Development and Deployment of DATCON, an FPGA-based Data Reduction System for the Belle II Pixel Detector

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von
Bruno Deschamps
aus
Suresnes, Frankreich

Bonn, April 2022

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter:	Prof. Dr. Jochen Christian Dingfelder
2. Gutachter:	Prof. Dr. Florian Bernlochner
Tag der Promotion:	13.04.2022
Erscheinungsjahr:	2022

Abstract

The SuperKEKB collider located in Tsukuba, Japan, will have a luminosity 40 times higher than its predecessor. With an expected occupancy of 3 %, largely dominated by beam backgrounds, the new pixel DEPFET detector (PXD) produces a theoretical data rate of about 20 GBps. To comply with the Belle II Data Acquisition System, the data rate produced by PXD has to be reduced. By using the hits of the surrounding layers of Silicon-strip Vertex Detectors (SVD), particle trajectories can be reconstructed to create so-called Regions of Interest (ROIs) on the pixel detector. Only the pixels inside a ROI are saved, therefore reducing the amount of PXD data. The DATCON, developed and deployed in the scope of this thesis, contributes to the data reduction. It uses the Hough Transformation technique to extract track parameters and to calculate the intersection of a track with the PXD sensors. The system is built around the μTCA standard and requires two types of Advanced Mezzanine Cards (AMC), each equipped with a Field Programmable Gate Array (FPGA). 52 optical links connect the SVD to the DATCON system and each data stream passes through a pre-processing and multiplexing step before being used for track finding. Two 2D track findings are performed to extract parameters in the $r - \phi$ and $r - z$ planes. Their results are merged during ROI creation.

The performance of the DATCON algorithm was evaluated with a series of simulations starting at the most simple case of a single track but also with physics data. The most important specifications that characterize DATCON performance are the Data Reduction Factor (DRF) and the ROI Finding Efficiency (RFE). The two are closely related, and parameters of the algorithm can be tuned to optimize them. The ROIs produced based on Belle II data are compared with the ones produced by the High Level Trigger (HLT), bringing the need for extensive analysis.

For $\Upsilon(4S)$ events, the overall data reduction factor of 3.5 and a ROI finding efficiency above 85 %, DATCON is not yet sufficient for standalone data reduction. However, it was shown in this thesis, that an FPGA-based, online data reduction works and has potential for further optimization.

Contents

1 Introduction

From time immemorial, mankind has sought to understand the universe and its constituents. Modern particle physics offers a structural interpretation of matter to the finest level, also known as the Standard Model (SM) of particle physics. The model includes all known particles, divided in three families of quarks and leptons. It also describes interactions between elementary particles with the gauge bosons acting as mediators between the weak, the strong and the electromagnetic forces. Gravity does not appear in the SM. The particle spectrum was completed with the discovery of the Higgs boson in 2012 [1] which is responsible for the masses of the elementary particles.

In the first second after the after the Big Bang, i.e. before the hadron epoch, an equal amount of matter and antimatter was created. However, everything we see in the universe today is mostly made of ordinary matter. This asymmetry suggests, that a difference of behavior exists between particles and antiparticles that should annihilate each other and produce only radiation. Such a difference can happen under certain conditions, one of which is the violation of the charge-parity ($\mathcal{CP}$)-symmetry as described by Sakharov [2] and discovered in 1964 by Cronin and Fitch, in the decay of neutral kaons [3]. Almost ten years later, Kobayashi and Maskawa proposed an explanation for the ($\mathcal{CP}$) violation by introducing a third family of quarks and the flavor-mixing CKM matrix [4].

By studying the decay of B mesons produced by dedicated e^-/e^+ colliders, also known as B factories, precise measurements can be made to validate the theories for origin of the ($\mathcal{CP}$) violation. Contrasting to the extreme energy colliders used to find new heavy particles, such as the Large Hadron Collider (LHC), the B factories rely on high luminosity to record huge samples of B mesons.

($\mathcal{CP}$) violation was discovered by B factories such as the KEKB collider[5] located at High Energy Accelerator Research Organization (KEK) in Japan with the Belle experiment [6] and the PEP-II collider with the Babar detector [7].

Despite the success of Belle, the need to discover physics beyond the SM requires a profuse amount of data which can only be provided by further increasing the luminosity. KEKB and Belle were therefore upgraded to SuperKEKB and Belle II .

One of the modifications made for the Belle II detector was the addition of a newly developed pixel detector Pixel Detector (PXD) which constitutes the innermost part of the detector. With a spacial resolution of a few micrometers, the new PXD can detect track vertices even

in the high particle rate environment at Belle II . The theoretical data rate of the PXD is approx 20 GBps with an occupancy of 3 %, largely dominated by beam backgrounds. Because the amount of data produced, by far exceeds the bandwidth limitation of the storage system, a data reduction is mandatory.
In this thesis, the Data Acquisition Tracking and Concentrator Online Node (DATCON), a Field Programmable Gate Arrays (FPGA)-based reduction system for PXD is developed and deployed. Using the hit information obtained from the surrounding SVD only, track finding is performed as well as extrapolation back onto the PXD. So-called Region of Interest (ROI) are then built around the intersections of the extrapolated tracks with the PXD sensors. Only the PXD hits inside ROIs are saved while the rest, assumed to result from background, are rejected. The goal is to achieve a data reduction by a factor of 10. This thesis is the continuation of Michael Schnell's work which laid the foundation of DATCON [8].
This thesis is divided in the following blocks:

- Overview of the Belle II detector and its sub-detectors with particular interest on the PXD and SVD systems. The specifications about the dedicated PXD Data Acquisition (DAQ) are detailed.
- Specifications of the DATCON hardware and its installation at KEK.
- Introduction to the methods used for track finding and extrapolation to the PXD sensors.
- Implementation of the different firmware, from the processing of the SVD hits to the track finding and ROI creation.
- Performance studies of the algorithm with simulation and Belle II data.

2 SuperKEKB and BelleII

In this chapter the SuperKEKB accelerator and the Belle II detector are introduced. The first part focuses on the SuperKEKB accelerator and the differences to its predecessor. The second part concentrates on the Belle II detector and gives an overview of its sub-detectors, with a particular focus on the SVD and PXD. Finally, the PXD and the Belle II data acquisitions are briefly presented, concluding on the need for data reduction. The information provided in this section is largely taken from [9].

2.1 The SuperKEKB Accelerator

The SuperKEKB, collider located at KEK, in Tsukuba, Japan, is a e^-/e^+ collider and the upgrade of the previous KEKB. The KEKB accelerator with its Belle experiment ran until end of operation in the summer of 2010. The new collider relies on asymmetric beams energies providing a Lorentz boost in the center-of-mass of $\beta\gamma = 0.28$. Together with a new vertex detector design, the boost, which is lower than for KEKB, allows for a better identification of the B-mesons. The SuperKEKB collider is composed of two rings, each with a circumference of 3016 meters. In the first one, the High Energy Ring (HER), electrons with an energy of 7 GeV are stored, whereas the Low Energy Ring (LER) stores positrons at an energy of 4 GeV. Beam currents of 3.60 A and 2.62 A are used for the LER and HER, respectively.
In addition, an injector Linear Accelerator (LINAC) with a 1.1 GeV Damping Ring (DR) is used. The linac is not only used for the LER and HER but also for the Photon Factory (PF) and the PF Advanced Ring (PF-AR) at KEK. To avoid any interference between the different storage rings, the linac uses simultaneous top-up injections at a rate of 50 Hz.
An overview of the accelerator structure is shown in fig. 2.1. While the tunnel and many components of KEKB were reused, the DR was completely redesigned to assure low-emittance positron beam injection. For the production of the electrons, and due to cost and space limitation, a high current RF gun is used. As for the positrons, they are produced by firing the electrons onto a tungsten target with a 14 mm thickness.
SuperKEKB has a luminosity increased by a factor of 40 compared to its predecessor and is today the collider with the highest luminosity in the world with a targeted peak luminosity of $8 \times 10^{35} cm^{-2} s^{-1}$.

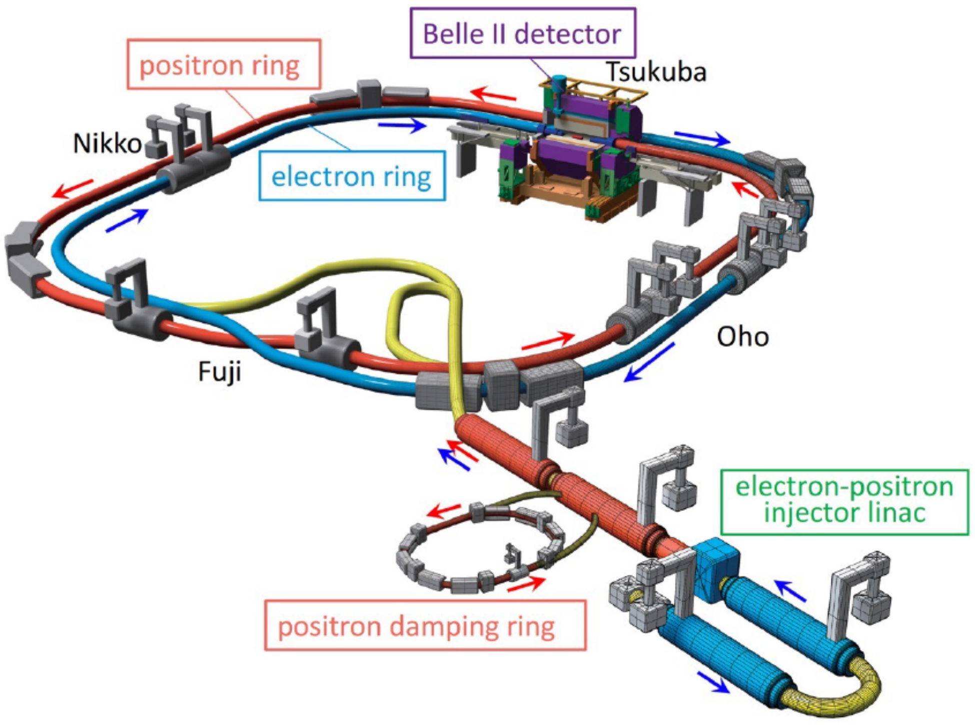


Figure 2.1: Illustration of the SuperKEKB accelerator complex. Electrons and positrons are produced by the injector linac and ramping ring before being stored in the LER and HER.

2.1.1 Luminosity and Nano-beams Scheme

To achieve such a luminosity, for the first time at a particle collider, the "Nano-Beam" scheme is used. It was firstly introduced for the SuperB factory in Italy [10]. The idea behind the nano-beam technique is to reduce the overlapping area of the colliding bunches at the Interaction Point (IP). The vertical beta function β_y^* is compressed by reducing the longitudinal size of the overlap region of the two beams. Figure 2.2 shows a view of the nano-beam collision scheme. The overlapping region is expressed by its size d also known as the effective bunch length which is considerably smaller than the bunch length itself (σ_z). The length d can be expressed as follow:

$$d \approx \frac{\sigma_x^*}{\phi} \tag{2.1}$$

where ϕ is the half crossing angle of the two beams and σ_x^* the horizontal beam spread. Intending to reduce d to 200 μm, a large ϕ of 41.5 mrad is necessary as well as small emittances and beta functions for both beams. The luminosity is computed with the following equation:

$$L = \frac{\gamma_\pm}{2er_e}\left(\frac{I_\pm \xi_{y\pm}}{\beta^*_{y\pm}}\right)\left(\frac{R_L}{R_{\xi y}}\right) \tag{2.2}$$

where the Lorentz factor, the electron classical radius and the elementary electric charge are represented by γ, r_e and e, respectively. Beams are assumed to be flat and of equal size. The subscript $\pm$ differentiates the electron or the positron beam. Finally, R_L and $R_{\xi y}$ correspond to the reduction factor for the luminosity and the vertical beam-beam parameter. Their ratio is normally being close to unity, only the total beam currents $I_\pm$, the vertical beam-beam parameter $\xi_{y\pm}$ and the vertical beta function $\beta^*_{y\pm}$ are essential. Compared to KEKB, the half

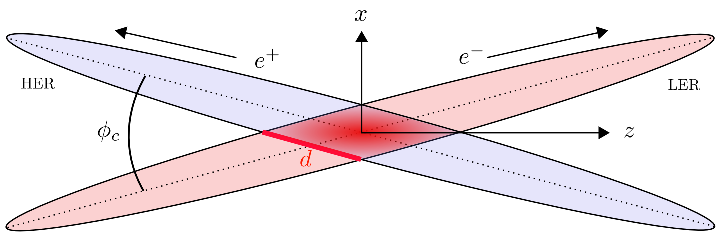


Figure 2.2: Illustration of the nanobeam collision scheme. The magnitude of the beta function at the interaction point is limited by d compared to the much larger bunch length

crossing angle is about four times larger whereas the beta function is reduced by a factor 20. To reach the desired luminosity, the beam currents had to be increased. The various parameters are summarized in table 2.1 and compared the ones previously used for KEKB.

	KEKB	SuperKEKB
Energy$\pm$ [GeV]	3.5/8	4.0/7.0
$I_\pm$ [A]	1.64/1.19	3.60/2.62
$\beta_{y\pm}$ [mm]	5.9/5.9	0.27/0.41
$\xi_{y\pm}$	0.129/0.090	0.090/0.088
Luminosity [$10^{35} cm^{-2} s^{-1}$]	0.21	8.0

Table 2.1: Essential parameters of KEKB and SuperKEKB.

2.2 The Belle II Detector

The Belle II detector, located at the interaction point of the SuperKEKB collider, is an upgrade of the Belle experiment. It has been adapted to cope with the luminosity increase

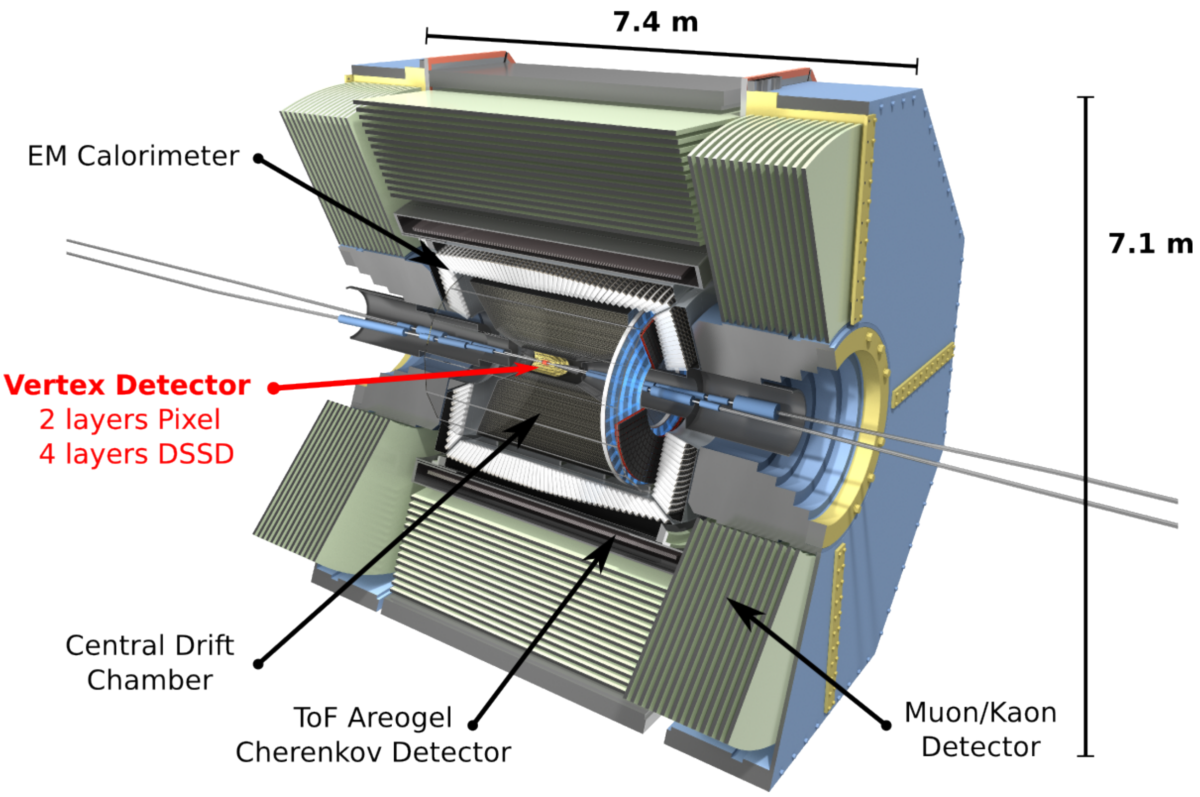


Figure 2.3: Cutaway rendering of the he Belle II detector. From the interaction point to outward the following sub-detectors are found: the pixel detector PXD, the strip detector SVD, the central drift chamber CDC, the particle identification TOP and ARICH, the solenoid coil and finally the muon and kaon detector KLM.

that has come with the SuperKEKB accelerator. Belle II is a multi-layer detector including tracking components as well as particle-identification and energy measurements capabilities. The Belle II detector measures and detects the type, charge, energy, momentum and trajectories of particles originating from the interaction point. Several sub-detectors are part of Belle II which, starting from the interaction point, are two layers of pixelated silicon sensors (PXD) surrounded by four layers of Double-Sided Strip Detector (DSSD) known as SVD. PXD and SVD together form the Vertex Detector (VXD). A CDC measures trajectories, momenta and dE/dx of charged particles. An array of TOP counters, arranged in a barrel shape, can reconstruct the ring-image of Cherenkov light cones emitted by particles passing through the quartz radiators bars. An additional Cherenkov counter (ARICH), with aerogel radiators is installed in the forward end-cap. The last layer inside the solenoid coil that generates a 1.5 Tesla magnetic field, is the electromagnetic calorimeter (Electromagnetic Calorimeter (ECL)) made of scintillator crystals. Outside of the coil, the KLM is used to identify K_L^0 and μ.

2.2.1 Pixel Detector (PXD)

Located at 14 and 22 mm from the IP, two layers of DEpleted p-channel Field Effect Transistor (DEPFET) PXD sensors are installed. DEPFET is a technology introduced in 1987 by Kemmer and Lutz [11]. It consists of a semiconductor detector concept combining detection and amplification in the same silicon substrate. The PXD is designed to resist an annual radiation dose of 2 Mrad with a 3% occupancy. The low PXD material budget of $0.2\% X_0$, necessary for low momentum particle detection, is achieved by thinning down the silicon of the sensitive area to 75 μm.

The DEPFET Sensor

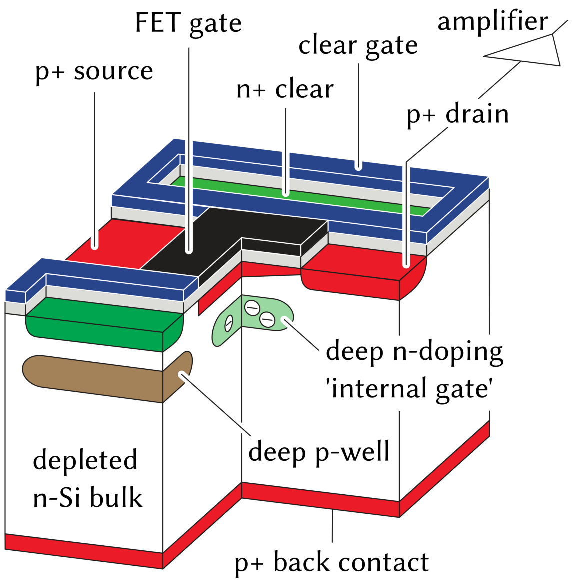


Figure 2.4: Illustration of a DEPFET pixel cell.

The DEPFET sensor is built using a p-channel Metal Oxide Field Effect Transistor (MOSFET) located in a fully depleted silicon bulk. In a P-FET, the current between source and drain is controlled by the gate potential. At a depth of about 1 μm under the transistor, a potential minimum is formed by sideward depletion thanks to an additional phosphorus implantation. When a charged particle passes through, electron-hole pairs are created in the fully depleted bulk. Their number depends on the deposited energy. The holes then drift to the back contact while the electrons are accumulated in the potential minimum also referred as the Internal Gate. Signal electrons within the Internal Gate modulates the source-drain current of the transistor. Fluctuation of this current is measured over the drain contact. After read-out of the drain current signal, the Internal Gate has to be cleared of electrons to get ready for the next charge-collection period. A Clear gate is implemented as a strongly n-doped contact to which positive voltage is applied and creates a path for the electrons to be evacuated. A DEPFET pixel cell is depicted in fig. 2.4

The DEPFET Module

The DEPFET pixel cells are arranged in a matrix with a dimension of 250×768 which is hereafter designated as a module. Two modules form a so-called ladder. Although the matrix dimension is the same for all modules, the pixel size depends on the layer and on the angular region. Each module has 512×250 large pixels and 256×250 small ones. The larger are close to the readout Application Specific Circuit (ASIC) used to steer and read out the sensor. A view of a PXD module is depicted in fig. 2.5. In order to speedup the reading process, a four-fold

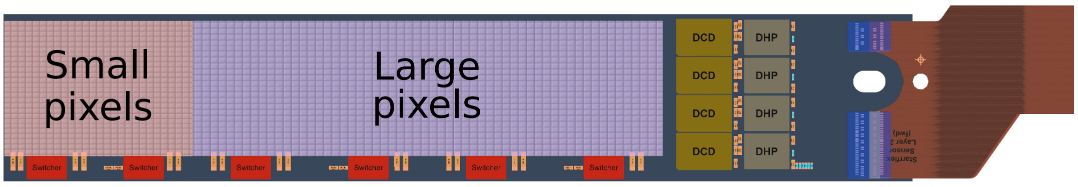


Figure 2.5: Illustration of a PXD module. Steering the Gate and Clear lines, the switchers are placed all along the module. The DCD and DHP chips process the pixel data before sending it to the back-end electronic via the flexible kapton.

readout scheme is introduced where Gate and Clear lines of four rows are connected together requiring only 192 drivers. Since a single pixel signal must be accessible, four drain lines are routed to the ASICs for every column, making a total of a thousand drain lines as depicted in fig. 2.6 . A drain-gate line pair is sufficient to identify a pixel.
The reading of a matrix is based on the so called *rolling shutter* scheme. For the reading process, a gate line is switched to a low potential to turn the DEPFET transistor into an 'on' state while the others are being kept off. After the drain current has been readout and compared to a previously recorded pedestal, transistors are cleared by setting up the Clear line. Gates are then brought back to an off state so that the next measurement can be made. In addition to the DEPFET matrix three different type of **ASICS!** (**ASICS!**)s are needed for the readout, switchers, DCDs and DHPs which are mounted directly on the silicon die. Their function is described as follow:

- A balcony with six **Switchers** are responsible to steer the Gate and Clear lines. Each Switcher can handle up to 32 of these 192 lines.
- Four **Drain Current Digitizer (DCD)** are placed at the bottom of the module can be found. Each DCD has 256 inputs to digitize the Drain currents. Every channel is equipped with an 8-bits Analog-to-Digital Converter (ADC). With an output rate of approximately 20 Gbps, one more ASIC is needed to lower that rate.

