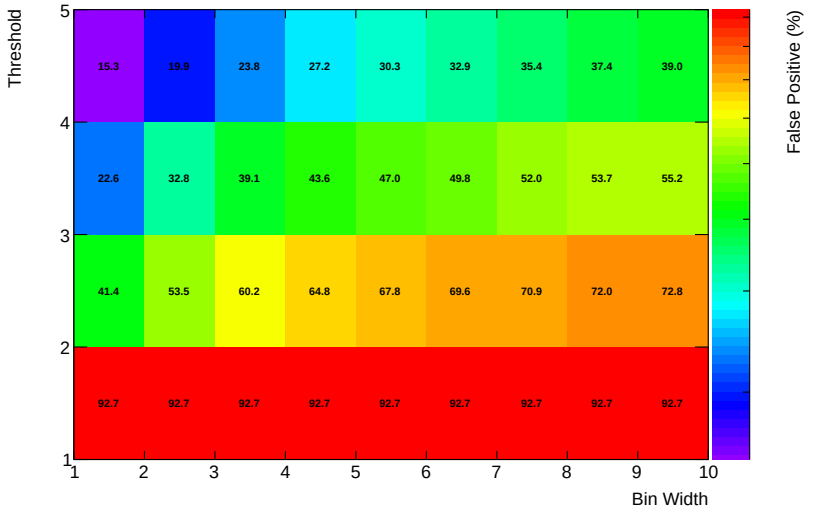


(b) Percentage of *False Positives*

Figure 4.13: Performances of the Hough transform track recognition algorithm, using only the position of the begin of the clusters. Data-set: run 5608.

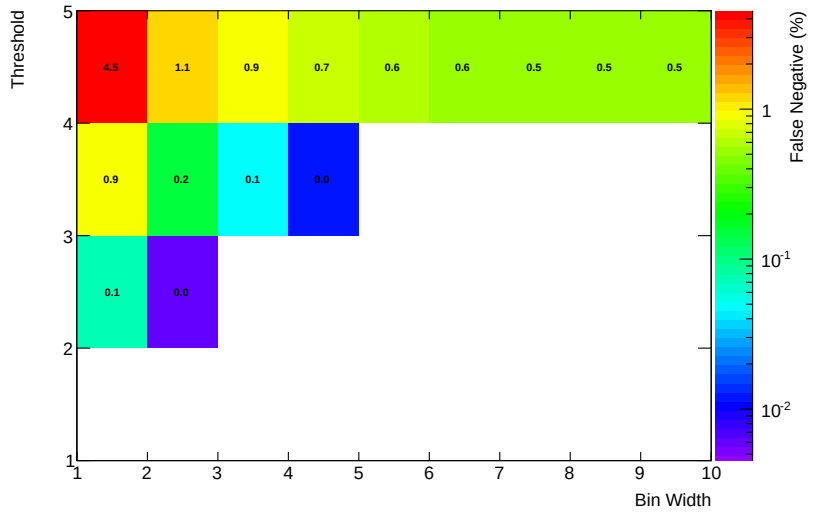


(a) Percentage of *False Negative*s

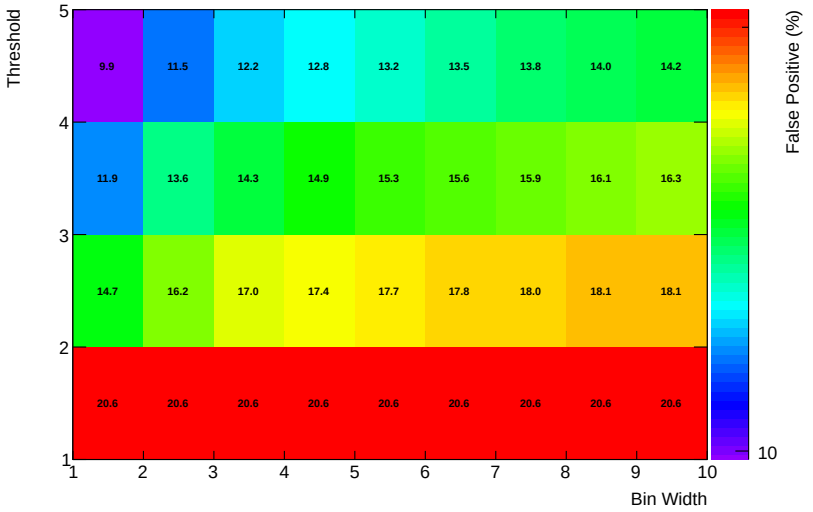


(b) Percentage of *False Positive*s

Figure 4.14: Performances of the Hough transform track recognition algorithm, using the position of the centroid of the clusters. Data-set: run 5608.