- The **Data Handling Processor (DHP)** reduces the bandwidth by performing common-mode correction, pedestal subtraction and data reduction via zero suppression. A single Kapton Printed Circuit Board (PCB), which includes 24 voltage lines, carries the processed data to the back-end electronics via 1.5 Gbps link.

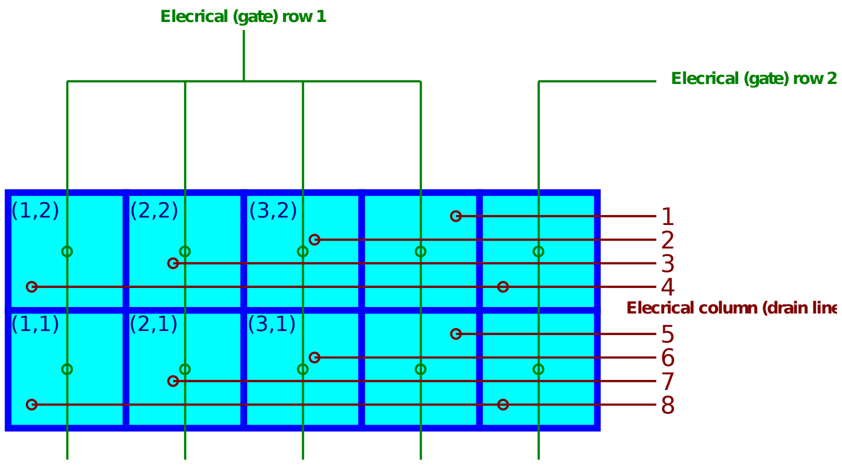


Figure 2.6: Illustration of the rolling shutter readout scheme. Pixels are addressed by (r, c) with r and c being the row and column number, respectively. The Gate signal is applied to four rows (green) and the drain is connected to every fourth pixel of a column. From [12]

.

PXD Overview

A PXD ladder is formed by two modules glued together and reinforced by ceramic joints and a self-supporting structure. The first PXD layer, located at 14 mm from the interaction point consists of eight ladders while the second layer, at 22 mm has twelve ladders. Each ladder is rotated by a specific angle and shifted in the local x axis, therefore creating a so-called windmill structure. With this geometry where a ladder overlaps with its neighbor, gaps are avoided. The PXD has an acceptance of 2π in the azimuth angle and covers polar angles from $17°$ in the forward region to $150°$ in the backward. The dimensions of the sensors and pixels for each layers are given in table table 2.2.

2.2.2 Silicon Vertex Detector (SVD)

The SVD, positioned around the PXD, is a DSSD that covers the same azimuth and polar angle range as the PXD. New sensors and ASICs allow for faster readout and improved timing resolution compared the strip detector of Belle. To cope with the forward boost due to the asymmetric beam energies and to avoid a large number of wafers of a purely cylindrical

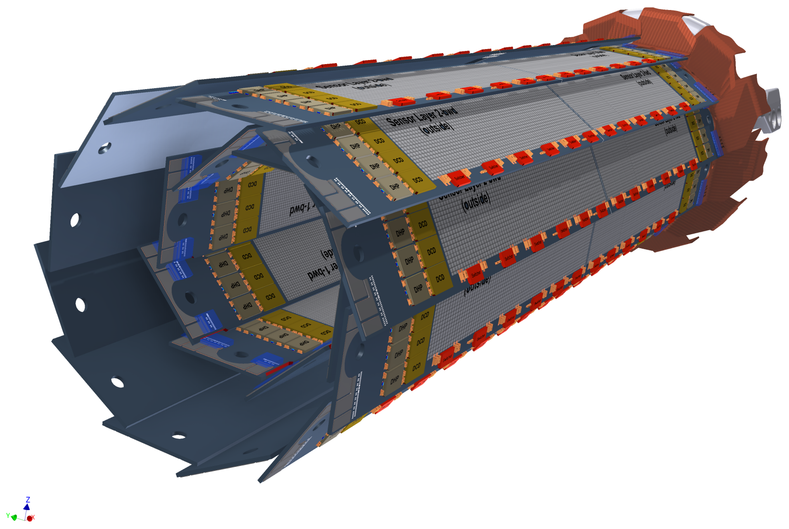


Figure 2.7: 3D view of the Belle II pixel detector. The two layers of ladders, composed of two PXD modules each, are rotated and shifted thus creating the windmill structures.

	Layer 1	Layer 2
Number of ladders	8	12
Radius [mm]	14.21	21.79
Ladder dimensions [mm^2]	15.4×67.975	15.4×84.975
Active area [mm^2]	12.5×44.8	12.5×61.44
Pixel pitch [μm^2]	50×55 50×70	50×60 50×85
Number of pixels	256×768	256×768
Thickness [μm]	75	75

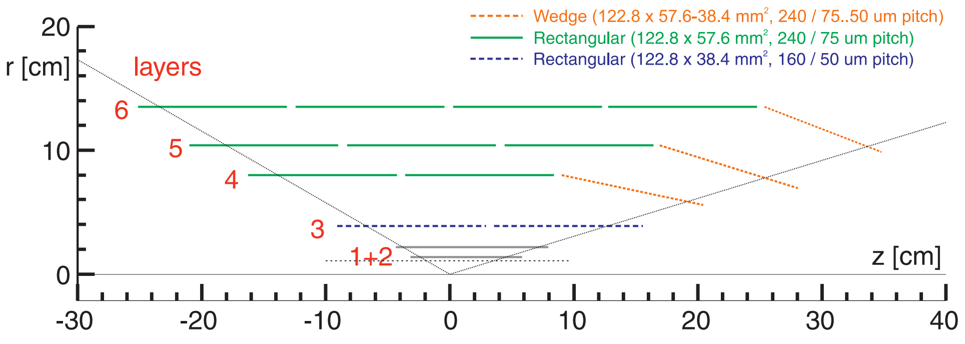
Table 2.2: Module parameters for each PXD layer.

construction, the characteristic lantern shape chosen with slanted sensors in the forward region. Each sensor is built from a high resistivity 300 μm N-type bulk. Depending on the side and the orientation, the strips implanted with P-type implantation are called $\mathcal{P}$ strips and are perpendicular to the $r-\phi$ plane. The N-type implantation is used for the $\mathcal{N}$ strips which are perpendicular to the z-axis.

Detector Layout

From the beam line, towards the outside of Belle II , the four SVD layers are located at radii of 38 mm, 80 mm, 105 mm and 135 mm as shown in fig. 2.8.
After layer 1 and 2 of PXD, the SVD layers are layers 3, 4, 5 and 6. They consist of 7, 10,

12 and 16 longitudinal modules,respectively, also called ladders, which are parallel to the z-axis. There three types of SVD modules, all DSSD, but with different strip pitch and numbers.



Wedge (122.8 x 57.6-38.4 mm², 240 / 75..50 um pitch)
Rectangular (122.8 x 57.6 mm², 240 / 75 um pitch)
Rectangular (122.8 x 38.4 mm², 160 / 50 um pitch)
r [cm]
layers
6
5
4
3
1+2
z [cm]
20
10
0
-30
-20
-10
0
10
20
30
40

Figure 2.8: Schematic configuration showing the three different sensor geometries

- On layer 3, two small rectangular sensors with a 320 μm thickness are installed and both have 768 strips on the $\mathcal{P}$ and the $\mathcal{N}$ side with a pitch of 50 μm and 160 μm, respectively.

- In the barrel region, the large rectangular sensors of layers 4, 5 and 6 are based on larger DSSD with 768 $\mathcal{P}$ strips with a pitch of 75 μm and 512 $\mathcal{N}$ strips with a 240 μm pitch.

- Finally, the forward slanted sensors mounted at an angle with respect to the z-axis, unlike the rest of the modules, do not have a rectangular shape but are trapezoidal. With the same number of strips as as the barrel sensors, the pitch on the $\mathcal{N}$ side varies from 75 to 50μm.

The total active area of SVD is about 1.1 m^2 for which, 1748 readout chips are needed. With a material budget of 0.7% of the radiation length, SVD has been designed to tolerate an integrated radiation dose of 10 MRd, more than expected during the Belle II lifetime. The sensor specifications are listed in table 2.3.

	Small rectangular	Large rectangular	Trapezoidal
Number of $\mathcal{P}$ strips	768	768	767
Number of $\mathcal{N}$ strips	768	512	512
$\mathcal{P}$ strips pitch [μm]	50	75	50-75
$\mathcal{N}$ strips pitch [μm]	160	240	240
Sensor active area [mm^2]	122.90×38.55	122.90×57.72	122.76×(38.42-57.59)

Table 2.3: Specifications of the three types of SVD sensors.

The APV25

The readout of the strips is done using the APV25 chip. This chip, originally developed for the Compact Muon Solenoid (CMS) tracker at the LHC [13], is an analog pipeline ASIC preprocessing the strip signals before DAQ. The chip is highly susceptible to noise due the capacitive load of the amplifier inputs. For this reason, the chip has to be placed as close as possible to the sensor strip. In order to obtain a sufficient Signal-to-Noise Ratio (SNR), all sensors have to be read out individually while minimizing the material budget. To address this issue for the central sensors, the so-called *Origami chip-on-sensor* concept where the APV25 chips are mounted on a flexible Kapton, connected on the $\mathcal{N}$ side of the module, is used. The sensors on the edges use conventional hybrids where the chips are mounted on an dedicated PCB and are connected to the strips via cables and connectors.

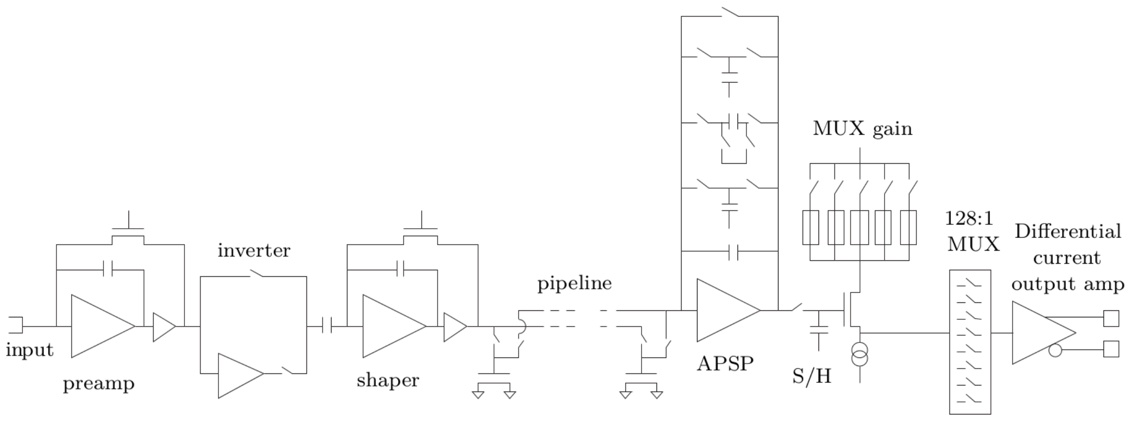


Figure 2.9: Block diagram of a single APV25 channel.

The readout chip consists of 128 channels of low-noise preamplifiers with 250 nm Complementary Metal Oxide Semiconductor (CMOS) technology. It is directly followed by a CR-RC shaper stage. The shaped signal is then stored in a 192-cell deep analog pipeline. On trigger request, signals can then be processed by a pulse shape processor (APSP) or simply forwarded into a multiplexer, the output being driven through a final differential output driver to an ADC. The blocks arrangement of the APV25 is shown in fig. 2.9. The APV25 can operate either in 3 or 6-samples mode which means that 3 or 6 samples can be read out of the ring buffer on trigger request. From the samples, the hit time and peak amplitude can later be extracted. Every strip is connected to one of the 128 inputs of the APV25.

Flash Analog to Digital Converter (FADC)

The role of the FADC is to digitize the signals coming from up to 48 AVP25 ASICs. Every APV25 differential output is de-serialized and digitized with a 10-bit precision (-511,511). To compensate non-linear transfer functions and to remove reflections due to long cables, the signal is processed by a Finite Impulse Response filter (FIR) with eight coefficients. The average amplitude of the strips is subtracted through two stages of Common-Mode Correction. Finally, zero suppression is applied to keep only strips with at least one sample higher or equal to three times the recorded noise level. The samples are finally converted to unsigned 8-bits values. The FADC system is build around 52 FADC units shared over four crates positioned on top of Belle II . Each FADC is a 9U VME card embedding a powerful FPGA.

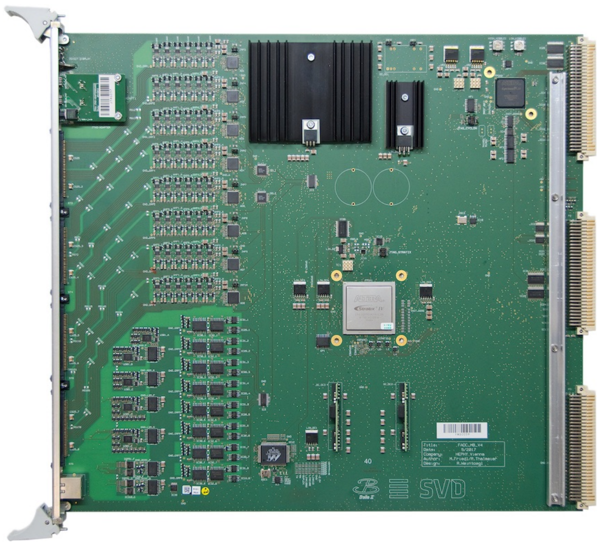


Figure 2.10: FADC board.

Finesse Transmitter Board (FTB)

On every FADC is connected one FTB whose purpose is to transmit FADC data to the Common Pipeline Platform for Electronics Readout (COPPER) and to the DATCON system. The FTB is built around a Spartan-6 FPGA from Xilinx whose task is to convert the digital FADC data to optical. The so-called belle2link [14] is implemented and duplicated for the two outputs. The Aurora protocol, detailed in section 4.3.1, is used for the data transmission based on a rate of 1.27 Gbps which is sufficient to avoid data buffering. The part of the firmware dedicated to DATCON is detailed in Chapter 5.

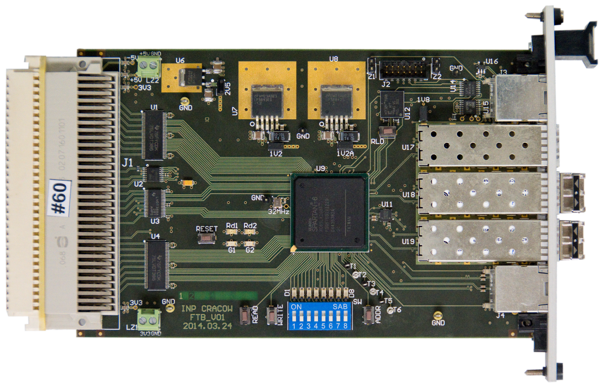


Figure 2.11: FTB board.

2.2.3 Background Contributions Affecting the VXD

The high luminosity of SuperKEKB leads to large beam-related backgrounds levels and the VXD needs to cope with high radiation on particle rates. As it is closest to the interaction point, the PXD sees the highest particle density of all sub-detectors. The total occupancy is dominated by background, requiring a data reduction approach as the one introduced in this thesis. Background sources in SuperKEKB can be separated into two main groups: single beam and luminosity induced backgrounds. The first one includes the following sources:

- Touschek background: Charged particle within a bunch tend to oscillate. Intra-bunch scattering of two electrons or two positrons make them leave the bunch and hit the beam pipe. Resulting particle showers can reach the detectors and produce background hits.
- Beam-gas scattering: Stored particles can collide with residual gas atoms in the beampipe which trough Coulomb scattering or Bremsstrahlung causes energy loss of the beam particles.
- Synchrotron radiation: Electromagnetic radiation, emitted by a charged particle moving in a magnetic field that deflects its trajectory. The synchrotron radiation that is seen in the detector are caused by the focusing and bending magnets.

The second group, the luminosity-related backgrounds are results of beam collision and their most important contributions are :

- Bhabha scattering: Scattering process between an electron and a positron. In radiative Bhabha scattering, at least on photon is emitted by the process $e^+e^- \rightarrow e^+e^- + \gamma$. Scattered particles leaving the interaction region with a small angle and hitting the beam pipe create secondaries that can produce hits in the PXD sensors. Bhabha scattering is considerably responsible for the background in the Belle II detector.

- Two photons process. The overall largest contribution of the background in the PXD. This QED process $e^+e^- \rightarrow e^+e^- + (\gamma\gamma) \rightarrow e^+e^-e^+e^-$ causes the production of extremely low momentum e^+e^- pairs that curls around the z-axis and reaches the PXD but not the outer detectors.

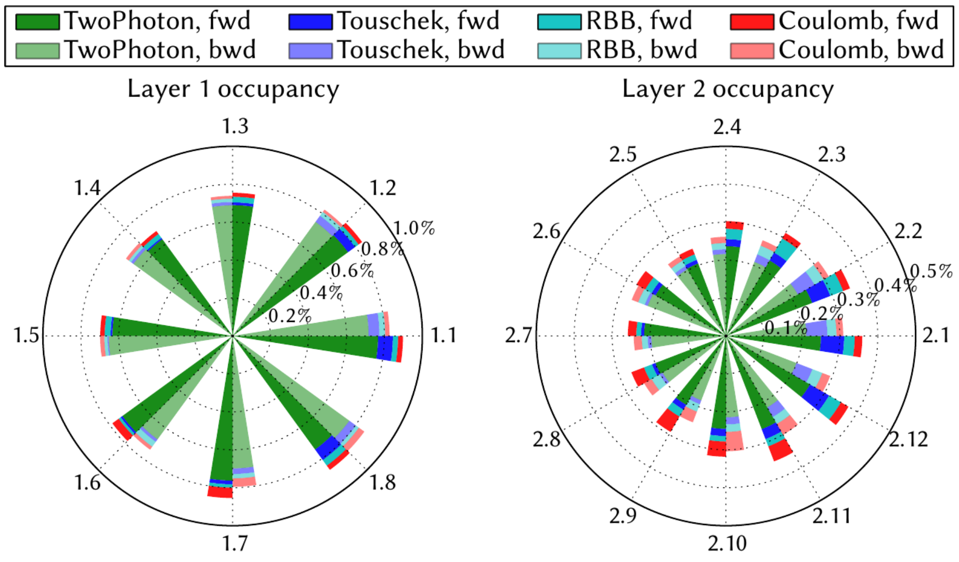


Figure 2.12: Different background contributions in the PXD, with distinction between backward and forward region.

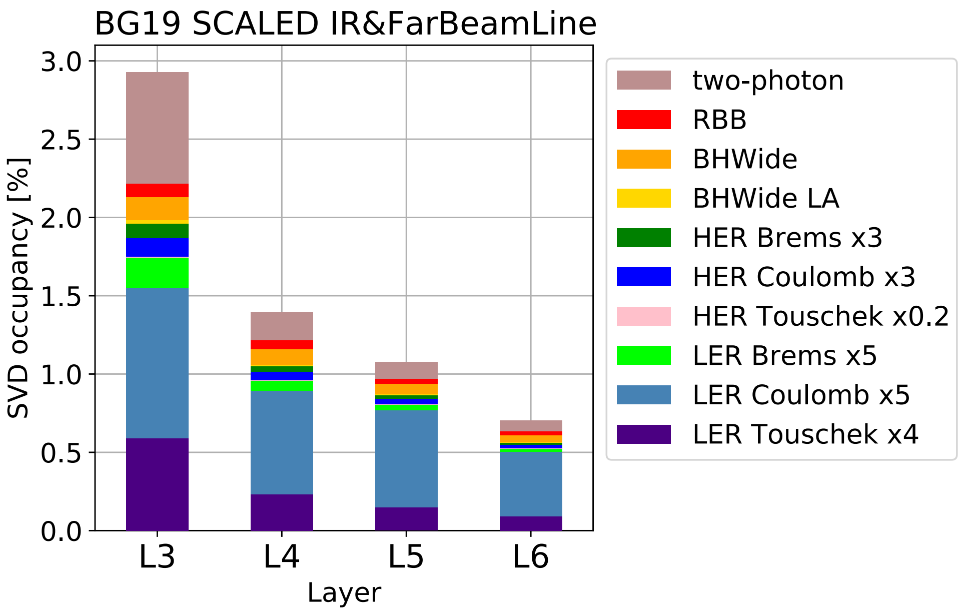


Figure 2.13: Occupancy of all SVD layers with 2020 background studies. RBB, BHWide and BHWide LA are referring to different photon emission angles

2.2.4 Central Drift Chamber (CDC)

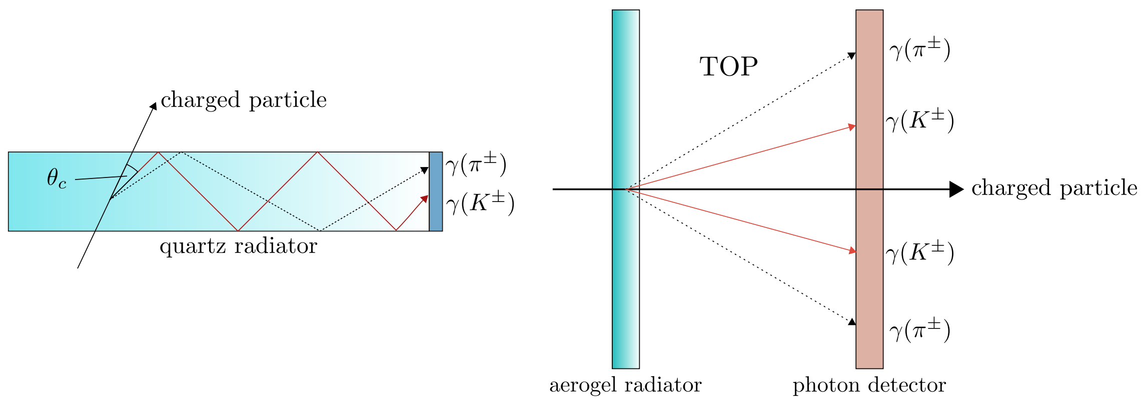
The CDC is responsible for the reconstruction of charged tracks as well as the measurement of their momenta. From the measurement of the specific energy loss (dE/dx), it performs particle identification. The CDC plays an important role for trigger signal generation.
The CDC is composed of eight superlayers, each composed of six layers of wires. The innermost superlayer is built with eight layers of wires. The chamber contains a gas mixture of 50% helium and 50% ethane. There are two types of wires in use. The first is needed to generate the accelerating electric field, the field wires, and they are made of aluminum. The other kind, necessary to collect the released charge, are the sense wires, and are gold-plated tungsten wires with a 30 μm diameter, considerably thinner than the 126 μm diameter of the field wires.
A charged particle crossing the CDC ionizes the gas and produces e^--ion pairs. The electrons, accelerated by the electric field, provoke a charge avalanche that can be measured on the sense wires. The track is reconstructed from drift time and hit wire position.
Three different types of layers are used and they differ but the way the wires are oriented. The axial wires are parallel to the z-axis and provide information about hit position in the r-ϕ plane. The axial layer is denoted as `A`. Nevertheless no information can be obtain about the z position with axial wires. For this reason another kind of wire arrangement is used. The wires are skewed around the beam line forming the so-called stereo layer. Stereo layers can be visualized as an axial layer with one of the end plate rotated. Depending on the direction of the rotation a stereo superlayers is either called`U` or `V`. Chosen to optimize the z resolution, the superlayer are arranged as follow: `AUAVAUAVA`.