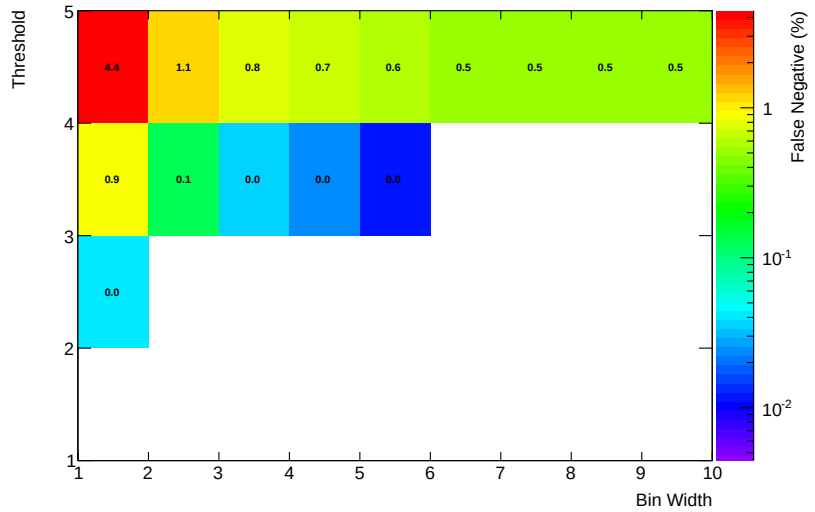


(a) Percentage of *False Negatives*

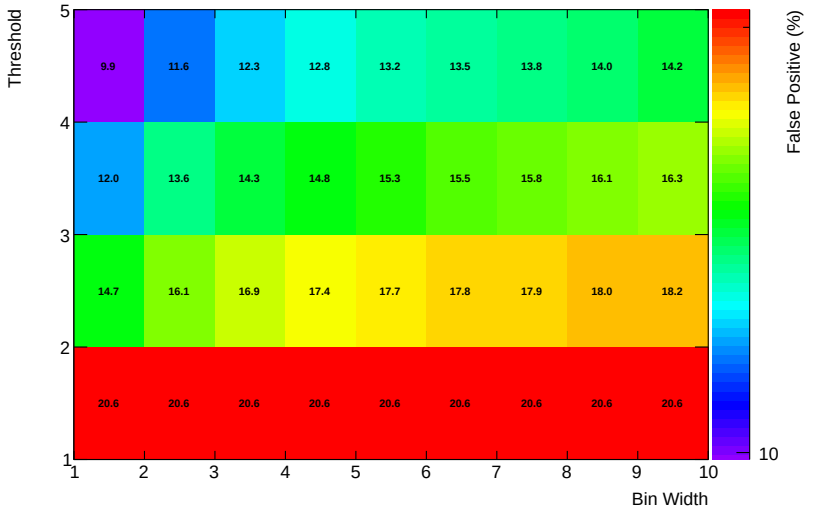


(b) Percentage of *False Positives*

Figure 4.15: Performances of the Hough transform track recognition algorithm, using only the position of the begin of the clusters. Data-set: run 5657.

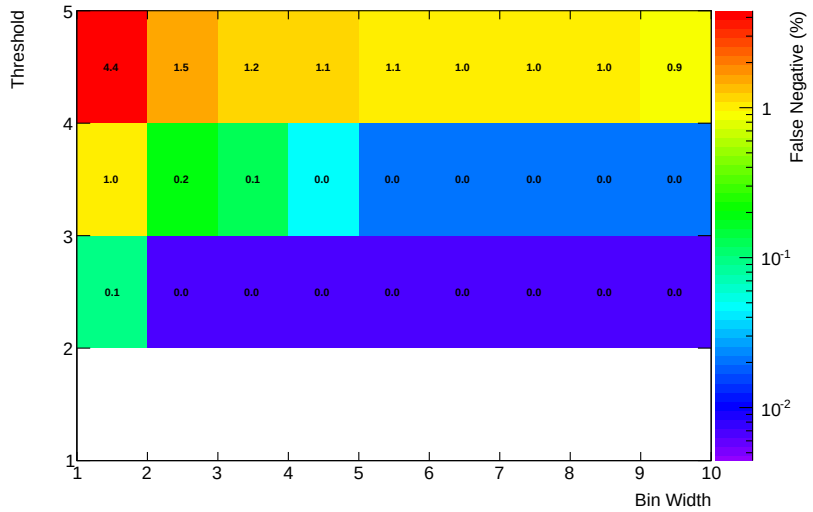


(a) Percentage of *False Negative*s

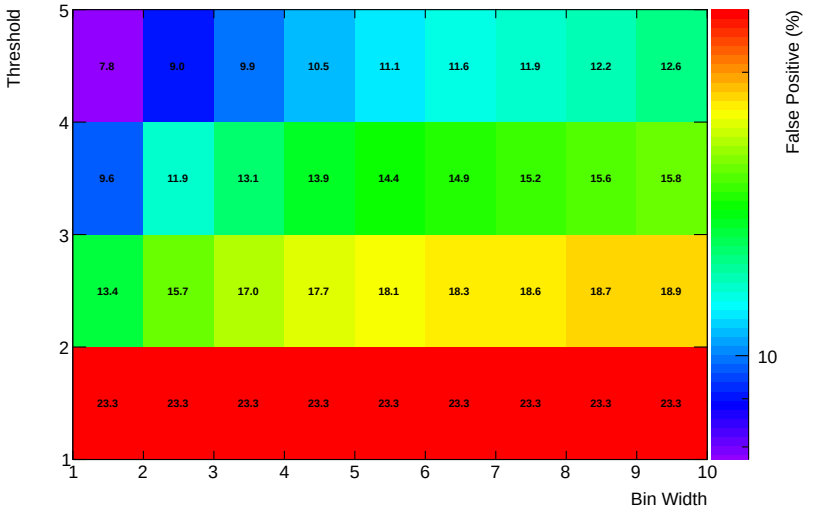


(b) Percentage of *False Positive*s

Figure 4.16: Performances of the Hough transform track recognition algorithm, using the position of the centroid of the clusters. Data-set: run 5657.

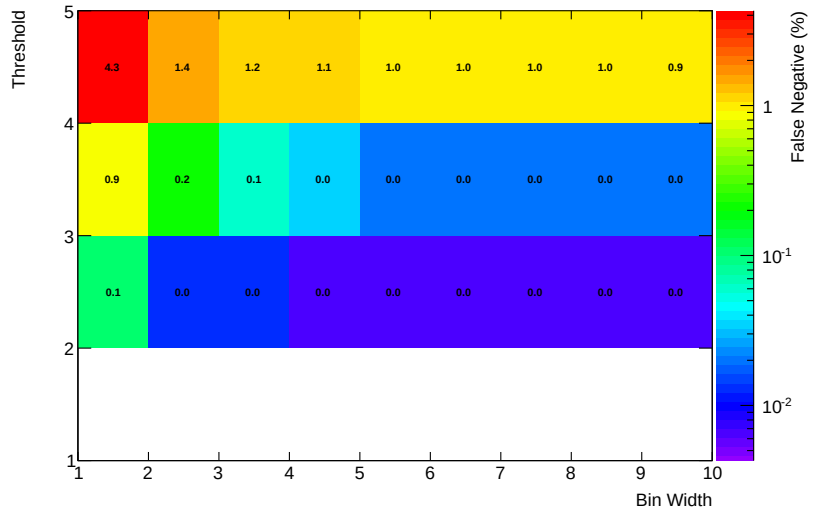


(a) Percentage of *False Negative*s

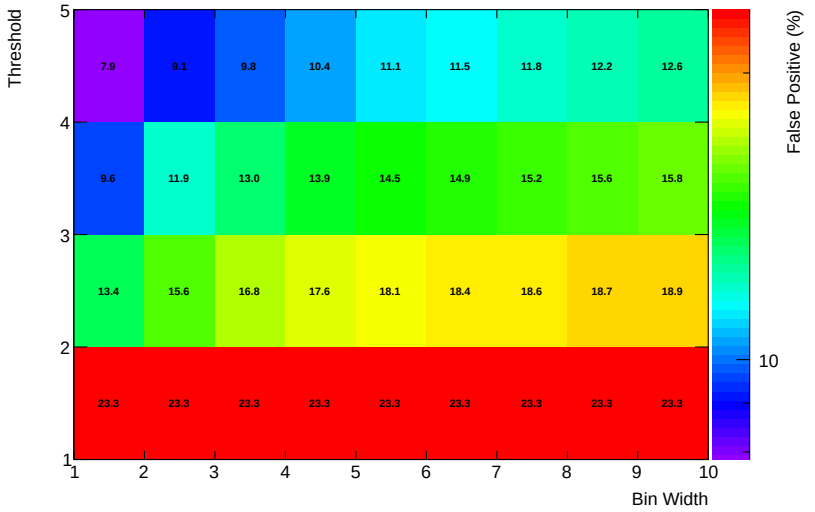


(b) Percentage of *False Positive*s

Figure 4.17: Performances of the Hough transform track recognition algorithm, using only the position of the begin of the clusters. Data-set: run 6945.