2.2.5 Particle Identification - TOP and ARICH

A reliable Particle Identification (PID) is important for measurements at the B-factories. For the Belle II detector the PID is performed by TOP counters [15] in the central region and the ARICH counters [16] in the forward endcap region. For both TOP and ARICH the goal is the imaging of the Cherenkov rings. Cherenkov radiation is the result of a charged particle passing through a material faster than the speed of light in that specific material. Depending on the particle velocity, the Cherenkov radiation is emitted at with a characteristic angle θ_C with respect to the particle track with the relation:

$$cos(\theta_C) = \frac{1}{n\beta} \tag{2.3}$$

where n is the refractive index of the medium. By measuring θ_C, the Cherenkov detector is capable of obtaining the velocity β of the particle. TOP is made of 16 single modules placed around the CDC at a distance of 1.2 m from the IP. Each module consists of a quartz bar inside of which the Cherenkov photons are created and internally reflected. A micro-channel plate



TOP ARICH

Figure 2.14: Working principle for the TOP and ARICH Cherenkov detectors.

photomultipliers (PMT) on the edge of the bar can then reconstruct the Cherenkov image in x and y and provide precise timing information.
ARICH was designed to separate kaons from pions. It is built around an aerogel radiator of 2 cm thickness. A 20 cm expansion volume lets the Cherenkov rings form on the photon detector surface from which the circle radius and the particle velocity can be retrieved.

2.2.6 Electromagnetic Calorimeter (ECL)

The electromagnetic calorimeter the Belle II detector is used for reconstructing photons and electron tracks and to measure their energy. Moreover it generates hardware and software trigger signals and is used to monitor the SuperKEKB luminosity. The ECL reuses most parts of its predecessor in Belle. It is formed by 8736 pyramidal shaped crystals made out of CSI(TI) (caesium iodide,thallium-dopped). Located at the external base of the crystals, photodiodes collect scintillation light produced by traversing particle. With a length of three meters, the calorimeter is divided into three regions which are the central barrel, the backward end-cap at $z = 196\,\text{cm}$ and the forward end-cap at $z = -102\,\text{cm}$, covering polar angle in the range [12°,155°]. The energy resolution of the ECL is given by :

$$\frac{\sigma_E}{E} = \sqrt{\left(\frac{0.066\%}{E}\right)^2 + \left(\frac{0.81\%}{\sqrt[4]{E}}\right)^2 + (1.34\%)} \tag{2.4}$$

where E is the energy in GeV and σ_E is the energy uncertainty. The first term of the equation represents the electronics noise contribution.

2.2.7 K_L^0 and μ detection (KLM)

All the previously described Belle II sub-detectors are located inside a superconducting solenoid magnet producing a 1.5 T field and surrounding the ECL in the barrel region.
Outside of the solenoid, the KLM system is located. Its purpose is to identify muons tracks and long-living neutral kaons.
With a length of 4 m along the z-axis and a diameter of 3.4 m it is divided into three parts which are the barrel region and the endcaps which all together cover polar angle between 20° and 155°. It is made of a succession of irons plates and detector layers. The detection layers are based on glass-electrode Resistive Plate Chambers (RPC) [17] for the barrel region and on scintillation strips for the endcaps since RPCs cannot cope with high background rate. Wavelength-shifting fibers transport the scintillation light to the photon detectors. Due to space limitations and high magnetic field, conventional photomultipliers are not viable and Silicon PhotoMultiplier (SiPM)s [18] are used instead. In total, there are 14 iron plates on all sub-parts, 15 detector layers for the barrel and 14 for the endcap.
Kaons reaching the KLM create hadronic showers in the iron plates which is detected by the RPCs and SiPM. For the muons, CDC track candidates are extrapolated into KLM using a pion mass hypothesis. If a hit exists in the outermost layer of the KLM and is located near the extrapolated CDC tracks, a new track reconstruction is done based on muon mass hypothesis.

2.3 Trigger and DAQ

The high bunch crossing frequency of 250 Mhz, faster than the detector collection time, makes the beam almost continuous. Events are dominated by Bhabha scattering and two photons processes and their contribution has to be discarded to decrease the load on the DAQ. This task is fulfilled by a complex trigger system whose operation is introduced in this section. It is followed by details given on the Belle II DAQ and on the specificity of the PXD DAQ which differs from the other sub-detectors.

2.3.1 The Trigger System

The Belle II trigger system consists of two levels that both use the signals from the sub-detectors even though CDC and ECL are the main necessary sources.

The first level, or simply Level 1 trigger, is based on hardware using FPGA that can perform extremely fast online analysis of the sub-detectors signals. Level 1 trigger detects with high efficiency, over 99.9 %, $\Upsilon(4\mathrm{S}) \to B\bar{B}$ events at a maximum trigger rate of 30 kHz with a fixed latency of 5μs, and respecting a minimum time of 200 ns between two events. The final rate can be defined by the rate of all the sub-processes that are expected. The CDC trigger finds and characterizes tracks of charged particles passing through it. Different types of tracking algorithms are implemented to reconstruct the momentum and trajectories of the tracks. Through a total energy calculation and a cluster counting process, the ECL provides a trigger for neutral and charged particles. The first is most responsive for high energy depositions while the second one focuses on low-energy and minimum-ionizing particles. The calorimeter trigger also participates in the online luminosity measurement by efficiently detecting Bhabha and $\gamma\gamma$ events. Further trigger lines are from TOP and ARICH that deliver helpful timing information and also from KLM for muon tracks.

All the sub-detector triggers are received by the Global Decision Logic (GDL) which makes the final decision if at least one of the sub-trigger is valid. To reject background events a veto logic is implemented. The L1 trigger signal is then propagated via the Fast Timing Switch (FTSW) [19] over a thousand components of the sub-detectors on the front-end side. The trigger signal consists of a 32-bit trigger number which is incremented by one for every accepted event. It always comes with a 14-bit run number as well as an 8-bit sub-run number which are reset every time a new run starts. Finally a 10-bit experiment number is attached, which is increased for significant changes in the data taking conditions.

2.3.2 Belle II Data Acquisition

When a trigger is issued, the Belle II DAQ system collects the data coming from the included sub-detectors, combines them and stores them after an event has been built.

As already mentioned in section 2.2.2, valid for SVD but also for every other sub-detectors, the COPPER boards receive data from the different Front End Electronics (FEE) which are all using the belle2Link protocol.
With a trigger from FTSW, every COPPER board computes a local sub-event which is then transmitted to Readout PC (ROPC). This ensemble of PCs is known as Event Builder 0 (EB0) [20]. Finally, data from EB0 are transmitted to the event builder PC farm Event Builder 1 (EB1), where all the sub-detector data with a common event are grouped together. The data rate after EB1 is about 2.5 GBps with a trigger rate of 30 kHz. Still too high to be storable, the rate is further reduced by the second level of trigger, the High Level Trigger (HLT) [21]. The HLT runs on thousands of CPUs and performs an online tracking based on SVD, CDC and ECL clusters. Event selection is done, and background removed based on extracted parameters such as track multiplicity, vertex position, total energy deposited and reconstructed track. Since processing an event can take up to few seconds, the overall output rate of the HLT is about 10 kHz. Events selected by HLT can then be moved to the Event Builder 2 (EB2) for final storage. An overview of the Belle II DAQ system is shown in fig. 2.15

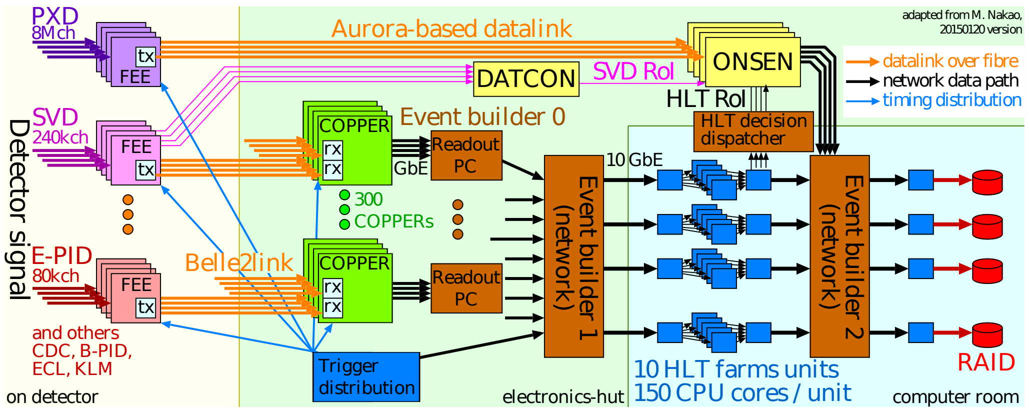


Figure 2.15: Overview of the Belle II data-acquisition system.

2.3.3 PXD Data Acquisition

The data acquisition of the PXD does not follow the same model as the other sub-detectors due to its huge output rate. Discussed in section 2.2.1 , the pixel detector, being very close to the beam pipe, is highly affected by backgrounds that represent up to 3 % occupancy. Since the effective signal occupancy is only 0.1 %, it does not play a significant role in the final bandwidth estimation. On average 2.5 bytes are necessary to encode the raw pixel data. Coupled with the level-1 trigger rate of 30 kHz and the total number of pixels of the PXD, the maximum expected bandwidth is about 3 %×7,680,000×2.5 B≈17.3 GBps. The PXD data is encapsulated into a dedicated protocol with a final rate of 20 GB/s. This estimation is about ten times higher than the data rate of the others sub-detector making it mandatory to have a dedicated DAQ for PXD. The PXD DAQ is built around the Data Handling Hybrid (DHH) [22] which is depicted on fig. 2.16. The four 1.5 Gbps DHPs links of a PXD module are connected to one

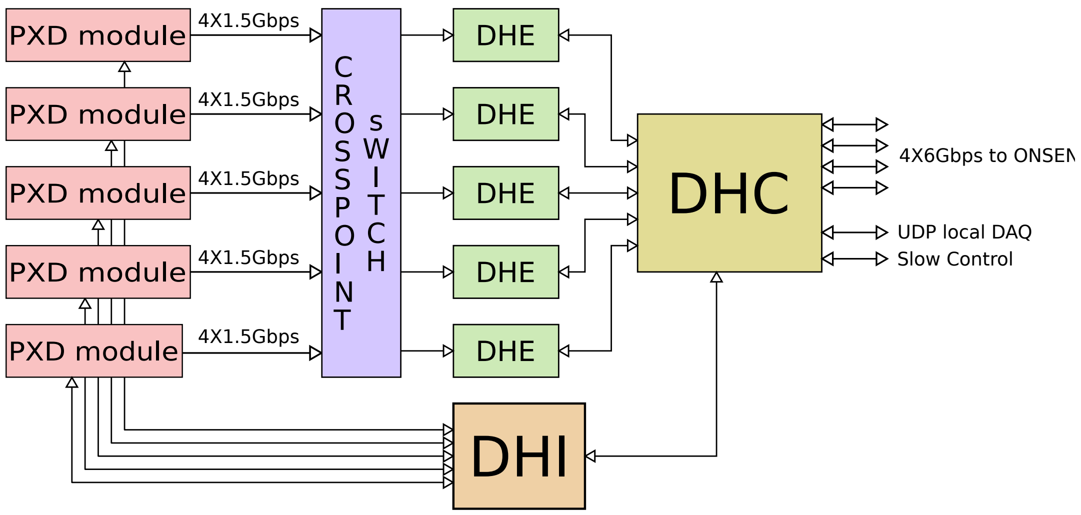


Figure 2.16: Topology of the DHH system used for the DAQ of 5 PXD modules. DHPs are connected to the DHEs via the crossing switch. They are connected to a single DHC which distributes slow control signals and communicates to ONSEN. The DHI is controlling the detector modules. From [23].

DHE via a crossing switch. The crossing switch offers flexibility to define optical link topology between the modules and the DHH The DHE is responsible for buffering data, using DDR3 memory, and for processing them via four identical streams. The task of the DHE is delicate. Considering the long integration time of 20 μs and a trigger rate of 30 kHz, multiple triggers can occur during a single PXD frame. A dedicated mechanism based on the time of arrival of the trigger is implemented to deal with this situation [23].
The DHC receives and merges the data from five DHEs by instantiating a 5-to-1 multiplexer. Sub-events are built and sent out over four optical links to the data reduction system also known as ONSEN [24]. One more piece of hardware comes to complete the PXD readout chain, the DHI which controls five PXD modules. It provides clock signal, control command and Joint Test Action Group (JTAG) slow-control information. A total of five DHEs, one DHC and one DHI compose one eighth of the DHH. They are plugged into Advanced Telecommunications Computing Architecture (ATCA) boards.

2.3.4 Data Reduction Principle

The large amount of data produced by PXD requires a specific DAQ system, whose bandwidth still needs to be reduced to be accepted by EB2. Most of the output generated by the PXD is due to background. The idea to remove backgrounds hits and reduce the amount of PXD data, is to find tracks and their intersection with the PXD planes, and to create a ROI at that location. Only the fired pixels inside a ROI are kept. Naturally, the data reduction highly depends on the quality of the rack finding and the number of tracks, some of which could be fake tracks. The principle of ROI creation based on the SVD hits produced by a single particle is shown in fig. 2.17.
Two systems are used to perform the data reduction. Already mentioned in section 2.3.2, the HLT, responsible for event selection, is also used for data reduction based on the CDC and the SVD data. Running on PC, its decision can take up to few seconds. The second system is DATCON, whose task is to perform track finding and extrapolation based on SVD data only

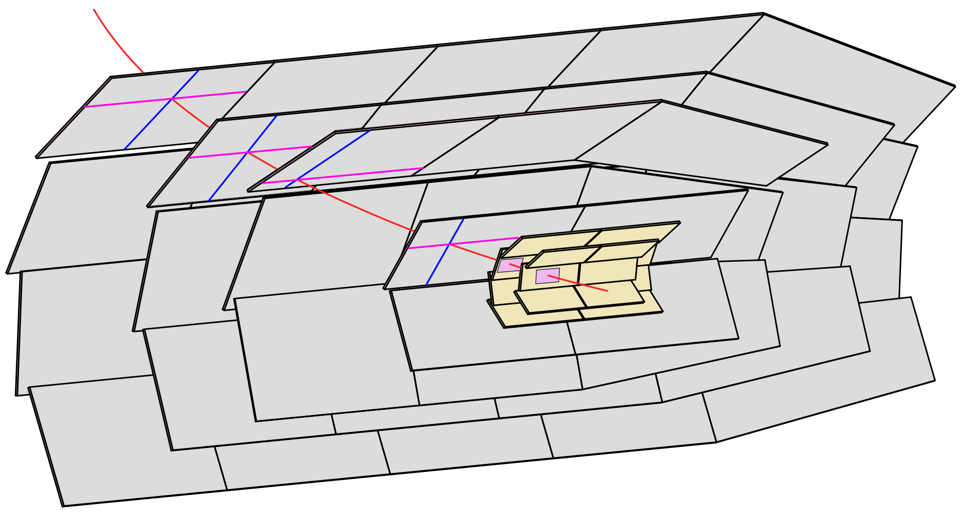


Figure 2.17: Illustration of the principle of track extrapolation and ROI calculation onto both PXD layers based on the surrounding SVD hits. One pair of strips (red are $\mathcal{P}$ strips, blue are $\mathcal{N}$ strips) is activated on each layer. The ROIs are shown as pink rectangles.

The main difference of DATCON compared to the HLT is that it runs exclusively on FPGA and it generates ROIs at roughly the same rate as the FTSW, much faster than the HLT.
The ONSEN data reduction system receives the two types of ROIs and is able to buffer the ones from DATCON but also the PXD data until it receives ROIs from the HLT. Giving HLT the highest priority, a ROI merging is performed and selection is applied on the PXD hits. The data rate from ONSEN to the EB2 is reduced from 20 GBps to 550 MBps.

3 Basics of Track Finding and Extrapolation

Based on the hits from the four layers of SVD sensors surrounding the PXD, DATCON can find tracks and extrapolate them back onto the PXD planes. The intersection of the reconstructed track with the pixel sensor is called the Most Probable Hit (MPH). A ROI is built around every MPH. This chapter describes the methods used to reconstruct tracks by using a Hough Transformation. The extrapolation of the MPH and the coordinate system used with its intrinsic errors are also discussed.

3.1 Hough Transformation

Originally introduced and patented by Paul Hough, the Hough transform was first used to detect straight tracks in photographs obtained with bubble chambers [25]. It was later adapted to detect various type of shapes and it is now widely used in image processing and high energy physics as well.
The simplest case for which the Hough transform can be used is to detect straight lines that resented by the eq. (3.1)

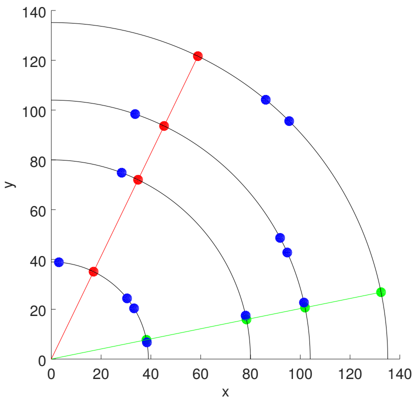
$$y = a \cdot x + b \tag{3.1}$$

The idea is that every point (x, y) in spacial coordinate can be described in a new space called Hough Space (HS) represented by the parameters a and b. The transformation is expressed as follow:

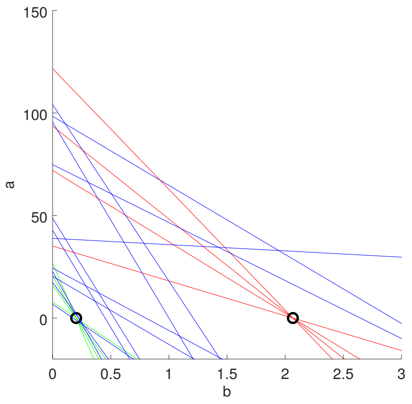
$$y_i = a \cdot x_i + b \xrightarrow[trafo]{Hough} b = -a \cdot x_i + y_i \tag{3.2}$$

Points are represented by a line in the HS and points originating from the same track have their lines intersecting in the HS as shown in fig. 3.1. This way of detecting straight lines quickly shows its limit for a few reasons. The first is the requirement to check the whole range for b.
Since it has to be suitable for FPGA, this could be a resource limitation problem and not a viable solution. The second complication concerns the case of tracks parallel to the y axis.

Hits belonging to this kind of tracks generate parallel lines in the HS that prevent them to intersect.



(a) SVD hits generated by two straight (red) tracks and additional background (blue).



(b) Hough space obtained after transformation of all the hits from (a). Lines intersect in two distinct locations.

Figure 3.1: Illustration of the Hough transformation from two straight track and randomly distributed hits.

To deal with this issue it has been decided to use a different parametric representation of a straight line. Based on L. Hesse's work and introduced by R. Duda and P. Hart[26], the Hesse parametrization represents a straight line by its closest distance d to the origin as well as the angle φ as depicted in fig. 3.2

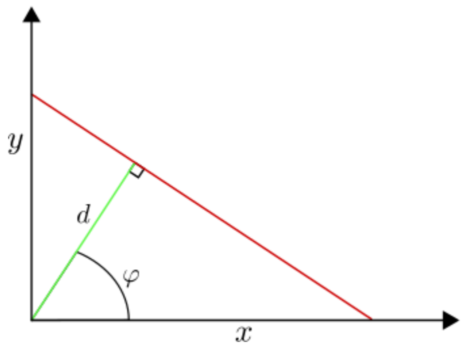


Figure 3.2: Hesse parametrization of a straight line described by distance d from the origin and angle φ.

Lines can therefore in the Hesse normal form be expressed as follow:

$$d = x_i \cos\varphi + y_i \sin\varphi \tag{3.3}$$

Using the Hesse parametrization, the range of φ is then reduced to $[-\pi/2, \pi/2]$. The maximum distance d can not exceed $\sqrt{x+y}$. Even if the original form of the Hough transformation will not be used in the rest of this thesis and is replaced by the Hesse parametrization, the term Hough transformation and Hough space will still be used. From the hit cloud of fig. 3.1a the Hesse parametrization generates the following Hough space depicted in fig. 3.3

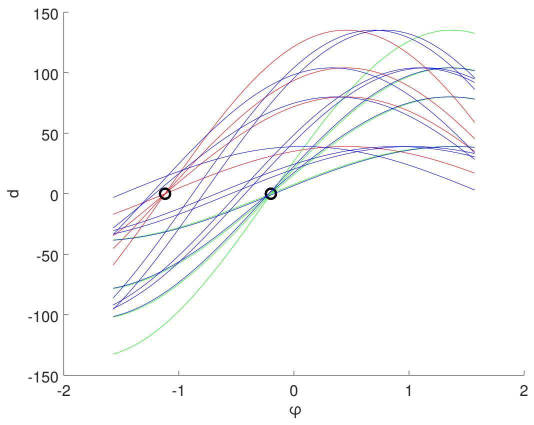


Figure 3.3: Hough space generated by using the Hesse transfomation. Lines that intersect come from hits that belong to the same track.

In the example the tracks have a common origin $(0, 0)$, therefore the shortest distance to the origin is 0. This can be verified in fig. 3.3 where curves intercept exactly on $d = 0$.

3.2 Conformal Mapping

It has been shown that the HS can be used to find straight tracks. However, at Belle II the magnetic field causes the particle trajectory to be bent in the r-φ-plane.
While the curvature of high-momentum tracks in a magnetic field is small and they can be approximated to straight lines, it is not possible for low momentum particles which also are important for the B-physics measurements of Belle II . Due to their circular trajectory, they can not be found with the basic approach of the HS which can only detects rectilinear tracks. With the introduction of the conformal mapping, points on an arc can be mapped into points on a straight line and therefor the Hough Transformation (HT) computed. A circle is defined by the following equation:

$$r^2 = (x-a)^2 + (y-b)^2 \tag{3.4}$$

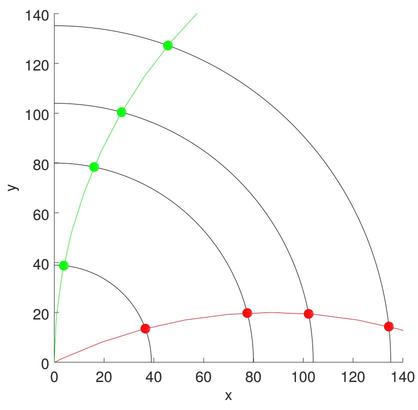
Here, a and b are the coordinates of the center of the circle. From the circle equation the conformal mapping can be derived

$$\begin{aligned} r^2 &= (x - r\cos\varphi)^2 + (y - r\sin\varphi)^2 \\ 0 &= x^2 + y^2 - 2r(x\cos\varphi + y\sin\varphi) \\ \frac{1}{2r} &= \frac{\cos\varphi}{x^2+y^2} + \frac{\sin\varphi}{x^2+y^2} \end{aligned} \tag{3.5}$$

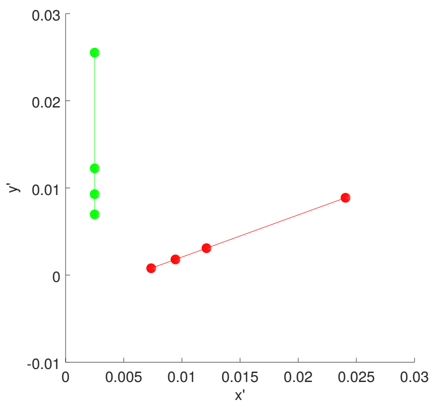
This equation is equivalent to the Hesse paramtetrization of a straight line as shown in eq. (3.3) when introducing (x', y) as:

$$x' = \frac{x}{x^2+y^2}, \; y' = \frac{y}{x^2+y^2} \tag{3.6}$$

In order to use the HS to find curved tracks, every hit is converted through the conformal mapping. With converted hit coordinates, the Hough transformation can be applied. An example of the comformal mapping of hits from a curved track is shown in fig. 3.4.



(a) SVD hits produced by two curved tracks.



(b) Corresponding hits after the conformal mapping was applied.

Figure 3.4: Illustration of the conformal mapping. Curved track coordinates are mapped into a new space where they appear as lines, since the Hough transformation is possible only for straight tracks.

3.3 Extrapolation and ROI Creation

Once the track parameters, ϕ, θ and r, have been extracted from the HS, extrapolated hits on the PXD layers can be determined. The goal here is to find the location of an helix intersecting a straight plane, in this case the PXD sensor. For solving this problem, it is more practical to use a unique reference plane orientation.

Every PXD sensor is therefore back-rotated by its rotation angle, which is responsible for the windmill structure (section 2.2.1). The newly rotated sensor is parallel to the y-axis and has a $0°$ angle. The track is also rotated by the same sensor angle. Figure 3.5 illustrates the concept of back-rotation. This method simplifies the intersection calculation between helix track and the detector plane, because the x coordinate is already known as the radius of the current sensor.

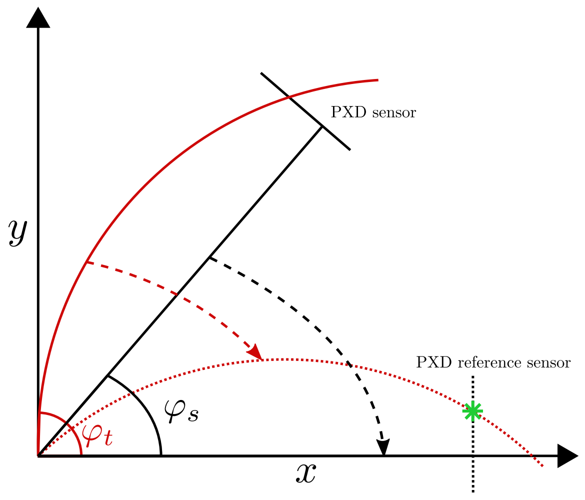


Figure 3.5: Concept for the MPH extrapolation onto the PXD planes, using an analytical method to precisely calculate the intersection highlighted in green. In red, the track with an angle φ_t is rotated by the sensor angle φ_s. The dashed track and sensor are the results of the rotation.

The track trajectory on the $x - y$ plane is similar to a circle following the equation:

$$(x - x_c)^2 + (y - y_c^2) = r_t^2, \tag{3.7}$$

where x_c and y_c are the center coordinates of the track and r_t its radius.

From eq. (3.7) the y coordinate of the intersection of the track and the reference sensor can be expressed as follow:

$$y = y_c \pm \sqrt{r_t^2 - (x - x_c)^2} \tag{3.8}$$

To adapt the calculation for any track intercepting any sensor, the center has to be rotated. Here, φ_t and φ_s are the angle of the track and the sensor respectively, and r_t and r_s are the radius of the track and the radius of the sensor. A new variable φ is introduced and can be expressed by:

$$\varphi = \varphi_t - \varphi_s \tag{3.9}$$

The track center is rotated by φ as follow:

$$(x'_c, y'_c) = (x_c, y_c)\begin{bmatrix}\cos(\varphi) & \sin(\varphi)\\ -\sin(\varphi) & \cos(\varphi)\end{bmatrix} \tag{3.10}$$

By replacing the original track center coordinates by the newly obtained in eq. (3.8), the location of the extrapolated hit onto the current PXD ladder is expressed by:

$$y_s = r_t\cos(\varphi) \pm \sqrt{r_t^2 - (r_s - r_t\sin(\varphi))^2} \tag{3.11}$$

The $\pm$ sign depends on the sign of the track radius sign obtained from the HS.
For one track candidate there are as many extrapolated hits as there are PXD sensors but it is considered as valid only if it belongs to the boundaries of the 0° PXD sensor.
For extrapolation in the r-z plane, the track is considered as a straight line which causes an error during calculation of the MPH location. An example of the extrapolation and the error induced is shown in fig. 3.6

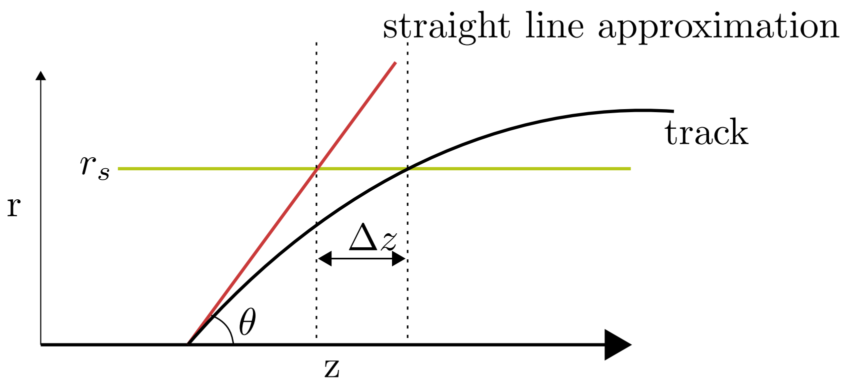


Figure 3.6: Illustration of z calculation and the error caused by approximating the track as a straight line.

The z-component can be determined with the track's polar angle θ in the following way:

$$z = \frac{y}{tan\theta}\ ,\ z = y\cdot tan\left(\frac{\pi}{2} - \theta\right) \tag{3.12}$$

Forasmuch as multiplication is preferable for FPGA over division, the two calculations for the extrapolation are shown. The extrapolated track onto the PXD is converted into a single pixel around which a ROI is built.

3.3.1 Error of the Straight Line Approximation

Using the straight line approximation for a track simplifies the calculation on the FPGA but it also generates an error on the z position of the MPH. An example of induced error is depicted in fig. 3.6 On the $r - z$ plane, the track has the shape of a sine function from which the amplitude and period must be known to calculate its intersecting position with the PXD layer. The amplitude corresponds to the radius of a track. From the synchrotron frequency:

$$\omega_c = \frac{qB}{m}, \tag{3.13}$$

where q is the charge, B the magnetic field and m the mass of the particle, the period is calculated as follow:

$$T = \frac{2\pi}{\omega_c} \tag{3.14}$$

The y and z components of the track as a function of time t are expressed by:

$$\begin{aligned} y(t) &= r_0 \cdot \sin(\omega_c t), \\ z(t) &= v_z \cdot t = \frac{p_z}{m\gamma} t = \frac{p_t \cdot \tan(\lambda)}{m\gamma} t \end{aligned} \tag{3.15}$$

where v_z is the z component of the particle velocity, p_z the z component of the momentum, γ is the Lorentz factor and r_0 is the track radius defined by $r_0 = \frac{p_t}{0.3 \cdot B}$.
The intersection of the track and its straight line approximation on a PXD sensors located at r_s are expressed by:

$$\begin{aligned} r_s(t) &= a \cdot t \text{ for the straight line,,} \\ r_s(t) &= r_0 \cdot \sin(\omega_c t) \text{ for the track,} \end{aligned} \tag{3.16}$$

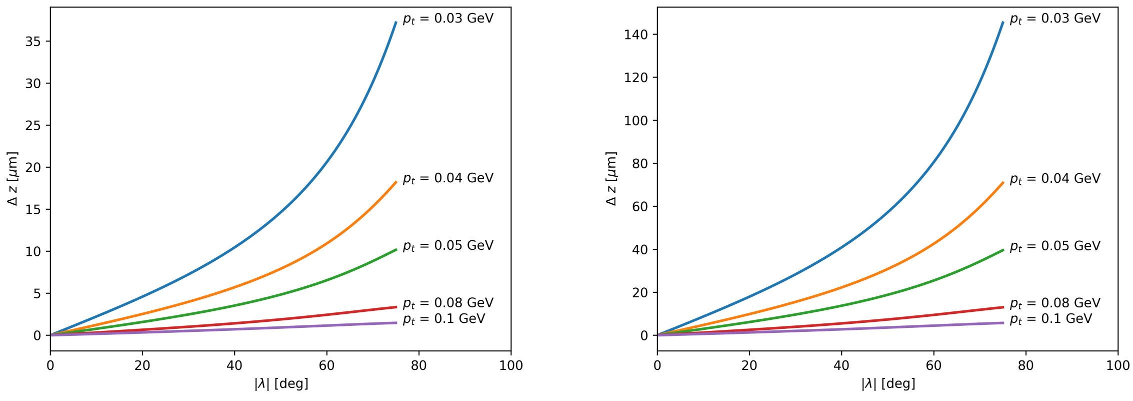
where the slope at $t = 0$ is defined by a. The time t_1 and t_2 at which the tracks intercept the PXD sensor can be expressed with :

$$\begin{aligned} t_1 &= \frac{r_s}{a} \\ t_2 &= \frac{1}{\omega_c} \sin^{-1} \frac{r_0}{r_s} \end{aligned} \tag{3.17}$$

The error Δz caused by a straight line approximation can be expressed by:

$$\Delta z = z(t_1) - z(t_2) \tag{3.18}$$

To illustrate the effect of the straight line extrapolation, a pion was used as it represents 70 % of the tracks produced at Belle II . Momenta from 0.3 GeV up to 0.1 GeV are used as well as λ in the range[0°,75°]. The error is shown in fig. 3.7. [27]
It appears that for all the cases, even for very low momentum and high λ values, the error is negligible. The largest error for layer 1 is less than one pixel and for layer 2, less than two pixels. The use of the straight line approximation to find the intersection of the extrapolated track with the PXD plane is justified.



(a) Error on the PXD layer 1.

(b) Error on the PXD layer 2.

Figure 3.7: Error on the z position of the intersection of the PXD layers with a track and with the straight line approximation for momentum in the range [0.03 GeV,0.1 Gev] and λ in the range [0°,75°] on the two PXD layers.

3.3.2 Error of the Radius Approximation

When calculating the z value of the MPH for DATCON, the radius of the track is considered to be the same as the radius of the PXD layer. The track radius r_t is defined as:

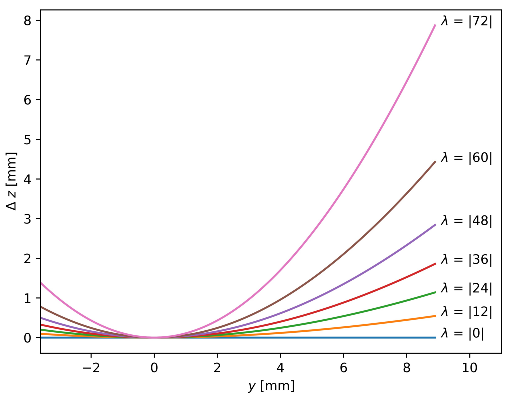
$$r_t = \sqrt{r_s^2 + y^2} - r_s, \tag{3.19}$$

with r_s the radius of the PXD, and depends on the ϕ angle of the track which is not available when calculating z. This causes additional error for the calculation of z, expressed as follow:

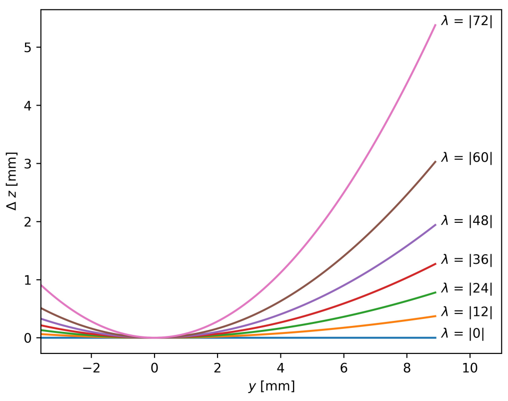
$$\Delta z = \left(\sqrt{r_s^2 + y^2} - r_s\right) \cdot \tan(\lambda) \tag{3.20}$$

From the y value of an MPH on the PXD sensor as shown on fig. 3.5, the resulting error on z is calculated based on an absolute λ in the range [0°,72°]. The calculated errors on z are shown on fig. 3.8. For the upper range of λ where the error is the most important, it reaches

5.4 mm or 70 pixels for the first layer of PXD and 7.8 mm or 92 pixels for the second. Out of the two sources of errors presented in the two last sections, the simplification of the radius is the most important.



(a) SVD hits produced by two curved tracks.

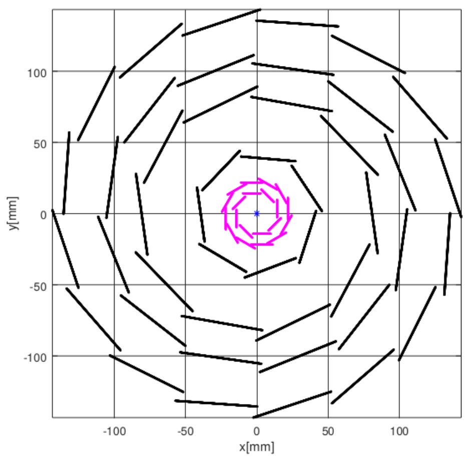


(b) Error on the z position of the track on the PXD layers caused by the radius representation.

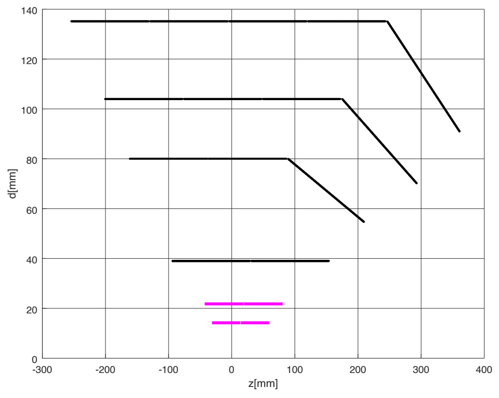
Figure 3.8: Error of the z position of the intersection of a track with the two PXD layers caused by the radius representation.

3.4 Strip Coordinate System

The DATCON system performs two independent track reconstruction, one in the the r-ϕ plane or $\mathcal{P}$ side, the other in the r-z plane or $\mathcal{N}$ side. They produce independent ROIs that are later merged. It is important to understand how a strip can be converted into geometrical coordinates necessary to compute the steps introduces in the previous sections. On the $\mathcal{P}$ side the coordinates used are the projection of the strips in x, y. For the slanted strips, a simplification is made as the inclination is not taken into account. With this simplification, every sensor of a ladder has the same coordinates. The used $\mathcal{P}$ strip positions are shown in fig. 3.9a. In addition to the four layers of SVD the two layers of pixel detectors are also shown in purple. For the $\mathcal{N}$ side the slanted strips are not neglected but another simplification is made. Every SVD ladder, which is normally rotated along the $z - axis$ and in the r-ϕ plane to create the windmill structure, is brought back to a one-ladder-per-layer representation as shown in fig. 3.9b. An $\mathcal{N}$ strip is then identified with the pair (z, d) where d is the distance from the beamline to the strip. The z component is preserved with this detector representation.



(a) Coordinates on φ. The slanted angle of on the forward region are not taken into account.



(b) Coordinates on θ. One ladder of each layer is used to represent the entire SVD coordinates on the r-z plane

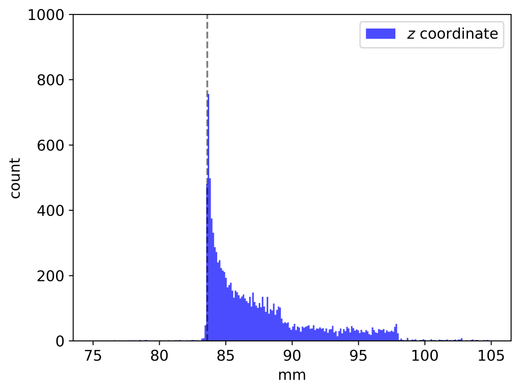
Figure 3.9: Representation of the possible strip coordinates used as input for DATCON

3.4.1 Source of Errors

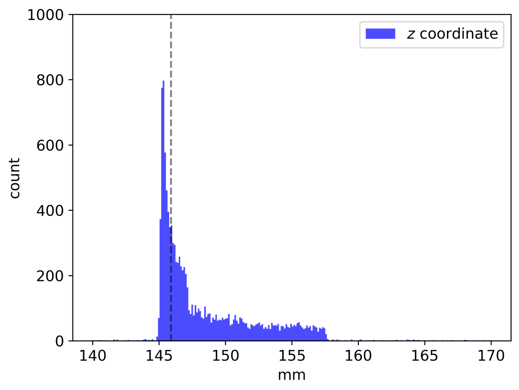
If the the coordinates system used for the $\mathcal{N}$ side provide a suitable input for 2D tracking, it is moreover prone to intrinsic errors. Indeed, due to the detector design, and considering a constant polar angle θ different from 90°, the distance from the IP to a sensor strip varies based on the value of ϕ.

To visualize this incertitude, 5×10^4 generated single tracks are used. Each of those tracks is identical on the $\mathcal{N}$ side with θ=40°. With the chosen angle, tracks generate a hit on every SVD layer and they also are passing through the slanted part of the detector. On the $\mathcal{P}$ side, tracks can take any ϕ value within the range [0°,360°]. The Monte Carlo (MC) event generation used for this section is detailed in Section 7.5.5.

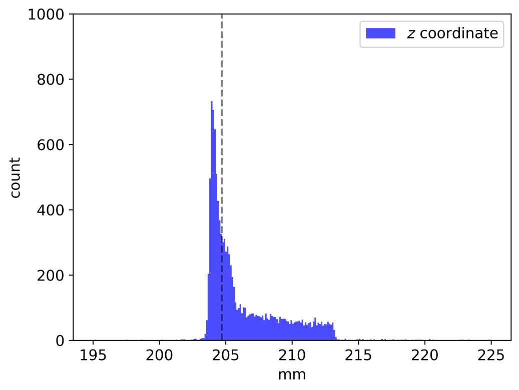
Since the track angle on $\mathcal{N}$ side is the same for each track, the z-value of a SVD hit should be the same for each layer. There is however, a considerable shift, as can be seen on the distribution of track intersection with the PXD sensors depicted in fig. 3.10. The dashed line represent the expected z position of the track in the current representation. The error of the strip position can cause an incorrect track angle detection. When the angle is correctly found, the track can alas appear shifted. Since the extrapolation is based on an IP located at the origin, a shifted track inevitably results in a shifted ROI possibly missing valuable PXD pixels.



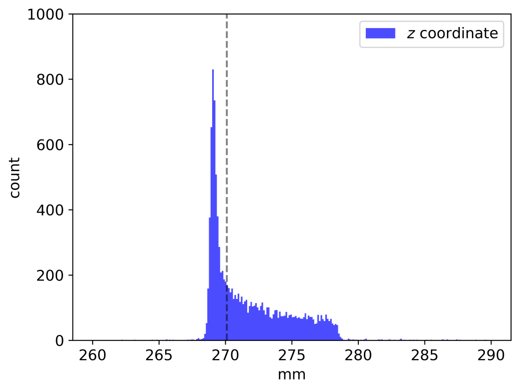
(a) z position on layer 3.



(b) z position on layer 4.



(c) z position on layer 5.



(d) z position on layer 6.

Figure 3.10: Intrinsic position error caused by the coordinate system used by DATCON to represent SVD on the $\mathcal{N}$ side. Dashed lines show the expected position of θ for each layer in the current representation (fig. 3.9b)

4 DATCON Structure and Hardware

In this chapter the different components of the DATCON system are introduced and described. Starting with a general overview of the structure and interconnections, a short introduction to FPGAs, which are the core of this project, follows. Two kinds of chips are used and are mounted on different sorts of PCB respecting the Advanced Mezzanine Card (AMC) standard. The first kind of FPGA is the Virtex 5 [28] and is installed on the so called Concentrator AMC which handles data stream coming from four FTBs (section 2.2.2). The details of the Concentrator firmware are described in Chapter 5.
For the tracking and the ROI creation, which require significantly more resources than the Concentrator, a Virtex 6 [29] was chosen. The tracking implementation is discussed in more detail in Chapter 6.
All the DATCON boards and the entire DATCON system is based on the Micro Telecommunications Computing Architecture (μTCA) standard for which a custom chassis was designed. Finally the DATCON hardware installation is described.