(a) Percentage of *False Negative*s



(b) Percentage of *False Positive*s

Figure 4.18: Performances of the Hough transform track recognition algorithm, using the position of the centroid of the clusters. Data-set: run 6945.

Conclusion

The TOTEM experiment at the LHC, thanks to the Roman Pot system, has measured the elastic proton-proton cross-section at the energy of $7\,TeV$ in the center of mass. Dedicated runs with a custom developed optics allowed the measure of the elastic cross-section for $|t|$-values down to $5 \times 10^{-3}\,GeV^2$.

The current DAQ configuration transmits and stores on disk all data acquired by the detectors without any compression or zero suppression. An analysis on real data showed that a huge percentage (up to $\sim 93\%$) of the data transmitted by Roman Pot detectors is rejected by the off-line track recognition software.

In chapter 2 a fast cluster searching algorithm has been proposed. This has been also used to implement a filter on the number of cluster per RP plane. Indeed, a plane without clusters is useless as well as a plane with too many clusters. This filter has been proved to be able to reject up to the $\sim 97\%$ of the RPs that will be rejected by the off-line reconstruction, without rejecting any reconstructable data.

The proposed filter has been implemented in one of the FPGA of the front-end electronics and it has been tested during a dedicated data taking. Its results have been compared with the software implementation with a 100% correspondence. This implementation can be used in the next future for an on-line data reduction without the risk of rejecting reconstructable data.

In chapter 3, two track recognition algorithms were proposed. The first is based on the fact that protons exiting from the Interaction Point have a small angle with the beam. Hence, a track is a series of clusters with the same position for all planes of the same RP detector, of the same projection. A peak in the histogram of the position of the cluster for all the planes is a clear hint for the presence of a track. This method is fast and, in a future hardware implementation, can be implemented to fill the histogram while data are streaming.

The second track recognition algorithm is based on Hough transform, a powerful tool widely used in pattern recognition. This algorithm is more accurate than the previous one and it can be used also to detect track due to protons that are not exiting from the Interaction Point, for instance to estimate the background. However, this method needs to correlate the position of the cluster beetween planes and so, it is compulsory to search for

clusters, store them and, finally, start the track recognition.

The implementation of a filter based on these track recognition algorithms could be a good way to improve the DAQ efficiency for low $\beta*$ runs, where the filter on the number of cluster per plane is not enough. It should be noted, anyway, that the use of these algorithms will require a systematic evaluation of their recognition efficiency. This is not required for the filter on the number of clusters, because the cluster counting has been proved to be 100% efficient.

Appendix A

Real-time application for cluster analysis

An useful application of the cluster searching algorithm is in the DQM (Data Quality Monitor). During data taking it is important to have a real-time application capable of monitoring data quality. Using the designed algorithm is possible to monitor the distribution of the number of clusters per plane for each bunch crossing. Indeed, when the number of clusters per plane is too high, bunch can be noisy and could be excluded from the trigger scheme. In a different way, an high occurence of empty planes could be a sympthom of some problem.

A standalone application has been developed using ROOT framework. This application consists of two threads, one for the cluster analysis and the other to display the results. It is possible to use the application connected to the stream of data while it is being acquired to plot the distribution of *empty* and *high multiplicity* planes, as shown in Fig. A.1. Moreover, it is possible to chose the threshold above which a plane is considered of *high multiplicity* and the refresh interval.