4.1 DATCON Structure

The DATCON system is composed of two custom chassis, each being responsible for performing tracking on either $\mathcal{P}$ or $\mathcal{N}$ side. All seven slots of the $\mathcal{P}$ chassis are occupied by a Concentrator card whereas for the $\mathcal{N}$ chassis only six slots are used.
Every Concentrator has four Small Form Plugable (SFP) cages equiped with optical transceivers that connect to the FTBs. After processing the SVD raw data, the Concentrator transmits the information to the Tracking units via electrical connection in the backplane. There are two Tracking boards, one for $\mathcal{P}$ side and one for $\mathcal{N}$ side. To reconstruct φ and track radius, the $\mathcal{P}$ samples are used while the $\mathcal{N}$ side hits are used for extracting θ. The SVD data-stream contains hits of both $\mathcal{P}$ and $\mathcal{N}$ side where only one type is necessary per chassis. For this reason the two Tracking units are optically connected. Each chassis has to send the unnecessary data it has received to the other one. After splitting the data, $\mathcal{P}$ or $\mathcal{N}$ SVD hits are brought to the corresponding Tracking unit. A set of hits is ready to be processed for tracking once data has been re-distributed over the two chassis. An additional optical link is used to send the ROIs that have been found on the $\mathcal{N}$ side to the main chassis.
At last, a single optical link is used for sending the calculated ROIs to ONSEN. The task

is performed by the $\mathcal{P}$ chassis which is also responsible for building the final ROIs. A block diagram of the DATCON structure is depicted in fig. 4.1

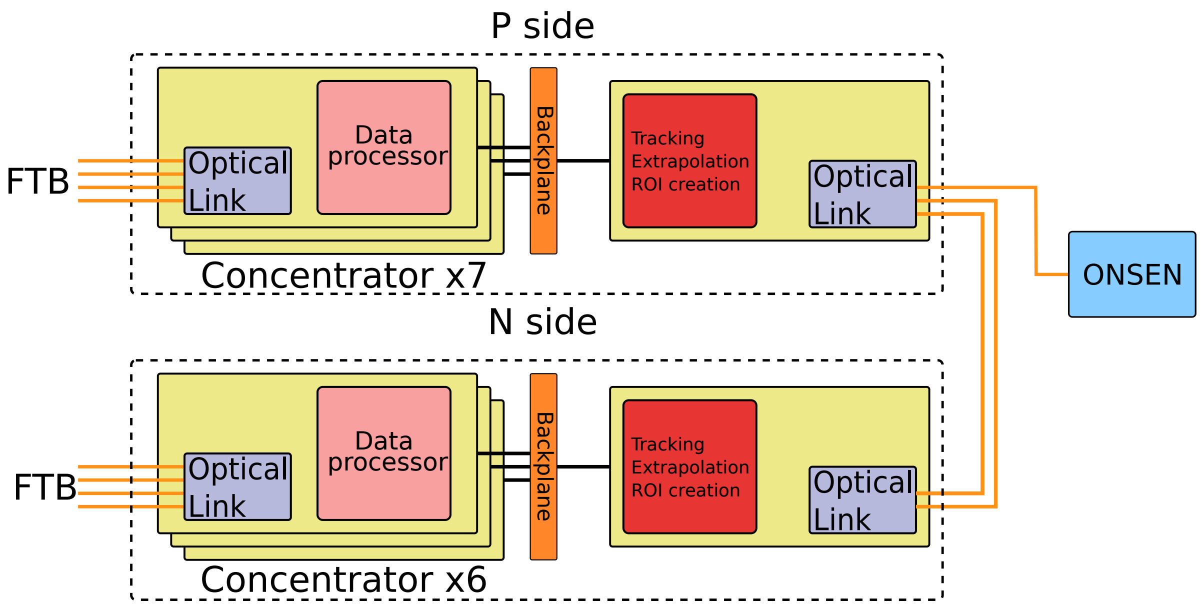


Figure 4.1: Layout of the DATCON structure divided into two main blocks. The concentrators are connected to the FTBs via optical fibers. Strip signals are processed before being sent to the tracking units. Two independent track reconstructions are performed in parallel and the $\mathcal{P}$ chassis sends the ROIs to ONSEN.

4.2 Field Programmable Gate Array (FPGA)

FPGA is a digital Integrated Circuit (IC) intended to be freely configured by the user, making it very versatile even if they have no pre-defined function at power up. FPGAs can be configured with a custom bitmap that dictates the configuration of the logic cells along with their interconnections.
The logic cell, called Configurable Logic Block (CLB), is the main component used for the design of sequential and combinatorial circuits. It consists of an n-bits function generator which integrates an address decoder and a Lookup Table (LUT). The goal of the CLB is to reproduce the behavior of any n-bit logic cell.
The output of the LUT can then be distributed inside the FPGA or stored in a D-Flip Flop (FF). A simplified overview of a CLB is shown in fig. 4.2 CLBs are arranged in a matrix within the FPGA and can be connected via programmable routing as shown in fig. 4.3. By doing so, various logical and sequential circuits can be created. The CLB matrices are surrounded by a large number of General Purpose Input/Outputs (GPIO) that can comply with variety of voltage level standards such as LVCMOS or LVTTL.

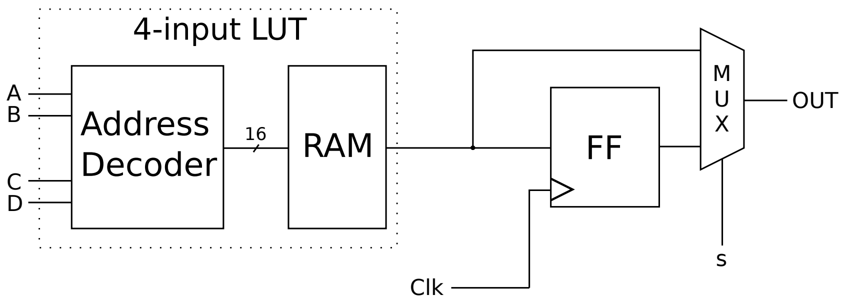


Figure 4.2: Configurable Logic Block layout. Logical functions can be implemented via LUT. The output, controlled by a multiplexer can be stored in a FF or directly propagated to the rest of the logic. [30]

GPIOs are the connections to the outside world, allowing for interfacing with a variety of components such as oscillators, memories or, for example, Ethernet physical transceivers. The FPGAs used for DATCON are equipped with differential high speed transceivers. They can be based on the serial signaling Low Voltage Differential Signal (LVDS) standard usually not exceeding a speed link of about 1 Gbps.
The second type of transceiver is the Multi-gigabit Transceiver (MGT) which can be used for data transmission up to 30 Gbps. On the chips used for DATCON, transceivers are designated by Xilinx as RocketIO GTP Virtex 5 and GTX for Virtex 6. A description of the transceivers is given in section 4.3. Configuration of an FPGA requires multiple prior steps which start

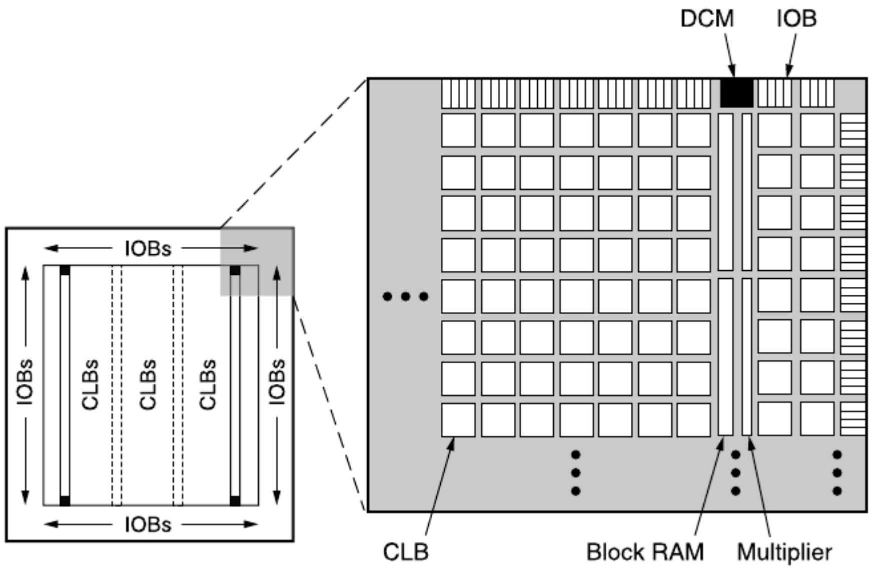


Figure 4.3: Example of an FPGA layout where the CLBs are arranged in a matrix which is surrounded by IOs and other elements such as block RAM. The different building blocks are referred as tiles. Interconnections within the matrix a changeable depending on the user design. [30]

by choosing a design entry method. A Hardware Description Language (HDL) is necessary to write a firmware and give a specific function to the FPGA. The two most common HDLs are the Very High Speed Integrated Circuit Hardware Description Language (VHDL) and Verilog. For this project, Verilog was chosen even though some external parts are written in VHDL. Both languages can be mixed within a project as they are both a high-level representation of a circuit based on the Register Tranfer Level (RTL) abstraction. To build the DATCON firmware, the Xilinx design suite ISE 14.7 was used [31]. ISE has now been discontinued but newer Xilinx tools are unfortunately not intended for the targeted FPGAs. Starting from a source code, ISE is capable of generating the bitstream file that will be flashed or downloaded into the FPGA.
Simulation are often recommended for large design before being ported on hardware. From the Verilog files only, a behavioral simulation can already be performed to validate the circuit operation. A Verilog Testbench file is necessary for simulating the design where stimuli, such as a clock signal, can be precisely defined. A more detailed example is described in section 6.8.

The complete list of compulsory steps to produce a bitstream file are the following

- **Synthesis**, equivalent to translating the RTL design into a netlist at a gate level which corresponds to the actual logic blocks in the FPGA.

- Successful **Synthesis** is followed by the **Translate** and **Map** phases where the netlists and constraints are fitted into the available resources of the target device.
  There are two important types of constraints: the placement constraints necessary to specify the location of some elements such as I/Os and the timing constraints. Timing constraints are applied to a given path or net and are primordial to match the desired timing performance of the design. They ensure that FF setup and hold time can be respected throughout all the blocks. A typical constraint example would be a clock input with the reference to the pin connected to the oscillator as well as its frequency.

- The **Place and Route** step is in the continuity of **Map** where ISE places and routes the design according to the constraints.

- Finally, the bitstream file is created and needs to be uploaded to configure the FPGA. If there are several methods to configure an FPGA, the one via JTAG was chosen for the DATCON boards. Chip configuration will be lost after a power-cycle of the board, but since they are meant to run continuously for long periods, this is not a problem.

4.3 High Speed Data Transmission

Used for optical link between FTB and DATCON and throughout the backplane between the different AMCs, high speed data transfer relies on multi-gigabit transceivers [32] wrapped in the Xilinx Aurora 8b/10b point-to-point link-layer protocol [33].

4.3.1 Xilinx Tranceivers and Aurora

All the FPGAs used for DATCON embed multi-gigabit transceivers which come with different naming and specification depending on the chip technology. For instance, the Virtex 5 of the Concentrators are equipped with **GTP**s [34] allowing a maximal data rate of 3.75 Gbps. The ones one the Virtex 6 of the Tracking AMCs use **GTX**s [35] which is similar to GTP but offers higher transmission speed, up to 6.25 Gpbs. Regardless of the naming, they all include, among others, de-serializer, comma detection and 8b/10b encoder and decoder.
To ensure proper communication between two transceivers, many protocols are available. One of them is the Xilinx Aurora 8b/10b protocol, which is used for this project. The protocol regulates data transfer across a channel which regroup one or more lanes. The entities communicating via a channel are called channel partners. After connection of two partners, Aurora sends idle frames to verify the channel stability before it can start data transmission. Aurora is part of the Xilinx LogiCORE IP [1] collection, and provides a wrapper for the transceivers as well as Aurora logic files. The core offers LocalLink interface that serves as connection between the user logic and the core by providing various ports such as Start Of Frame (SOF) and End Of Frame (EOF), source and destination ready and most importantly, the data received or sent. It is the task of the designer to build a design controlling those ports.
On the Virtex 5 FPGAs, a GTP tile contains two transceivers each connected to an Aurora lane. The two lanes are bonded together to form an Aurora channel. Each transceiver is physically connected on the PCB to the SFP cages, and each of those cages is intended to be connected to one FTB link. In order to create two single-lane channels, the generated Aurora HDL files had to be modified to make the transceivers independent.
On the Virtex 6, transceivers are grouped by four. However, they each are on a different tile and no modification is necessary.

4.3.2 8b/10 Encoding

In a baseband transmission, the signal is usually distributed to the transmission line only after line coding. The line code is a pattern used to represent a digital data that needs to be transmitted by adapting it to the physical properties of the medium. Long sequences of 0 or

[1]Xilinx provides a catalog of highly parameterized Interllectual Proporties (IP) that can perform a variety of functions. It is intended to help the designer gain time and efficiency for developing a system.

1 that do not provide many transitions can make clock recovery on the receiver side difficult and are also prone to errors.
There are different line codes available such as Manchester or RZ code, but the one in use for the Aurora protocol is the 8b/10b code. The 8b/10b encoder converts 8-bits data into 10-bits codes. There are in total 256 data characters referred as $D_{x.y}$ and 12 control characters $K_{x.y}$. Control characters are used for low-level functions such as idle, alignment or data delimiters. The 8b/10b encoding splits the 8-bits input into one block of 3 bits (y), and one of 5 bits (x).

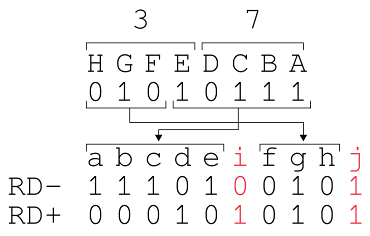


Figure 4.4: Result of the 8b/10b encoding of the 8-bit value 37.

A 5b/6b and 3b/4b encoding are performed on those two blocks. The initial bits are arranged following the order H, G, F, D, C, B, A and the resulting bits of the code are named a, b, c, d, e, i, f, g, h, and i. The bit order is reversed for serial transmission.
8b/10b coding is DC-free, meaning that the ratio between the number of 0 and 1 transmitted is 50 %. This is done by constantly reversing the so called disparity (RD), which can be neutral, positive or negative depending of the numbers of 0 and 1 sent. For this reason some 8-bit inputs have two 10-bit codes like the example depicted in fig. 4.4 showing the encoding of the hexadecimal value 37.

4.4 μTCA System

The DATCON algorithm is intended to be running on FPGAs, hence a carrier board or PCB, power supply and connectivity are required. There are many powerful test kits embedding FPGAs. As they are not meant for scalability, their use and interconnection can quickly become disordered and the adoption of a communication standard is preferable. Such a solution must nevertheless fulfill the following criteria:

- Right choice of FPGA chip capable of running the entire firmware. For DATCON, Virtex 5 and 6 are used.
- Offering optical connection via SFP cages up to 3.25 Gpbs at a wavelength of 850 nm. This is required for communication with FTB and ONSEN.
- High speed connection between the FPGAs.
- JTAG access for configuration and debugging.

Because of its back-plane bandwidth limitation, a Versa Module Europa (VME) based system was rejected. A few projects withing the Belle II collaboration have their systems based on the ATCA standard which fulfills all the requirements. An improved and more flexible version of ATCA, called μTCA was selected for DATCON.
μTCA is a modular standard created by PCI Industrial Computer Manufacturers Group (PICMG) [36]. Designed as a complementary system of ATCA, and originally built for telecommunication applications, μTCA has quickly created interest also in others fields and is nowadays widely spread. The main component of a μTCA system is the chassis, or shelf which

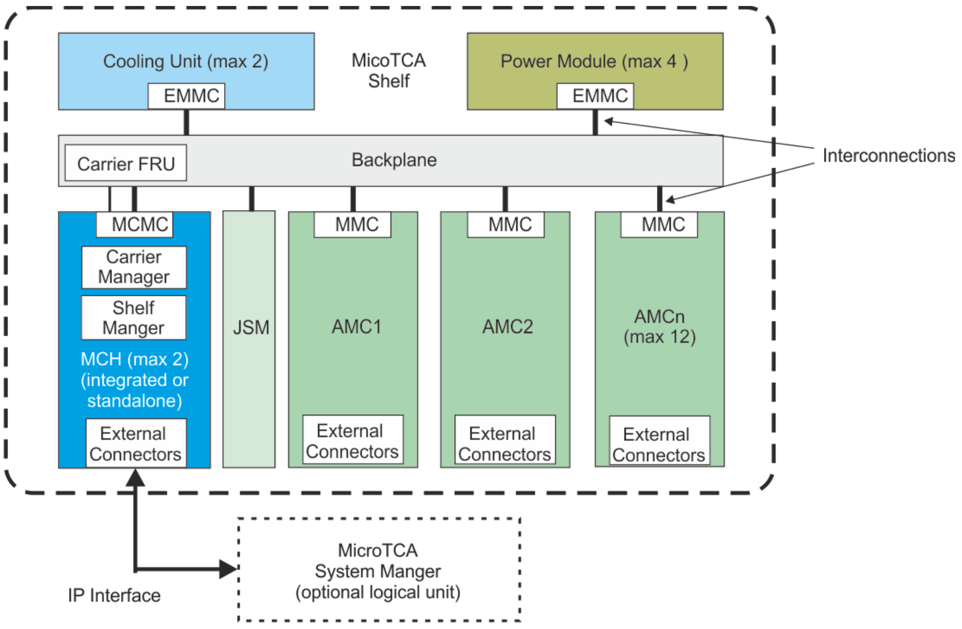


Figure 4.5: μTCA functional block diagram. AMCs are connected directly to the backplane. Power module and MCH are indispensable components [37].

provides the physical support for all the sub-components. Its organizational structure is depicted in fig. 4.5. Data transmission and power connection are available via the backplane.
The chassis offers slots into which AMC cards can be plugged. They were originally designed for the ATCA system and grouped in mezzanine into the carrier. In a μTCA architecture, with mini-blade format, they are directly connected to the backplane. AMCs offers μTCA all its modularity and scalability. The brain of the chassis is the MCH, it is in charge of controlling the different modules of the system such as AMCs or Power Module (PM). The PM is an intelligent module that converts the supplied power and distributes it through the chassis. Just as the MCH, PM can be redundant in the system. The MCH and PM chosen for the DATCON chassis are from N.A.T Europe. [2].

[2]N.A.T Europe is German company developing high-performance network interfaces for industrial computers, most notably components for μTCA standard.

4.5 DATCON AMCs

For the DATCON project, three different types of AMCs are in use, two of them are embedding an FPGA. The first kind is the AMCv3, also refereed as Concentrator. It is directly connected to four FTBs fibers via the available SFP cages. It features two Double Data Rate 2 (DDR2) memory slot as well as an Ethernet port necessary to interact with the board via slow control. The Concentrator is placed at the beginning of the DATCON processing chain. Its task is to unpack the incoming SVD data, filter and cluster strip information before transmitting them to the rest of the system for tracking. The board, developed by the IHEP Beijing, in China, includes a vlx50t Virtex 5 FPGA from Xilinx. The FPGA is cadenced with a 127.21 MHz oscillator which is the standard Belle II DAQ clock frequency. The small Virtex 5 of the

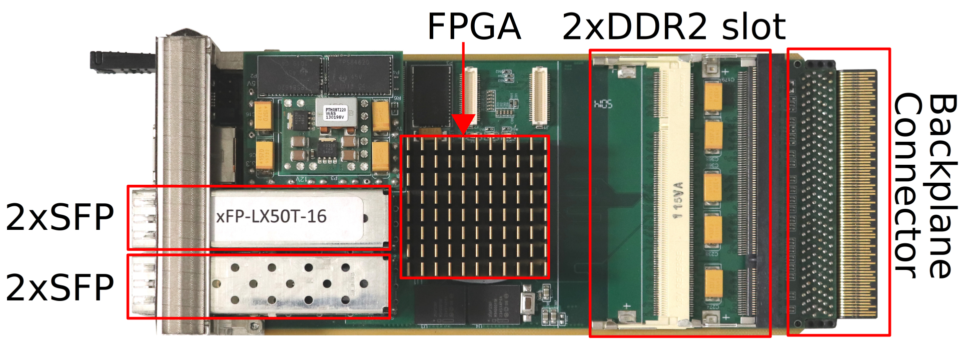


Figure 4.6: Concentrator AMC

Concentrator does not have the resources necessary to perform the track finding. For this purpose, a second type of AMC is needed. It is based on the DHE used for the PXD DAQ (see section 2.3.3) but equipped with a larger FPGA. The Tracking AMC for DATCON uses a vlx240t Virtex 6 from Xilinx which offers a quarter million logic cells and up to 12 high speed GTX transceivers. The board is mounted with two oscillators. To communicate with the Concentrators, the same 127.21 MHz oscillator is used. A second one, generating a differential 125 MHz clock, is used to for the transceiver dedicated for ONSEN communication. The RJ45 and camera link connector of the DHE are not required for DATCON purposes thus, only the backplane is used to communicate with the rest of the system.
If communication with the Concentrators is done electrically via backplane, connection to slow control and to ONSEN has to be made with Ethernet cables and optical fibers. Accordingly, an Extension board, complementary to the DHE is needed. The Extension AMC routes the backplane signal to SFP cages into which optical transceivers can be inserted. The card features eight cages whose of which only four are used. As mentioned, one is to communicate with the slow control and one is the actual DATCON output through which ROIs are sent. Final two connections are to exchange dispatch data between the two chassis. Both, the Tracking and

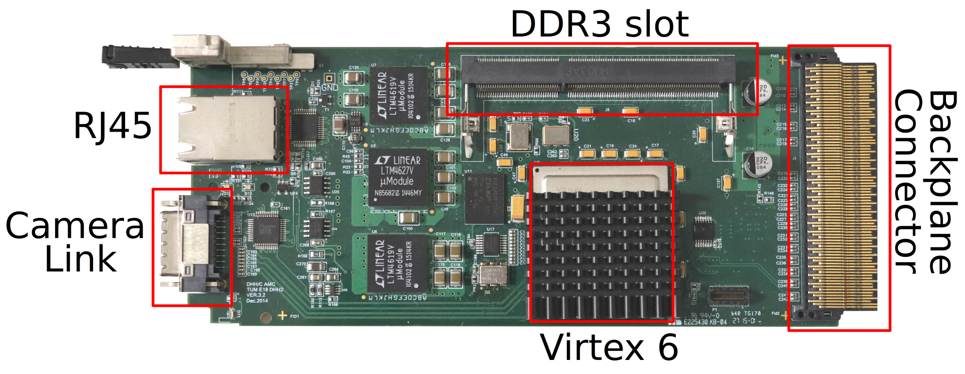


Figure 4.7: Tracking AMC

Extension AMC were designed and provided by Technische Universität München (TUM) [38].

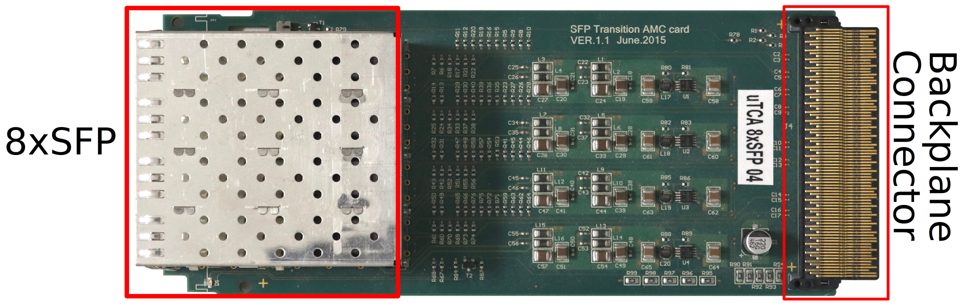


Figure 4.8: Extension AMC.

4.6 Custom Chassis

The chassis chosen to hold the previously introduced AMC is based on a commercial nine-slot μTCA chassis designed by Schroff [3]. The original retail product had to be modified to comply with DATCON hardware's specifications. The backplane port routing as well as a solution for programming the FPGAs are discussed in this section.

4.6.1 Backplane

The original commercial backplane configuration of the chassis does not fulfill the requirements for DATCON based on the AMC port configuration. As shown in section 4.1, all Concentrators have to be connected to the Tracking unit. Prior to purchase, a new backplane scheme was designed and implemented by Schroff. All ports #6 of the Concentrator cards are connected

[3]Schroff is a German company building chassis and equipment racks for a variety of markets such as telecommunications and data centers.

to the Tracking board. The backplane topology and connections are depicted in fig. 4.9 The

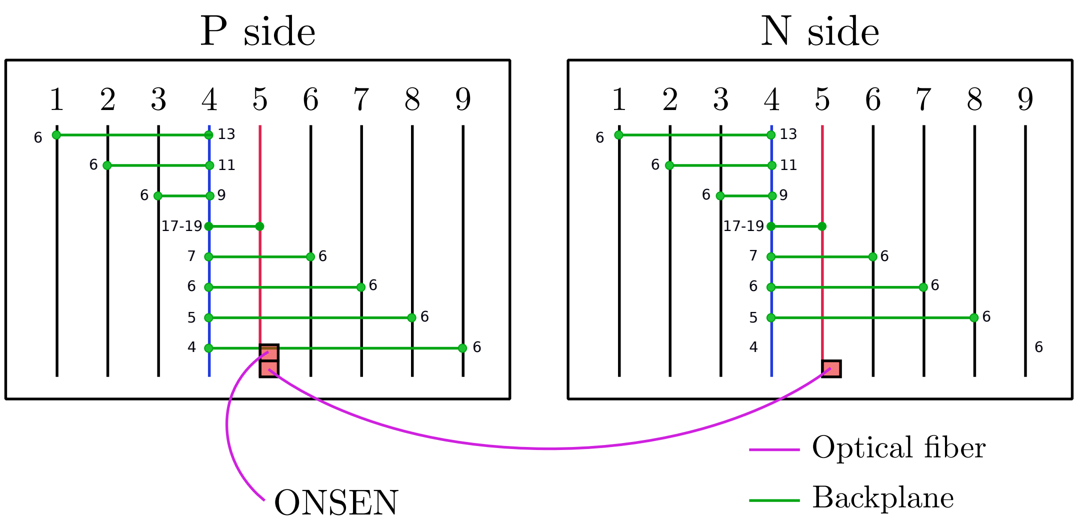


Figure 4.9: Backplane topology and connection for DATCON. Concentrators are plugged into slots 1, 2, 3, 6, 7, 8 (and 9). Via the backplane, their ports #6 are connected to the Tracking AMC located on slot #4. Tracking boards are connected to the Extension AMC located on slot #5. It offers optical connection between chassis and to ONSEN.

Tracking AMC is located on the 4th slot and is linked via the backplane to the 5th slot where the Extension board is located. Every other slot is occupied by one Concentrator card.

4.6.2 JTAG Access

The option which was chosen for the DATCON carrier, which conforms with the μTCA standard and uses custom backplane requires an additional change. A JTAG access to the on-chip Test Access Port (TAP) of all the FPGAs was implemented by modifying the chassis backplane PCB.

A JTAG connection is necessary to program and to debug the FPGAs as described in section 4.2. None of the AMCs natively includes a JTAG connector on the front plate or on the PCB itself. The node can only be reached via the backplane connector of the card.

The JTAG Switch Module (JSM), using the backplane, is a usually chosen solution for JTAG access. It can be placed in any AMC slot allowing the user to flash the desired AMC. This method was tested with the DATCON AMCs, but unfortunately with no success. The DATCON AMCs follow most of the μTCA requirements but it does not suffice to be paired with a JSM. Since every slot on the $\mathcal{P}$ chassis is occupied by an AMC, it does not leave space for an additional JSM module, therefore excluding this choice.

The first idea was then to re-route the backplane JTAG connections of each AMC to an individual JTAG connector in the back, resulting in as many connectors as there are AMCs. The

simplicity of this solution would have required a large amount of JTAG programmers. For this reason, a different approach was adopted, maintaining the constraint of JTAG connector placed in the back of the chassis.
The JTAG interface allows multiple devices to be connected to a single controller in a daisy-chain configuration. It is commonly based on a four-pin interface, pins are respectively named TMS, TCK, TDI and TDO. The TMS signal controls the JTAG state machine, while TCK corresponds to the clock distributed to each device. In a chain, the output, TDO, pin of an element is connected to the input, TDI, of the next until it comes back to the controller to close the loop.
A successful boundary scan requires that the chain stays continuous. If a TAP is missing, the chain breaks and the data can not be transmitted. It can happen, for example in the $\mathcal{N}$ chassis, that a slot is not occupied by an AMC. Nonetheless, it should not disrupt the loop. The solution was to completely remodel the JTAG routing onto the backplane PCB directly by using bypass switches [4]. Consequently, when an AMC is not detected, the TDI signal is bypassed and directly transmitted to the next slot. On the contrary, when an AMC is plugged in, the bypass chip is not activated and the TDI signal is normally forwarded to the FPGA. One of many pins of the AMC connector is the PS1 pin which is used to inform the PM that a card is currently inserted and that is should be turned on. The PS1 signal is controlled by the Module Management Controller (MMC) [5], the lowest management entity of a μTCA

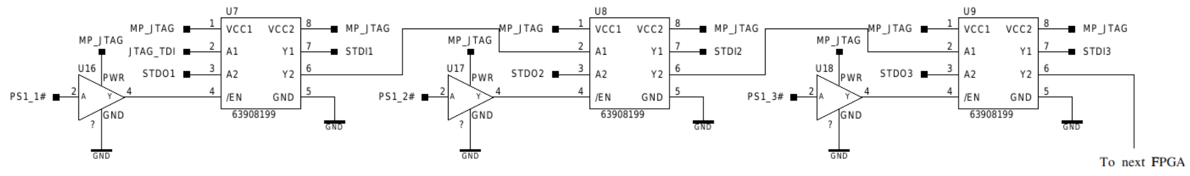


Figure 4.10: JTAG daisy chain bypass schematic. The PS1 signal controls the *enable* pin of the bypass switch. If an AMC is plugged, the input TDI signal must go to the AMC. On the switch, A1 input is thus routed to output Y1. The TDO of the AMC connected to the switch input A2 is sent to the next AMC via the output Y2. If there was no AMC plugged, the input A1 is directly routed to the output Y2. The loops stays close.

system, present on all the Concentrator AMCs. The Tracking boards do not yet have an MMC and adjustments were directly made on the PCB to set the PS1 pin to high therefor forcing the card to be detected. The PS1 pin is used as the enable signal for the bypass switches,

[4]The low voltage ADG3233 switch from Analog Device is used.
[5]MMCs are responsible for the management of individual AMCs providing the link to the MCH and transmitting monitored data via IPMI.

its connection along with the daisy chain schematic are shown in fig. 4.10. To distribute the JTAG clock to every port with a minimal skew, two LVCMOS fanout buffers were added.

4.7 DATCON Installation

After shipping the hardware components of DATCON to KEK, the system was successfully installed. The hardware is located in what is called the eHut which is located right next to the Belle II detector. The SVD readout system is, on the other hand, located on top of Belle II . It is divided into two crates with one located in the forward and the other one in the backward region.

Optical patch panels with reserved slots for DATCON are installed on top of Belle II in the forward region and also in the eHut. They are connected together by MPO cables which contains 12 independent channels, reducing the number of cables going from the top of the detector to the eHut.

Considerable work on fiber installation was therefore necessary. The first part was to connect the 52 FTBs to the patch panels. Individual fibers were packed in groups of 4 into sheatsh to simplify pulling cables under the floor above Belle II .

In the eHut, the patch panel located in the same rack as the DATCON chassis simplifies the fiber routing. Short fibers of 1 and 2 meters were sufficient to finalize the connections.

Two fibers connect the chassis together while another communicates with the ONSEN system located on the neighboring rack to the left of DATCON.

Every AMC card is connected via Ethernet to the Slow Control (SC) switch located under the chassis. The complete DATCON system is shown in fig. 4.11. The entirety of the fibers are connected to the 13 Concentrator cards (2). Located in the 4th slot of each crate, the Tracking board, to which the extension AMC (3) offers connectivity, is the core of the system.

On the picture, a card (4) used by ONSEN appears, it was installed as an emulator during the early stages but is not currently used.

A Personal Computer (PC) dedicated to DATCON is used as access to take control, program or debug the hardware. The PC is also running the SC Input/Output Controllers (IOC) whose process is detailed in Chapter 8.

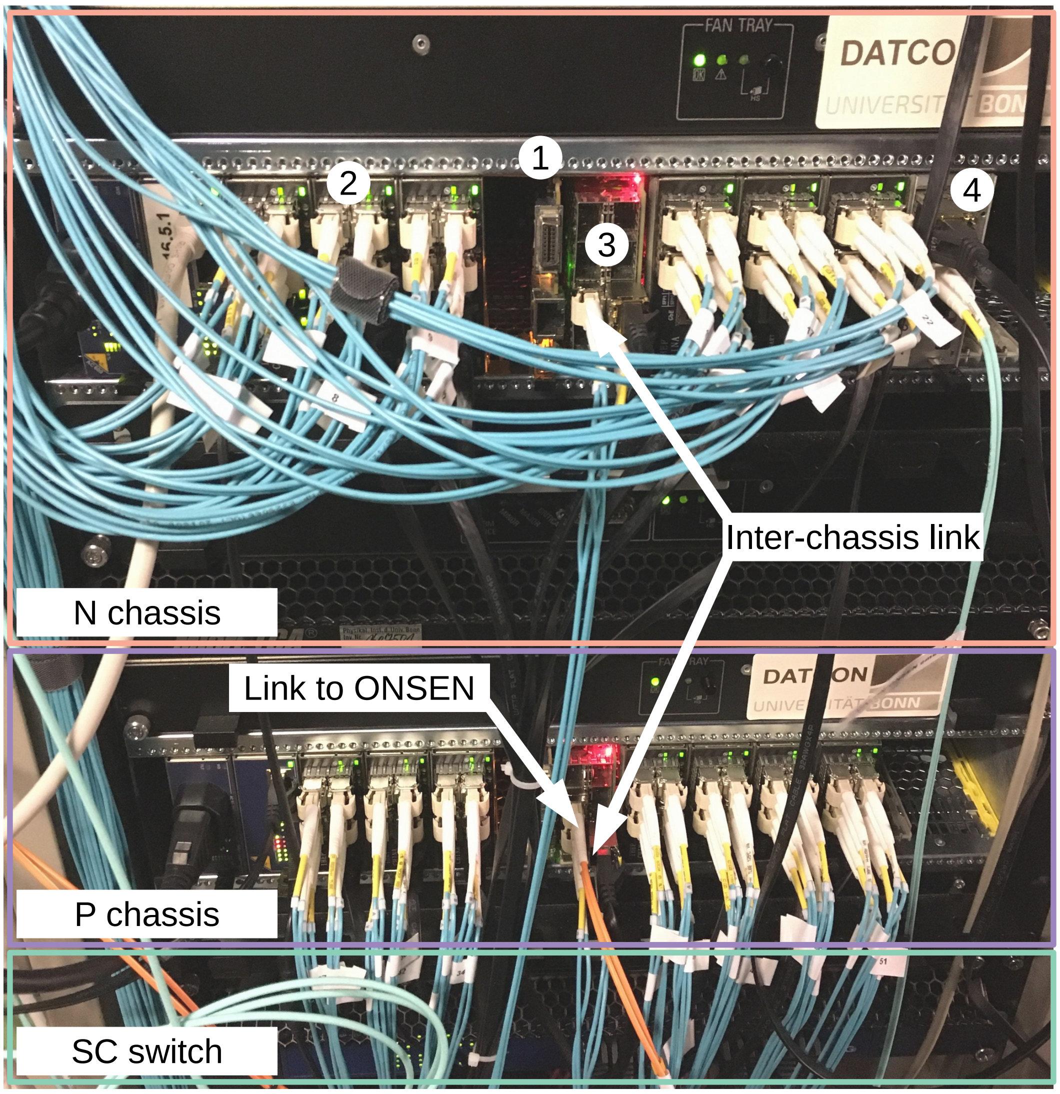


Figure 4.11: DATCON running at KEK with the 52 links to FTBs. The $\mathcal{P}$ and $\mathcal{N}$ chassis are inserted into a rack alongside with the SC switch.

5 Data Aquisition and Preprocessing

In this chapter, the different steps used to process data provided on the SVD side by the FADC and via the FTB (see section 2.2.2) are presented. Although FTB is a part of the SVD system, it is also the first stage of DATCON's processing chain. A section of the FTB firmware is dedicated to encapsulate the FADC data into a specific format and send it via optical fiber to DATCON.
On the DATCON side, the received raw SVD data, must transit through several elements before it can be used for track finding.

- First, the incoming FTB frames are received and unpacked.
- The strip signals are then filtered before performing the cluster detection.
- Finally the cluster position is sent out to the Tracking unit.

All these steps are performed by the Concentrator AMC, presented in section 4.5, and are depicted in fig. 5.1.

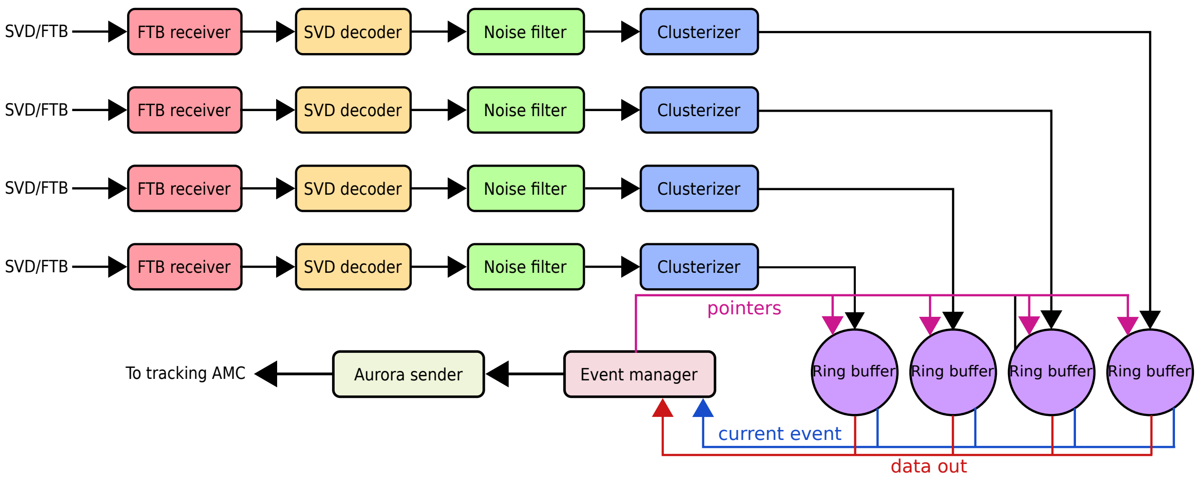


Figure 5.1: Functional block diagram of the Concentrator firmware including four identical data processing channel which consist of a frame receiver, a frame decoder to extract strips and their signal, a noise filter and a strip clusterizer. The entirety is controller by the event manager.

Event building plays an important role for DATCON. Concentrator links are processed independently and data are not guaranteed to arrived in sync. Therefore it is necessary to ensure that event number and corresponding data remain linked throughout the entire processing chain.

5.1 FTB Firmware

The FTBs are designed for optically transmitting the SVD data to COPPER and DATCON. This section focuses only on the fraction of its firmware which is dedicated to DATCON.
The FTB first checks for possible errors in the incoming FADC packets by checking the Cyclic Redundancy Check (CRC) and the presence of the protocol header and trailer. Provided by the FTSW, the event and run numbers are used to build the FTB protocol header which is then followed by the FADC data. Finally, a CRC of the current packet is calculated and added. The newly built 32-bit data words are completed with 2 bits serving as indicator of the start and end of an event. They are then stored in a DATCON dedicated 34-bits wide First-In First-Out (FIFO). A module, called DATCON-link framer, is responsible for reading out the FIFO and transmitting its output data to one of the Concentrator AMC's input at the rate of 1.27 Gbps. The transmission is based on the Aurora protocol introduced in section 4.3.

5.2 Concentrator Firmware

The preprocessing of the SVD data for Tracking and ROI calculation is performed on the Concentrator AMC. A total of 13 Concentrators cards are used to connect with the 52 FTBs. Data are received on each AMC by four optical links, from which they are independently processed in a fully pipelined chain. This section discusses the main components of the Concentrator firmware.

5.2.1 Receiver

The four inputs of the Concentrator are processed in the same way and are independent from each other. Therefore, this section focuses on a single processing chain only. As mentioned previously, the FTB-DATCON link is implemented using the Aurora protocol with a bandwidth of 1.27 Gbps. The receiver, refereed as the b2link receiver, is the bridge between the Aurora Locallink and the rest of the firmware logic. A detailed view of the frame format is depicted in fig. 5.2
The receiver is intended to sort the incoming data. On one side, the strip and hit data, on the other, the event and global run numbers. Verification of the header, controlling CRC and possible error detection is another functionality of the receiver. Originally, propagating the

run number was not needed but it was later requested to be transmitted to ONSEN. The run number was therefore positioned before the header in the data frame.
Once the header and the CRC are discarded and the run and event numbers thoroughly recovered, only the FADC data is left. Data words are 16 bits long, same as the Aurora lane width. They are stored in a 32-bit output FIFO which is automatically read out as soon as it is not empty.

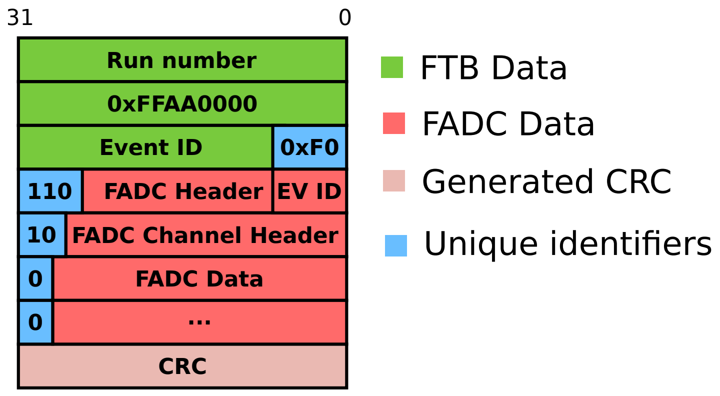


Figure 5.2: FTB-DATCON data format. Header, run and event number (green) are added to the FADC data (red). A CRC is computed for every transmitted packet.

5.2.2 FTB Decoder

After the receiver has collected the FADC data and guaranteed its integrity, the decoder extracts the different information such as the FADC sample mode, the strips and their signal samples values. The hit information is composed of a strip ID, an APV25 and an FADC ID, as discussed in section 2.2.2. Figure 5.3 shows the detailed format of an FADC frame to be processed by the decoder. Strip samples are contained in one or two words depending on the sample mode used.

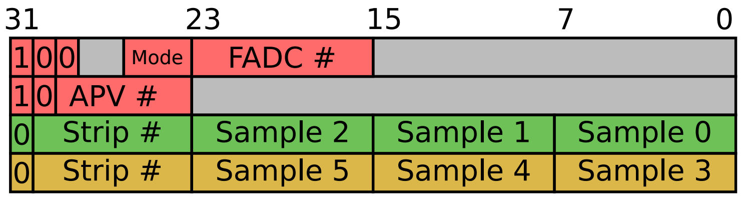


Figure 5.3: Detailed format of an FADC frame to be processed by the decoder. Strip samples are contained in one or two words depending on the sample mode used.

The decoder Finite State Machine (FSM) first checks the three MSBs [1] of the incoming word to confirm it is a valid FADC header. From this, the mode is extracted to decide if three or six SVD samples have to be used. From the header, the FADC board number is extracted. In the next step, the APV chip number is returned after controlling the 2 MSBs of the current word. FADC and APV IDs are an absolute necessity to later locate the current strip to specific SVD sensors and obtain their spatial coordinates.
The decoding of one hit signal is made in one or two clock cycles depending on the mode, three or six-samples. Regardless of the mode, it starts with the first three samples. The peak value within those samples is extracted, its position and value are stored in a register. In the case of six-sample data, the process is repeated for the next three samples on the following clock cycle. The peak value is once again returned and compared to the one previously found. The maximum of the two becomes the current peak value of the hit. For the three-samples mode, the last step is skipped. Independent of the hit being provided with the three or six-sample mode the following step is identical. The hit has to be filtered to estimate if it is noise or valuable data.
The FSM continues for every new incoming hit until the FTB trailer has been detected. The trailer indicates the end the current event.