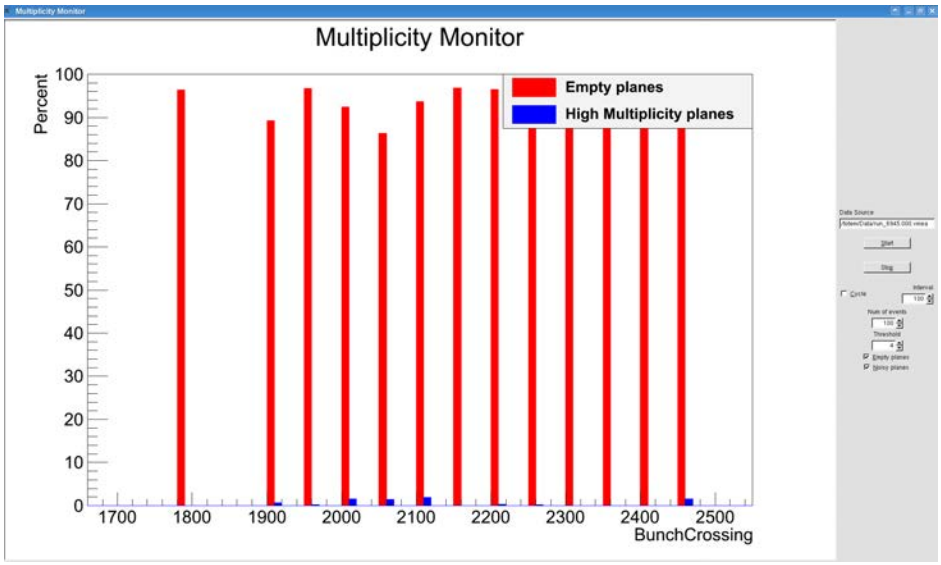


Figure A.1: The application uses the presented cluster analysis algorithm to monitor in real-time the distribution of number of clusters.

List of Figures

1.1 CERN's accelerator complex. 2
1.2 Compilation of total (σ_{tot}), inelastic (σ_{inel}) and elastic (σ_{el}) cross-section measurements [5]. 3
1.3 Graphical representation of the most common event types in $p-p$ collisions. The leftmost pictures are graphical representations of the processes and in the middle typical angular and pseudorapidity distributions are shown. . 5
1.4 The LHC beam line, the TOTEM forward trackers T1 and T2 embedded in the CMS detector and the Roman Pots at 147 m (RP147) and 220 m (RP220). 6
1.5 Left: Detector coverage in the pseudorapidity-azimuth plane. Right: pseudorapidity distribution of charged particle multiplicity and energy flow for generic inelastic collisions at $\sqrt{s} = 14TeV$. 6
1.6 T1 telescope at the test beam facility. The five CSC planes are visible. . . . 8
1.7 Picture of T2 during construction. 9
1.8 In T2's GEM, holes are conical and their diameter goes from 55 μm to 70 μm. 10
1.9 Schematic view of the triple GEM detector used in TOTEM experiment. Electric fields inside T2 chambers are provided by a resistive divider, useful to maintain the right voltage proportions between all the electrodes during power on/off operations. Each foil is also connected with a series resistor to the divider to limit the current in case of discharge. 10
1.10 Position of RP detectors with respect to the impact point. 12
1.11 Cross section of a strip detector. Ionizing particles generate electron-hole pairs that drift to the strips thanks to the applied bias voltage. Strips are AC coupled to the readout electronics by the thin insulating silicon dioxide layer. 13
1.12 Silicon detectors inside each Roman Pot. 13
1.13 When two (or more) hits have been recorded for each projections, it is not possible to uniquely reconstruct the tracks. Indeed, there is no way to choose which pair of circles (empty or full) corresponds to the correct pair of tracks. 14
1.14 Block diagram of the TOTEM electronics system [7]. 14
1.15 Schema of the TOTEM data readout chain. 15

1.16 Block diagram of the VFAT chip. [13] 15
1.17 Picture of the *TOTFed* hostboard with three *OptoRx*s. 17
1.18 Schema of the *TOTFed* hostboard architecture. 17
1.19 Picture of the *OptoRx* card. 18
1.20 Schema of the *Optical Receiver* (*OptoRx*) architecture. 19
1.21 Data format used in TOTEM raw data. 20
1.22 Schema of the arrangement of the VFAT frames in the TOTEM RP system. 20
1.23 Schema of the arrangement of the GOH frames in the TOTEM RP system. . 21
1.24 The structure of the RP simulation and reconstruction software [16]. . . . 22
1.25 For a track parallel to the beam, and so perpendicular to the detector, hits through the 5 silicon planes fall in the same bin of the positions' histogram. 23

2.1 Example of a cluster of size 2 in the VFAT payload. 26
2.2 Working principle of the Look Up Table. 27
2.3 Data format used in TOTEM raw data (See section 1.4.3). 28
2.4 Block diagram of *bitStat* module. 31
2.5 State transition chart of the FSM that generates *beginOfCluster* and *endOfCluster* signals. 32
2.6 State transition chart of the FSM that manages reading procedure of FIFOs. 34
2.7 Block diagram of GohStat module. 34
2.8 Block diagram of *planeStat* module. 35
2.9 Block diagram of OptoRxStat module. 37
2.10 State transition chart of the FSM that generates *newEvent* and *vfatPayload* signals. 38

3.1 Coordinate system for rod search algorithm 40
3.2 Example of an *elastic* track. In Fig. 3.2(a) is shown the hit distributions for the five planes; In Fig. 3.2(b) is shown the histogram filled with hits acquired by all five planes together. The histogram has a bin with 4 entries. 41
3.3 Example of an *non-elastic* track. In Fig. 3.3(a) is shown the hit distributions for the five planes; In Fig. 3.3(b) is shown the histogram filled with hits acquired by all five planes together. The histogram has no bin with more than one entry. 41
3.4 Flow chart of the dynamic binning algorithm. 44
3.5 If the points in the *source space* lay on a straight line (blue points in Fig. 3.5(a)), their Hough transform in the *parameter space* will show intersecting lines (solid blue lines in Fig. 3.5(b)). If the points in the *source space* are randomly chosen (red triangles in Fig. 3.5(a)), their Hough transform in the *parameter space* will show lines that do not intersect in the same point (dashed red lines in Fig. 3.5(b)). 46

4.1 Percentage of accepted RP, *OptoRx*s and events. Red lines shows the percentage of RP, *OptoRx*s and events where the off-line software found a track. Data-set: run 5608. 53
4.2 Percentage of accepted RP, *OptoRx*s and events. Red lines shows the percentage of RP, *OptoRx*s and events where the off-line software found a track. Data-set: run 5657. 54
4.3 Percentage of accepted RP, *OptoRx*s and events. Red lines shows the percentage of RP, *OptoRx*s and events where the off-line software found a track. Data-set: run 6945. 54
4.4 Simplified view of the impact of the parameters on the algorithms. 55
4.5 Size of clusters found in all data set (red) and only in detectors where a track has been recognized by the off-line software (blue). Data-set: run 5608. 57
4.6 Performances of the histogram based track recognition algorithm, using only the position of the begin of the clusters. Data-set: run 5608. 58
4.7 Performances of the histogram based track recognition algorithm, using the position of the centroid of the clusters. Data-set: run 5608. 59
4.8 Performances of the histogram based track recognition algorithm, using only the position of the begin of the clusters. Data-set: run 5657. 60
4.9 Performances of the histogram based track recognition algorithm, using the position of the centroid of the clusters. Data-set: run 5657. 61
4.10 Performances of the histogram based track recognition algorithm, using only the position of the begin of the clusters. Data-set: run 6945. 62
4.11 Performances of the histogram based track recognition algorithm, using the position of the centroid of the clusters. Data-set: run 6945. 63
4.12 Percentage of *False Negative*s using the histogram based track recognition algorithm, using the position of the centroid of the clusters. Only tracks with angles less than 1 *mrad* in both u and v projections have been considered. Data-set: run 5608. 64
4.13 Performances of the Hough transform track recognition algorithm, using only the position of the begin of the clusters. Data-set: run 5608. 65
4.14 Performances of the Hough transform track recognition algorithm, using the position of the centroid of the clusters. Data-set: run 5608. 66
4.15 Performances of the Hough transform track recognition algorithm, using only the position of the begin of the clusters. Data-set: run 5657. 67
4.16 Performances of the Hough transform track recognition algorithm, using the position of the centroid of the clusters. Data-set: run 5657. 68
4.17 Performances of the Hough transform track recognition algorithm, using only the position of the begin of the clusters. Data-set: run 6945. 69

4.18 Performances of the Hough transform track recognition algorithm, using the position of the centroid of the clusters. Data-set: run 6945. 70

A.1 The application uses the presented cluster analysis algorithm to monitor in real-time the distribution of number of clusters. IV

List of Acronyms

ALICE A Large Ion Collider Experiment

AFEC Anode Front-End Card

ASIC Application Specific Integrated Circuit

ATLAS A Toroidal LHC Apparatus

BC Bunch Crossing

CERN European Organization for Nuclear Research

CFEC Cathode Front-End Card

CMS Compact Muon Solenoid

CSC Cathode Strip Chambers

DAQ Data AcQuisition

DQM Data Quality Monitor

EN Event Number

FPGA Field Programmable Gate Array

FSM Finite State Machine

GEM Gas Electron Multipliers

GOH Gigabit Optical Hybrid

I2C Inter Integrated Circuit

IP Interaction Point

LHC Large Hadron Collider

LHCb Large Hadron Collider beauty

LV1A LeVel One trigger Accept

QPLL Quartz crystal based Phase-Locked Loop

ROC Read-Out Card

RP Roman Pot

SRAM Static Random Acess Memory

T1 Telescope 1

T2 Telescope 2

TOTEM TOTal cross section, Elastic scattering and diffraction dissociation Measurement at the LHC