5.2.3 Noise Filtering

With the aim of reducing the SVD occupancy by removing noisy strip data and keeping only the hits produced by the passage of a particle, it is necessary to filter the incoming FTB data. As introduced in section 2.2.2, the slope of a strip signal follows the one of a CR-RC shaper whose transfer function is:

$$V_o = V_i \frac{t}{\tau} e^{-\frac{t}{\tau}}, \tag{5.1}$$

where τ corresponds to the 50 ns shaping time of the APV25, V_i and V_o are the input and output voltages and t is the time. It is the time-equivalent of the standard deviation of the shaper input pulse. With cubic splines and an exponential fit of the waveform, the peak time and value can efficiently be extracted thus removing off-time hits. Implementing such a solution is, however, outside the scope of this thesis. As a consequence, a more basic solution is implemented and detailed in this section.
Considering the case of six-sample mode, the strip information is contained in two words as shown in fig. 5.3. Two steps, in which three samples are processed, are necessary. As mentioned in the previous section, the peak sample and position are obtained. A simple shape filter ensures that the peak is placed at position 2, 3, 4 or 5. It also computes a neighbor

[1]Most significant bit, in a binary representation, is the furthest to the left with respect the convention in positional notation.

threshold by shifting the peak value by one bit to the left. For validating that the samples distribution follows the one of a CR-RC shaper, the left and right samples of the peak must be over the new threshold. The effect of this approach is depicted on fig. 5.5. Including early Phase3 backgrounds, 5×10^3 events were simulated to obtain FTB output. The details of the event simulation are given in Chapter 7.
In parallel to the waveform shape validation, the peak value alone can also be compared to a low and high threshold. Originally a valid peak should be between 4 and 180 ADCs. An example of a valid set of samples is depicted in fig. 5.4, where the third sample has the highest value while being contained within the defined range. If any of the three conditions introduce fail, peak position, neighboring threshold and range threshold, the strip is rejected. The last

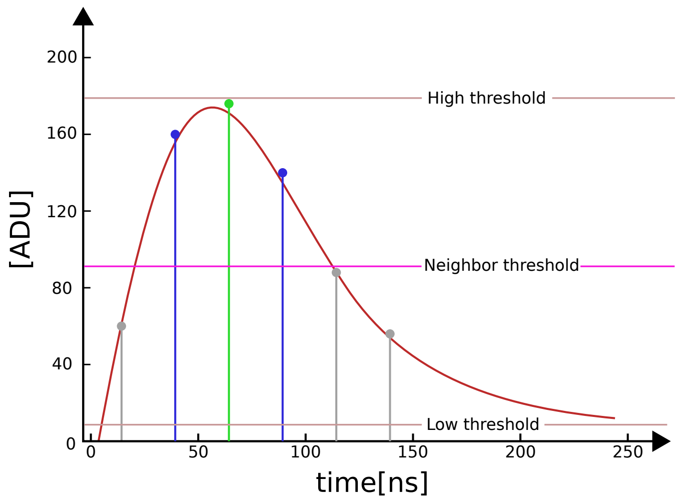
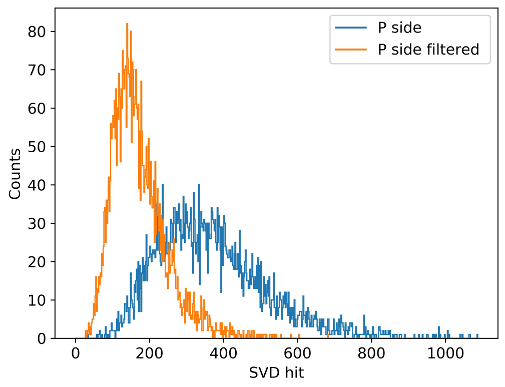


Figure 5.4: Six-mode APV samples of a valid strip signal. The peak sample (green) is in position #3 and is contained between the two thresholds. The waveform shape is respected.

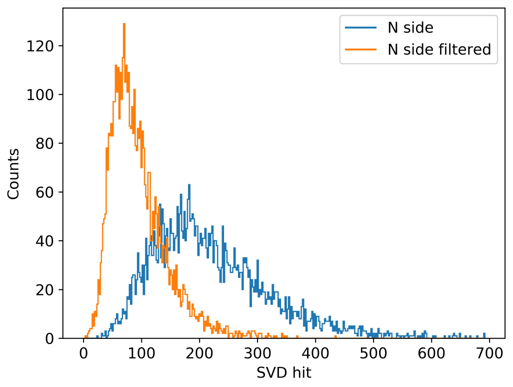
option for filtering the incoming strip signal is by using their SNR. After a run, it is possible to obtain a list of the SVD strips with a high recorded noise in ADC units. From the noise, the expected SNR can be calculated and it becomes the new minimum threshold to accept a strip. A good peak signal is expected to have at least SNR of 5. A Python script computes the new threshold values from recorded noises and place them into a Verilog LUT file.
During the strip filtering, the peak value is returned and compared to the LUT output for the specific strip. If the signal is lower that the returned threshold the strip is rejected.

5.2.4 Clusterizer

When a particle passes through the strip detector, depending on its angle and energy, more than one strip can be activated. A group of neighboring strips is called a cluster. For this



(a) Number of $\mathcal{P}$ SVD hit before and after the sample shape filter.



(b) Number of $\mathcal{N}$ SVD hit before and after the sample shape filter.

Figure 5.5: Effect of the sample shape filter on the number of SVD hits.

reason, when a strip passes the noise filtering it is necessary to investigate if it has active neighbors and if it can be included into a cluster. Before doing so, the strip is converted into a global strip, with regards to the SVD sensor. Indeed, so far the strip number is relative to an APV and FADC pair. An XML file contains the mapping of the strips on an SVD sensor with the APVs and FADCs. It therefore becomes possible to precisely locate the current strip on a specific sensor. The mapping directly reflects the physical connection of the SVD strips to the readout chips.
With regards to the APV only, a strip is within the range [1-128] and is converted into a global one in the range [1-512/768] depending on the SVD $\mathcal{P}$ or $\mathcal{N}$ side. The XML mapping is firs converted into a Verilog file which later included into the firmware. During this step the FADC and APV numbers are used to generate a unique 16-bit sensors ID computed as follow:

$$sensor_{id} = (layer_{num} - 3) \ll 12 + ladder_{num} \ll 4 + sensor_{num} \tag{5.2}$$

Each layer of the SVD is composed of up to 5 sensors referred as $sensor_{num}$. Once the strip is linked to its sensor ID, the clusterizer FSM can perform its task. The first incoming strip is chosen as the seed for the current cluster. On the next valid strip, the absolute distance, in strip, is computed and if it is exactly 1, the strip is added to the current cluster. The operation is repeated until the distance is at least 2 or until the new strip does not belong to the current sensor. The clusterizer output resolution is therefore in strip and not spacial coordinates. The conversion is done during the tracking process.

5.2.5 Event Management

On every Concentrator, the four inputs from the FTB do not receive data synchronously. In addition, the time necessary to process the incoming SVD signal may vary from one link to another. An example is provided in fig. 5.6. The event manager is a key component whose task is to assure that for a specific event, the corresponding Concentrator data are sent synchronously to the tracking unit. If it slightly differs depending on its use, the event management is present in every Concentrator and also every Tracking board where the different backplane links have to be synchronized. Its principle is discussed in this section based on the Concentrator case. The SVD strips obtained after the decoder, noise filter and clusterizer are

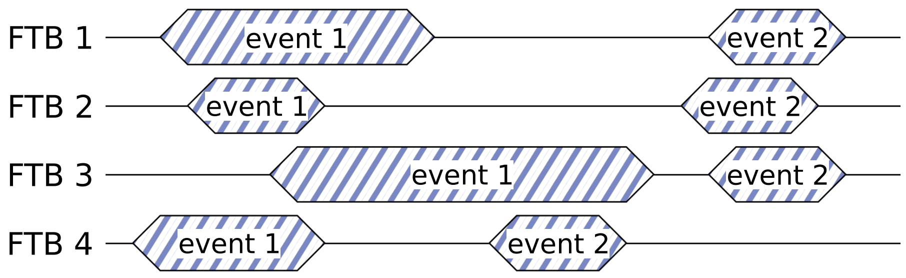


Figure 5.6: Example FTB-DATCON frame diagram where the start of a frame or its length is not pre-defined. On link FTB 4, the second has arrived before the link FTB 3 has finished receiving the first event.

stored in a ring buffer based on a true dual-ported block memory using the available FPGA memory. Input data are 32-bit word and output only 16-bit to match the link size that is used to send out the Concentrator data for track finding. For every channel, a 32 kbit memory is used. A total of eight ring buffers are required, one for each $\mathcal{P}$ and $\mathcal{N}$ side of all four inputs. The ring buffer requires three more memories of depth n. The first one contains the event number, the next two are used to store the *start* and *end* address of an event in the main Random Access Memory (RAM).
The value of n is currently set to 512. It corresponds to the limit of the block memory available on the FPGA. This means that the ring buffer can at its maximum occupancy, store the strips of 512 distinct events. The event manager handles eight pointers that are used as addresses for the sub-memories of each ring buffer. The current event IDs, corresponding to the current pointer, are compared to certify that events are not mixed. One by one, all the ring buffers are read out based on their current *start* and *stop* addresses, after which all the pointers are incremented. The event size is determined by summing up the memory occupancy of each circular buffer. To keep the principle of separating $\mathcal{P}$ and $\mathcal{N}$ side, the four $\mathcal{P}$ buffers are read out first followed by the $\mathcal{N}$ buffers. Once an event is complete and stored in the ring buffers,

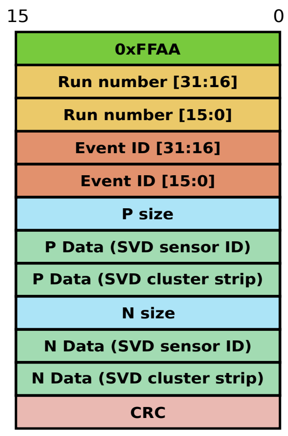


Figure 5.7: Concentrator-to-Tracking protocol. After a header, the run and event number are sent and are followed by the number of strip, or event size which precedes the $\mathcal{P}$ and $\mathcal{N}$ clustered strips.

it can be sent to the Tracking board. The connection is made electrically via the back-plane and the Aurora protocol is used at a rate of 1.27 Gbps, the same as for the FTB-DATCON links. The output of the event management is encapsulated into a custom protocol depicted in fig. 5.7. The frame start with a header indicating that the incoming packet contains the strip of a new event. The event and run number directly follow. Before encapsulating the actual data, it is necessary to attach the event size, or the number of strip clusters. The transmitter sends the data as they are provided by the event management unit, meaning the $\mathcal{P}$ side hits first, followed by the $\mathcal{N}$ side hits. For every event, even in the case of no clusters, a frame is sent to the Tracking part. The words for the sizes are consequently set to zero.

6 FPGA Implementation

As introduced in Chapter 3, the track finding principle, based on the Hough transformation along with the extrapolation methods and ROI calculation need to be implemented in the FPGA. The previous chapter presented the indispensable preprocessing steps to provide inputs for the track finding. In this chapter, the implementation on the FPGA of the algorithm along with the inherent logic is discussed.

6.1 Demultiplexing and Data Management

Since there are multiple connections with the Concentrators, but also between the chassis, this section describes the data management, how the inputs synchronization is achieved and events are built. The track and ROI finding are discussed later in the chapter.
The tracking AMCs are connected to seven Concentrators on the $\mathcal{P}$ side and six on the $\mathcal{N}$ side. As it has been shown in section 5.2.5, the incoming data contain both $\mathcal{P}$ and $\mathcal{N}$ clustered strip information. Every Concentrator input has its dedicated receiver whose task is to unpack the incoming frames and forward the data to either a $\mathcal{P}$ or an $\mathcal{N}$ side ring buffer.
The buffer's implementation is similar to the one used for the Concentrator. It is based on a 4 kbit RAM that can store up to 512 events. There are twice as many buffers as there are Concentrator inputs to cope with the two strip types. Data are written to the buffers with their corresponding event numbers and are read out upon request. Two different clocks are used on the FPGA. The first one is set at 63.53 MHz and is based on the transceiver rate used to communicate with the Concentrators. This clock is used to write data into the buffers. The second clock, running at 78.125 MHz, is based on the transceivers instantiated for ONSEN communication and is used for reading the buffers. Because of the two running clocks, a clock domain crossing is implemented in the buffers. The rest of the firmware is therefore synchronized with the 78.125 MHz clock. The different links from the Concentrators are not expected to be in sync as Concentrators automatically send data once an event is built and ready. Event-related data have to be regrouped and processed as a whole. This step, referred to as «event building», is repeated three times at different levels. An overview of the data handling and event building is shown in fig. 6.1.
The first level synchronizes the Concentrators clustered strips. From the perspective of the $\mathcal{P}$ chassis, two event types are built for each strip side. The $\mathcal{P}$ event remains in the buffers

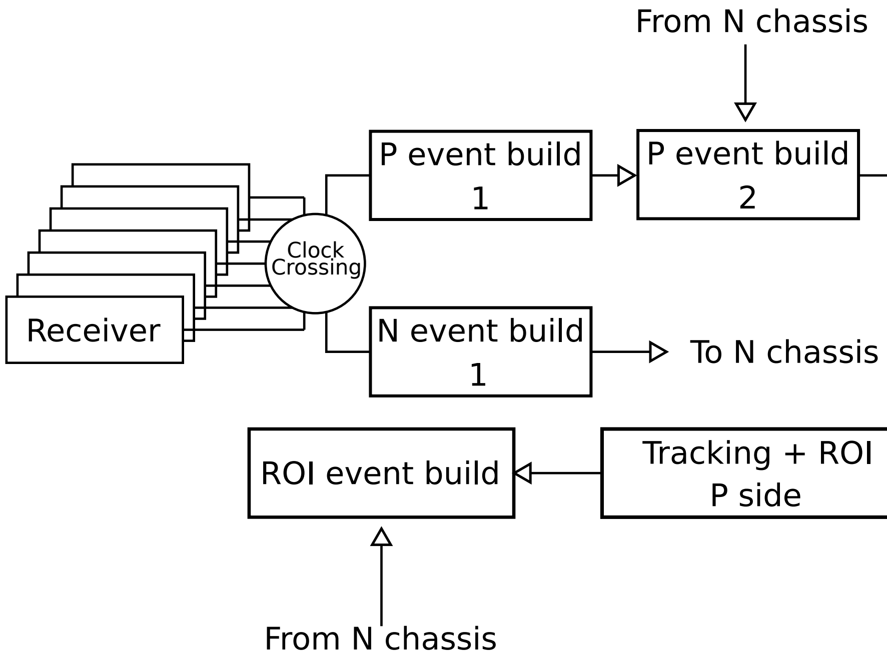


Figure 6.1: Block diagram of data management and event building on the $\mathcal{P}$ chassis.

whereas the $\mathcal{N}$ event is sent to the $\mathcal{N}$ chassis. At the same time, the $\mathcal{P}$ event is received from the $\mathcal{N}$ chassis.
Then, the second event builder comes into play. Its task is to synchronize the local ($\mathcal{P}$ data from $\mathcal{P}$ chassis) and external ($\mathcal{P}$ data from $\mathcal{N}$ chassis) data. Each DATCON unit performs this step with its corresponding type of strip data.
At this moment, strips are correctly redistributed and track finding together with ROI calculation is performed. The resulting local $\mathcal{P}$ ROIs must be combined with the $\mathcal{N}$ ROIs, thereby a final event building is performed with the local and external ROIs received from the $\mathcal{N}$ chassis.

6.2 Convertion to Spacial Points

The strip clusters have to be converted into spacial coordinates in order to compute the track finding and reconstruction. The coordinate system used for DATCON is described in section 3.4. It should be noted that every strip sent from the Concentrators produces two distinct pairs of coordinates on both $\mathcal{P}$ and $\mathcal{N}$ side.
A part of the available 14.98 Mbits of block-RAM on the FPGA is used to implement a Read Only Memory (ROM) that stores all the coordinates. Each sensor has a maximum of 768 strips which can then be placed in banks of 1024 coordinate pairs. Since the resource needed to store the strip coordinates is low compared to the total memory available, the usage of over-sized banks simplifies memory addressing by bit shift. The sensor ID attached to a strip

gives information about the layer, the ladder and finally the sensor number. Four base addresses are used for each layer and are stored in the *ptr_init* register. Depending on the strip orientation, the ladder for $\mathcal{P}$ or the sensor number for $\mathcal{N}$ is used to select the bank. The strip number automatically leads to the corresponding position in the selected bank.
Address pointers can therefore be obtained depending on the tracking side following the equations:

$$\begin{aligned} ptr_{P\_side} &= ptr\_init(layer) + (ladder - 1) \ll 1024 + strip\_id \\ ptr_{N\_side} &= ptr\_init(layer) + sensor\_num \ll 1024 + strip\_id \end{aligned} \tag{6.1}$$

Since the magnetic field only affects the trajectories of particles in the r-φ plane, or simply $\mathcal{P}$ side, the strip converter directly returns the coordinates after the conformal transformation, as seen in section 3.2. For the $\mathcal{N}$ side, only (y,z) coordinates are used.
It was decided, for converting the coordinates as well as for all the rest of the mathematical computations, to use signed fixed point numbers. Based on the IEEE 754 standard [39], floating point representation is feasible on FPGA and is far more precise but its use implies additional complications. Even simple arithmetic operations such as addition or multiplication requires the implementation of complex logic.
A dedicated C program is written to compute the coordinates of every strip according to section 3.4 and generate COE files [1] ,necessary for initializing ROM. Proportional to the strip pitch, the coordinates are expressed in mm. Due to the need for integer representation and to avoid precision loss the coordinates were multiplied by 10^3.
For the $\mathcal{P}$ side, where the conformal mapping is used, it is slightly different. Still following the idea of avoiding floating point numbers, the result of the mapping is multiplied by 10^6 and rounded to the closest integer.

6.3 Fast Hough Transform (FHT)

As discussed in section 3.1, a set of hits that belongs to the same track will have intercepts in the HS. To build the HS, every hit has to be computed following eq. (3.3). This section describes the solution that was developed to build the HS on FPGA while profiting from its parallelism.
For every event, the HS are built and require their dimensions to be defined and hard-coded. The size and resolution of the HS are predefined, hence the horizontal (φ and θ) and vertical range (d) are chosen, as well as the number of sectors necessary to cover the entire HS. The size of each sector has a direct influence on the track finding efficiency which is further discussed

[1] The Xilinx COE file is used to initialize ROM and RAM content. It consists of of a list of values of the memory cells written in a specified radix.

in Chapter 7. If large sectors automatically increase the number of track candidates, track fitting will degrade. On the other hand, a very fine granularity will make track finding more resource onerous, but increase the accuracy of extrapolated tracks. Thus, a compromise had to be made.

For both, $\mathcal{P}$ and $\mathcal{N}$ side, it has been decided to use a HS of size 128×64 cells with $\varphi \in [-180°, 180°]$ and $\theta \in [20°, 157°]$. The HS construction process is the same for each tracking side, thus its functioning is only described for $\mathcal{P}$ side.

The part of what is covered by a sector is the full range divided by the number of sectors. Every column of the HS is thenceforth defined by a lower and an upper limit φ_1 and φ_2, with φ_2 being the lower limit of the next sector. The main component of the HT is a sequencer which, in its basic form, is defined with 128 main states for the 128 columns. Each main state is composed of three sub-states.

In every main state, denoted with $h \in [1 : 128]$, the *cos* and *sin* values of φ_1 and φ_2 of the current column are returned. Xilinx offers solutions for computing trigonometric functions. However, for saving computation time and resources it has been chosen to precalculate and hard-code the *cos* and *sin* values for every state. The small number of values to be stored makes this decision for this solution favorable. The values are consequently referred to cos_{1_h}, cos_{2_h}, sin_{1_h} and sin_{2_h}. The sine and cosine are sent to a sub-module known as Sector Check Module (SCM).

Its function, based on embedded Digitial Signal Processor (DSP)s, is to calculate the result of the HT for every incoming hit x_i,y_i. The pair y_{1_h}, y_{2_h} is calculated following the equations:

$$\begin{aligned} y_{1_h} &= x_i \cos_{1_h} + y_i \sin_{1_h} \\ y_{2_h} &= x_i \cos_{2_h} + y_i \sin_{2_h} \end{aligned} \tag{6.2}$$

An illustration of the sector finding is depicted in fig. 6.2 To go to the next step $h+1$, the HT has to be computed for every hit of the current event.

The second task of the SCM is to determine which one of the 64 vertical sectors v of the column h is passed by a line. To determine if a sector contains the result of the HT the following Boolean condition is used:

$$\neg(y_{1_h} > y_v \land y_{2_h} > y_v) \lor (y_{1_h} < y_{v+1} \land y_{2_h} < y_{v+1}) \tag{6.3}$$

If a line appears to be passing through a sector, it does not necessarily mean that this sector will be selected for further processing. A sector is marked as active when, at least, three lines from three different layers have crossed it. For this purpose, every sector is associated to a layer filter implemented as a 4-bit register. Each bit corresponds to one of the four layers. When computing the FHT for an SVD hit from layer n, the bit $n-1$ is set to 1. After processing

all coordinates, each cell will either be set to 1 if at least 3 of the layer filter bits are asserted, otherwise it remains 0.
Since the entire set of SVD hits are necessary in every state of the sequencer, they are stored

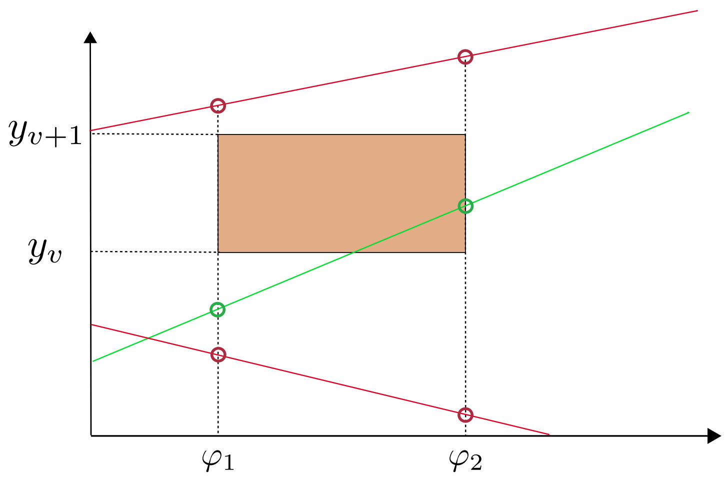


Figure 6.2: Illustration of the detection of a line passing though the cell v. Only the line in green is valid for the highlighted sector since the corresponding y value for φ_2 lies between y_{v+1} and y_v .

in dedicated block-RAM memories. Two are used to store the hit coordinates and one is necessary to store the corresponding layer. Storages are all written only once per event. For each incoming pair of coordinates, a common address pointer is instantiated and used to write the coordinates and layer into memory. On every state, a second unique reading address pointer is used to readout the event data. This process is repeated to build each column of the HS.
The time necessary to construct the HS can be expressed as:

$$t = (H + S_{HS} \times (H + 3)) \times \frac{1}{f_c}, \tag{6.4}$$

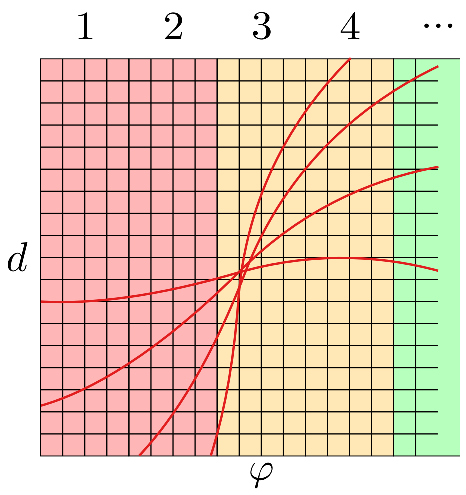
where S_{HS} is the number of HS columns. In the current setup $S_{HS} = 128$. The number of SVD hits to process is referred by H, 3 is for the number of sub-states and f_c is the clock frequency at which the firmware is running, here set at 78.125 MHz.
The 64 cells that form a HS column are computed in parallel, but only after every hit was processed, a 64-bit vector is transmitted from the SCM to the main sequencer.
As mentioned, in its basic form, the Fast Hough Transformation (FHT) sequencer completes its task within 128×3 states to cover all the columns of the HS. Nevertheless, it has been decided to profit again from the parallelism that FPGAs offer to compute more than a single column at a time and, therefore, reduce the computation duration. The grouping number G defines how many columns have to be built in parallel. The resulting number of states necessary to

complete the entire HS is inversely proportional to G. Increasing the effort to build the HS in parallel necessarily increases the resource usage. Table 6.1 shows the FPGA utilization of the FHT implementation for various parallelization factors. Grouping by 32 needs more resources than available and 16 do not leave enough for the rest of the firmware. In the current version the columns are computed by groups of 8. Contrary to the previous implementation, in every

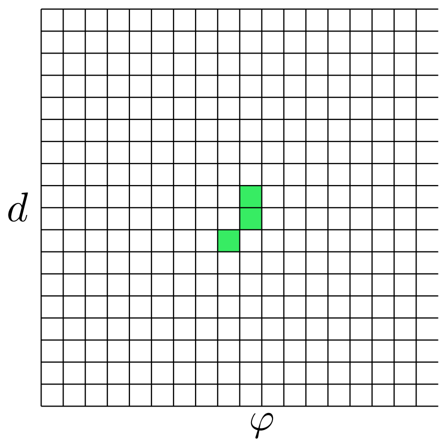
N	Register usage	CLB usages
1	0.27%	3.43%
4	0.47%	9.49%
8	0.90%	18.62%
16	1.75%	36.73%
32	3.56%	104.72%

Table 6.1: Resources usage for different parallelization factors of the HS in terms of register and CLB. A value of 8 was chosen for $\mathcal{P}$ and $\mathcal{N}$ tracking.

state, G sets of cos_1, cos_2, sin_1 and sin_2 are now returned.
By the same logic, $G\times$64-bit vectors representing the columns are generated at every stage. An example of the left part of the grouped HS construction is depicted in fig. 6.3a. Sectors, which have at least three lines passing through it, are marked as active resulting in the matrix shown in fig. 6.3b.