TTC Timing, Trigger and Control

TTS Trigger Throttling System

UHV Ultra High Vacuum

VME VERSABUS Module Eurocard

VFAT Very Forward Atlas and Totem

Bibliography

[1] G Antchev, P Aspell, I Atanassov, V Avati, J Baechler, V Berardi, M Berretti, E Bossini, M Bozzo, P Brogi, E Brücken, A Buzzo, F S Cafagna, M Calicchio, M G Catanesi, C Covault, M Csanád, T Csörgő, M Deile, K Eggert, V Eremin, R Ferretti, F Ferro, A Fiergolski, F Garcia, S Giani, V Greco, L Grzanka, J Heino, T Hilden, M R Intonti, J Kašpar, J Kopal, V Kundrát, K Kurvinen, S Lami, G Latino, R Lauhakangas, T Leszko, E Lippmaa, M Lokajíček, M Lo Vetere, F Lucas Rodríguez, M Macrí, L Magaletti, T Mäki, A Mercadante, N Minafra, S Minutoli, F Nemes, H Niewiadomski, E Oliveri, F Oljemark, R Orava, M Oriunno, K Österberg, P Palazzi, J Procházka, M Quinto, E Radermacher, E Radicioni, F Ravotti, E Robutti, L Ropelewski, G Ruggiero, H Saarikko, A Santroni, A Scribano, W Snoeys, J Sziklai, C Taylor, N Turini, V Vacek, M Vitek, J Welti, and J Whitmore. Measurement of the forward charged particle pseudorapidity density in pp collisions at $\sqrt{s}$ = 7 tev with the totem experiment. *Europhys. Lett.*, 98(arXiv:1205.4105):31002. 7 p, May 2012. Comments: 7 pages, 3 figures.

[2] G. Antchev, P. Aspell, I. Atanassov, V. Avati, J. Baechler, V. Berardi, M. Berretti, E. Bossini, M. Bozzo, P. Brogi, E. Brucken, A. Buzzo, F.S. Cafagna, M. Calicchio, M.G. Catanesi, C. Covault, M. Csanad, T. Csorgo, M. Deile, K. Eggert, V. Eremin, R. Ferretti, F. Ferro, A. Fiergolski, F. Garcia, S. Giani, V. Greco, L. Grzanka, J. Heino, T. Hilden, R.A. Intonti, J. Kaspar, J. Kopal, V. Kundrat, K. Kurvinen, S. Lami, G. Latino, R. Lauhakangas, T. Leszko, E. Lippmaa, M. Lokajıcek, M. Lo Vetere, M. Macrı, T. Maki, A. Mercadante, N. Minafra, S. Minutoli, F. Nemes, H. Niewiadomski, E. Oliveri, F. Oljemark, R. Orava, M. Oriunno, K. Osterberg, P. Palazzi, J. Prochazka, M. Quinto, E. Radermacher, E. Radicioni, F. Ravotti, E. Robutti, F.L. Rodrıguez, L. Ropelewski, G. Ruggiero, H. Saarikko, A. Santroni, A. Scribano, J. Smajek, W. Snoeys, J. Sziklai, C. Taylor, N. Turini, V. Vacek, M. Vıtek, J. Welti, and J. Whitmore. Measurement of proton-proton elastic scattering and total cross-section at *sqrt* s = 7 tev. oai:cds.cern.ch:1472948. (CERN-PH-EP-2012-239), Aug 2012.

[3] Georges Aad and ... Abajyan. Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. oai:cds.cern.ch:1471031.

Phys. Lett. B, XX(arXiv:1207.7214. CERN-PH-EP-2012-218):XX. 39 p, Aug 2012. Comments: 24 pages plus author list (39 pages total), 12 figures, 7 tables, submitted to Physics Letters B.

[4] Serguei Chatrchyan and ... Khachatryan. Observation of a new boson at a mass of 125 gev with the cms experiment at the lhc. oai:cds.cern.ch:1471016. *Phys. Lett. B*, XX(arXiv:1207.7235. CMS-HIG-12-028. CERN-PH-EP-2012-220):XX. 59 p, Jul 2012. Comments: Submitted to Phys. Lett. B.

[5] G. Antchev, P. Aspell, I. Atanassov, V. Avati, J. Baechler, V. Berardi, M. Berretti, E. Bossini, M. Bozzo, P. Brogi, et al. First measurement of the total proton-proton cross-section at the lhc energy of. *EPL (Europhysics Letters)*, 96:21002, 2011.

[6] J. Kaspar. *Elastic scattering at the LHC.* PhD thesis, Faculty of Mathematics and Physics, Charles University in Prague, 2011. Chapter 1.

[7] G. Anelli, G. Antchev, P. Aspell, V. Avati, MG Bagliesi, V. Berardi, M. Berretti, V. Boccone, U. Bottigli, M. Bozzo, et al. The totem experiment at the cern large hadron collider. *Journal of Instrumentation*, 3:S08007, 2008.

[8] F. Sauli. Gem: A new concept for electron amplification in gas detectors. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 386(2):531–534, 1997.

[9] C. Altunbas, M. Capéans, K. Dehmelt, J. Ehlers, J. Friedrich, I. Konorov, A. Gandi, S. Kappler, B. Ketzer, R. De Oliveira, et al. Construction, test and commissioning of the triple-gem tracking detector for compass. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 490(1):177–203, 2002.

[10] E. Oliveri. *The forward inelastic telescope T2 for the TOTEM experiment at the LHC.* PhD thesis, Department Of Physics, University of Siena, 2010.

[11] U. Amaldi, R. Biancastelli, C. Bosio, G. Matthiae, JV Allaby, W. Bartel, G. Cocconi, AN Diddens, RW Dobinson, and A.M. Wetherell. The energy dependence of the proton-proton total cross-section for centre-of-mass energies between 23 and 53 gev. *Physics Letters B*, 44(1):112–118, 1973.

[12] G. Ruggiero, E. Alagoz, V. Avati, V. Bassetti, V. Berardi, V. Bergholm, V. Boccone, M. Bozzo, A. Buzzo, MG Catanesi, et al. Planar edgeless silicon detectors for the totem experiment. *Nuclear Science, IEEE Transactions on*, 52(5):1899–1902, 2005.

[13] P. Aspell, G. Anelli, P. Chalmet, J. Kaplon, K. Kloukinas, H. Mugnier, W. Snoeys, and PH-EP. Vfat2: A front-end system on chip providing fast trigger information, digitized data storage and formatting for the charge sensitive readout of multi-channel silicon and gas particle detectors. *Proc. of the Topical Workhop on Electronics for Particle Physic (TWEPP2007), Prague, Czech Republic*, 2007.

[14] System verilog web page. `http://www.systemverilog.org/`.

[15] V. Avati, M. Berretti, M. Besta, E. Brücken, P. Dadel, F. Ferro, F. Garcia, S. Giani, L. Grzanka, J. Hallila, P. Janhunen, J. Kašpar, G. Latino, R. Leszko, D. Mierzejewski, H. Niewiadomski, T. Novak, T. Nuotio, E. Oliveri, K. Österberg, F. Oljemark, S. Sadilov, M. Tuhkanen, T. Vihanta, M. Zalewski, Z. Zhang, and J. Welti. Offline Software for the TOTEM Experiment at the LHC. In C. Leroy, P.-G. Rancoita, M. Barone, A. Gaddi, L. Price, and R. Ruchti, editors, *Astroparticle, Particle and Space Physics, Detectors and Medical Physics Applications*, pages 658–665, April 2010.

[16] J. Kaspar. *Elastic scattering at the LHC.* PhD thesis, Faculty of Mathematics and Physics, Charles University in Prague, 2011.

[17] H. Niewiadomski. *Reconstruction of protons in the TOTEM roman pot detectors at the LHC.* PhD thesis, Faculty of Engineering and Physical Sciences, University of Manchester, 2008.

[18] E. Gamma, R. Helm, R. Johnson, J. Vlissides, et al. *Design patterns*, volume 1. Addison-Wesley Reading, MA, 2002.

[19] P.V.C. Hough. Method and means for recognizing complex patterns, December 18 1962. US Patent 3,069,654.

[20] Cms collaboration, "timing, trigger and control interface". `http://ttc.web.cern.ch/TTC/`.

[21] P. Aspell. Vfat2-digital specification (version5). *ReH*, 101(2):2, 2005.

[22] C. Augier, J. Bourotte, M. Bozzo, A. Bueno, R. Cases, F. Djama, M. Haguenauer, V. Kundrat, M. Lokajíček, G. Matthiae, et al. Predictions on the total cross section and real part at lhc and ssc. *Physics Letters B*, 315(3):503–506, 1993.