(a) Hough space grouped build. Columns are constructed in parallel by group of 8, highlighted in different color.



(b) Resulting Hough space matrix.

Figure 6.3: Illustration of the Hough space building where multiple columns are calculated in parallel. The result in a HS binary matrix where active cells had at least 3 lines passing through it.

6.4 Cluster Engine

Building the HS for an event has to be followed by further steps before a ROI can be calculated. The first is to extract the relevant information such as the track radius and angle. The final HS can be considered as a binary matrix where groups of active cells, also called clusters, can be found. The role of the clusterizer is to process those matrices, detect clusters, select them if they match size and shape requirements and finally generate the center of gravity for each.

6.4.1 Implementation

The HS clusterizer is based on the Depth First Search (DFS) algorithm [40]. Initially used for searching recursively tree or graph data structures, the algorithm starts from a root node and runs through the graph to find connected edges. It also appears to be a viable solution for reading the HS. The pseudo-code of the DFS algorithm is detailed in fig. 6.4. Recursion is very common for software design, but it is rare on FPGAs without the use of embedded processor [41]. For this reason the algorithm has been implemented in its iterative form.

The coordinates of the cell are referenced with the pair i, j. A stack S implemented as a

```
1: mark cell as visited
2: for all neighbor n_cell of cell do
3:   if n_cell not visited then
4:     DFS(HS,n_cell)
5:   end if
6: end for
```

Figure 6.4: Pseudo code of the recursive depth first search algorithm.

block RAM is introduced to keep track of the visited cells, it is indexed with the variable k. Every time a new active cell is visited, its coordinate is added to the stack, k increases and the cell is set to inactive or 0. This prevents it from being visited again. If the current cell has no active neighbor the cluster engine returns to the last visited cell to investigate if it had neighbors. This is done by decreasing the value of k. Once the variable k is back to its initial value, and no active neighbors are present, a cluster of cells has been found.

A cluster starts from a seed cell that is first found by checking all the cells of a column at once. Columns are passed until a non-empty one was found. The first active cell of the selected column becomes the seed in the DFS algorithm. Active neighbors of the seed are checked in the next steps. In most of the cases there are 8 possible neighbors to a cell. However, if the seed is located in a corner, there are only 3 and, when on the edge only 5. Within the basic form of a DFS all the directions are checked but, for DATCON purposes, the possible search directions were constrained for $\mathcal{P}$ and $\mathcal{N}$ side. For the $\mathcal{P}$ side, the clusterizer only selects rising cluster (x and y position increase), whereas for $\mathcal{N}$ only decreasing (x position increase and y

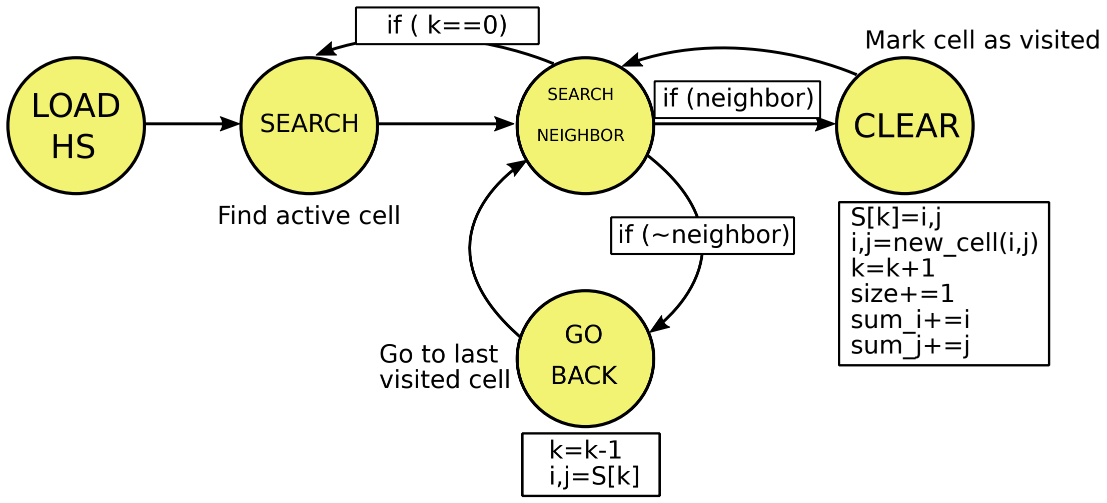


Figure 6.5: Simplified diagram of the clusterizer state machine. After loading the HS matrix, a seed cell is found. From there active neighbors are searched. If found, it is visited and marked as such. Once no neighbor can be found, the clusterizer goes back to the previous visited cell until it comes back to its initial position.

position decrease) shapes are searched. Limiting the possible paths is necessary to efficiently differentiate cluster and track candidates as mentioned in section 6.4.1. An example of a HS generated by two tracks is depicted in Figure 6.6. Only because of the constraints applied, the two clusters can be found. The module starts with the seed cell 1 and proceeds without invading the second cluster. The latter, starting on cell 2 can then be discovered, the cells of the first cluster that was visited are all inactive, thus, the cluster can not be seen anymore.
In addition to only finding connected cells inside the HS, and to make the clusterizer task complete, the sum of cell vertical and horizontal coordinates as well as the size of the current cluster is computed while the DFS searches the HS. Every time a new cell is visited, the cluster size is increased by one, and the i and j coordinates are respectively added to the horizontal and vertical sums. The center of gravity of a cluster is determined by dividing the horizontal and vertical sum by the cluster size metionned in in section 6.4.2 in eq. (6.5).

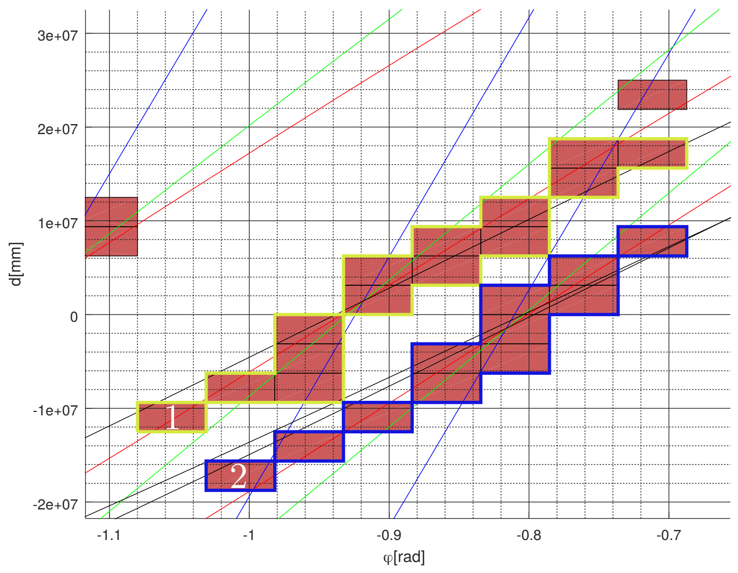


Figure 6.6: Example of two touching clusters created by two tracks on the $\mathcal{P}$ side. The clusterizer can distinguish them albeit they touch on several points.

6.4.2 Center-of-Gravity Calculation

The horizontal and vertical sums of the cell coordinates as well as the size of the cluster are constantly updated while it is processed. After it has been successfully detected, its center of gravity is computed as follow:

$$Cog_{x,y} = \frac{\sum cell_{x,y:i}}{cluster\_size} \tag{6.5}$$

The dividend and divisor are both encoded with 16 bits and the Euclidean division of the two unsigned integers has to be implemented and performed.
Division is a complex computation on hardware that normally requires recursive algorithms while, subtraction or multiplication only need bit manipulations. The existing division algorithms can be separated into two classes [42][43]. The first, the fast class is based on multiplication iteration and converges quadratically, whereas slow classes, with a time complexity proportional to the divisor length, produce a single bit of the quotient per iteration. Examples of this class are the restoring, the comparison or the non-restoring method. The restoring process was chosen for DATCON and is depicted in fig. 6.7. R is the remainder, even though it is not used for HS cluster calculation, N is the numerator, D is the denominator and n the number of bits of the quotient Q set to 16. The quotient's digits are selected from the set [0,1] depending on the sign of the partial remainder. It takes exactly n clock cycles or iterations to compute the division. More complex, non-restoring algorithms rely on the digit set [-1,1]

```
1:  $R = N$
2:  $D = \text{D} << \text{n}$
3:  while i < n do
4:      $R = 2 \times R - D$
5:      if $R \geq 0$ then
6:          $Q[i] = 1$
7:      else
8:          $Q[i] = 0$
9:          $R = R + D$
10:     end if
11: end while
```

Figure 6.7: Restoring Division pseudocode.

and can reduce the computation time by a factor 2. Faster algorithms are unnecessary as the computation duration remains below the average time in which two clusters are found.
In fig. 6.8, the distribution of the number of clock cycles necessary to detect a cluster is shown. The time between two found clusters was measured based on generated events with 10 tracks and included background. The details of the event generation will be given in Chapter 7.
It appears that the number of clock cycles needed to compute a cluster is always greater than the division latency and does not cause any additional delays. The restoring division is a reasonable solution for computing the center of gravity of the clusters.

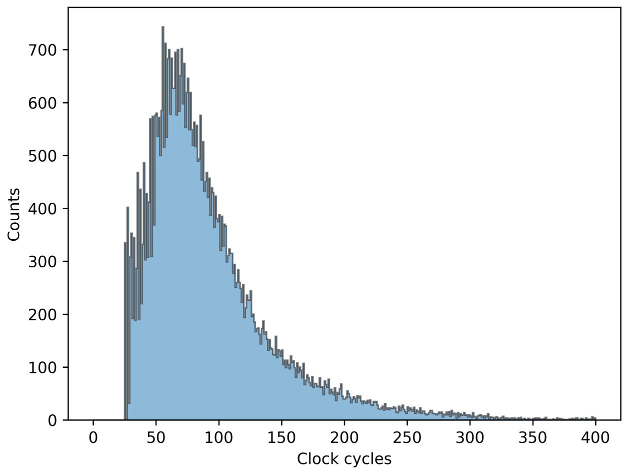


Figure 6.8: Number of clock cycles to detect a single cluster on $\mathcal{P}$ side.

6.4.3 Test Results

In order to measure the performance of the clusterizer it was ported onto the Tracking FPGA board.
A Python tool is used to randomly generate matrices, or HS, that are sent to the FPGA and processed by the clusterizer module. The clusters are then sent back to the PC along with their sizes and coordinate sum. Upon reception, they are compared with the ones extracted locally. Over 10^4 HS have been generated and injected into the FPGA. Every clusters was correctly found and only the rounding of the cluster position causes shift. The difference of positions between the clusters computed by the FPGA and by the Python script are shown in fig. 6.9. The maximum absolute distance corresponds to the center of gravity being exactly one cell away. Possible improvement would be to increase the resolution of the HS or to accept non-integer values for the center of gravity. For the rest of the chapter, however, integer values for a HS cluster position are used.

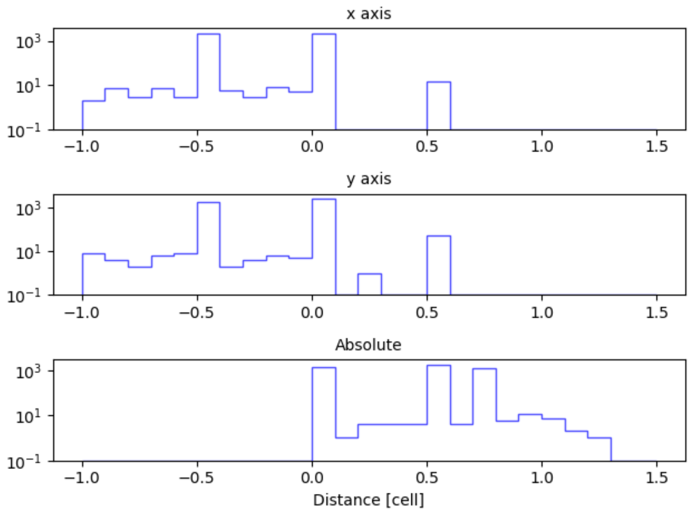


Figure 6.9: Residuals between the center of gravity of clusters computed locally and clusters computed on FPGA.

6.5 Hit Extrapolation and ROI Creation

Depending on the side where the extrapolation is performed, the computation differs. On the $\mathcal{P}$ side, the track is considered as a circle and extrapolated as such, whereas on the $\mathcal{N}$ side it is reduced to a straight line. The details about the calculations are shown in section 3.3

6.5.1 $\mathcal{P}$ Side Extrapolation

The extrapolation for all the PXD sensors is done using a single Verilog module instantiated twenty times, one for every PXD ladder. The x coordinate of the extrapolated hit is simply the radius of the sensor while the y component has to be calculated as follow:

$$y_s = -r_t cos(\varphi) \pm \sqrt{r_t^2 - (r_s - r_t sin(\varphi))^2} \tag{6.6}$$

For computing the square root, Xilinx offers the LogiCORE CORDIC IP which implements a generalized coordinate rotation digital computer algorithm. Described the first time by J.E Volder in 1959[44] and later generalized by J.S Walther [45], it allows for the calculation of elementary functions such as the square root. The core has been implemented to compute the square root of an 48-bit integer. To finalize the extrapolation of a track candidate the cosine and sine of $\varphi = \varphi_t - \varphi_s$ is needed. φ_t and φ_s are the angle of the track obtained from the Hough Space and the angle of the PXD ladder, respectively. Thus, two LUTs of size 128, the same as the HS horizontal length, are instantiated for every ladder.
Since a track candidate will, in most of the cases, generate a ROI in at least two different ladders, one on each layer, the extrapolated hits are serialized and only one instance of the ROI creation module is needed. Thereby, any conflict can be avoided. On the $r - \varphi$-plane the extrapolated hit does not give information on which PXD half module the hit is located, but only on which ladder. For this reason, every hit found is attached with a ladder ID within the range [1:20].

6.5.2 $\mathcal{N}$ Side Extrapolation

The extrapolation on the $\mathcal{N}$ side is much more straight forward than for $\mathcal{P}$. Indeed, the projection of the track on the $z - y$ plane is reduced to a straight line. A single LUT is needed to return the tangent value of $\frac{\pi}{2} - \theta$ from which the extrapolated z coordinate is computed as follows:

$$z = y \cdot \tan{(\frac{\pi}{2} - \theta)} \tag{6.7}$$

In this previous equation, y can be set to the radius of either the PXD first or second layer. From the coordinates of the extrapolated track, the most probable pixel is returned and a ROI can be generated around it, the same way it is done for the $\mathcal{P}$ side.

6.5.3 ROI Creation

From the obtained extrapolated track, the principle of ROI creation is the same for both $\mathcal{P}$ and $\mathcal{N}$ side.

The pixel local coordinates of the first ladder of PXD sensors are hardcoded and are used as comparison reference to locate an extrapolated hit. From this comparison a unique pixel ID is returned. Depending on the side it is either a column or a row ID. Around the found pixel, a ROI of predefined size is built. Originally, a fixed size of 80×80 pixels was adopted. The size can be changed on the fly from the slow control introduced in Chapter 8. One improvement would be to adapt the size of the ROI depending on the momentum of the track. In accordance with the extrapolation step, a ROI on the $\mathcal{P}$ side is necessarily attached with a ladder ID.
In case of a hit being too close to the edge of a PXD sensor on the $\mathcal{N}$ side, a ROI is directly created on the corresponding half module. Figure 6.10 shows a case where two tracks were found and two ROIs were created on the $\mathcal{N}$ side. On the forward side the ROI is too close to the edge of the PXD module and its size would not reach the requirements. A ROI is consequently created on the backward module even though no track was extrapolated. Possible inaccuracy of track finding or extrapolation could result in a misplaced extrapolated hit. A region of interest around the true hit would never be created. An extrapolated hit having its z coordinate exactly in between the PXD sensors causes two ROIs, one on both half ladders to be created.

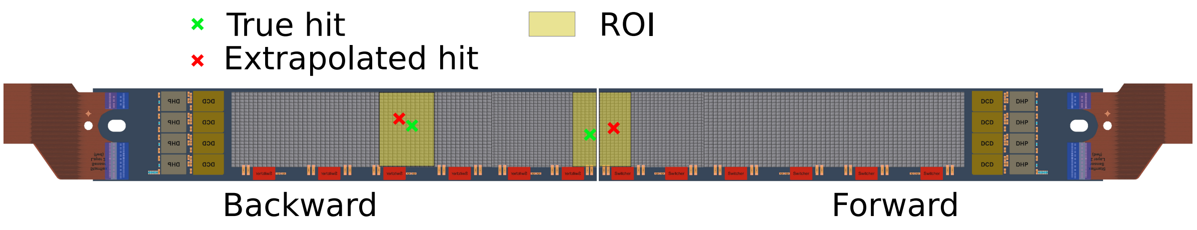


Figure 6.10: Example of two $\mathcal{N}$ side ROIs created around the extrapolated hit (red). The hit found near the center of the ladder causes the ROI to be extended to the backward PXD sensor and covers the true hit produced by a track.

6.6 Implementation on LUT

During the calculation of the coordinates for extrapolated track candidates, roundings are introduced. Multiple use of rounded values has a negative effect on overall performance and should be avoided. In order to decrease the negative effect of rounding, the use of LUTs was extended. The HS's dimensions are predefined, and they contain 8192 cells each. As described in the previous sections, finding a track candidate starts from a HS cluster position which is, by design, a set of integer coordinates that corresponds to a unique cell. From this cell, the extrapolation location onto the PXD layers is calculated, followed by the ROI computation. Since all the steps are directly resulting from one of the 8192 possible cells, a practical simplification is to access the final ROIs or at least their centers directly from a cell coordinate and

therefore avoiding multiple computations on the FPGA
The low number of cells, compared to the FPGA resources, makes the use of LUTs a viable solution. The LUT is implemented as a 8192-deep ROM whose address corresponds to the HS cluster cell number. For the $\mathcal{P}$ side, the output consists of 24 ROIs, one for each of the 8 ladders of the first layer and the 12 of the second. On the $\mathcal{N}$ side, a maximum of four ROIs have to be stored for every HS cell. Based on a ROI center position coded on 10 bits, the overall block-RAM occupancy is below 2.2 % for $\mathcal{N}$ side and just above 13% for the $\mathcal{P}$ side
To build the described solution, a Python script is used for generating the ROM initialization files as introduced in section 6.3 by running the entire extrapolation and ROI creation chain. After setting up the ranges and dimensions of the HS, the script retrieves the track angle and radius of each cell from which the extrapolated hit coordinate is computed. By dividing the obtained coordinate by the corresponding pixel pitch, a pixel ID is calculated. From this pixel, a surrounding ROI is created based on the predefined size. Depending on the SVD side, ROI is defined by the pair (col_1, col_2) for $\mathcal{P}$ or (row_1, row_2) for N.

6.7 ROI Merging and Transmitter to ONSEN

As described in the previous sections two independent track reconstructions and ROI creations are performed. Whether the ROIs are obtained from computing all the necessary steps on FPGA or by the LUT approach developed in the previous section, the ROIs have to be combined as the $\mathcal{P}$ ones only provide information about the columns and $\mathcal{N}$ ROIs only about the rows.
After an event has been processed, and at least one track was found and a ROI produced, the ROIs are stored in a ring buffer waiting for readout and merging.
Merging the ROIs is done by the merger module which is instantiated 40 times, one for every PXD half module. The task of this module is to create every $\mathcal{P}$ -$\mathcal{N}$ ROI pair with the previously computed ROIs. This naturally creates fake ROIs as shown in fig. 6.11. The pair of ROIs is formatted accordingly to the DATCON-ONSEN protocol where a ROI is encapsulated into two 32-bit words as depicted in fig. 6.12. A ROI is in this case described based on its lower and upper column and row numbers. The lower edge of a ROI is referred as $(col1, row1)$ while the upper edge as $(col2, row2)$. The column and row IDs are respectively encoded with 8 and 10 bits. A ROI is always attached to a PXD sensor ID, and every PXD sensor is controlled by a DHE as seen in section 2.3.3, thus a DHE ID is generated and propagated with the current ROI. The DHE ID is a unique identifier depending on the ladder number, the layer and the direction of a sensor. It is computed with:

$$DHE_{id} = (layer - 1) \times 32 + ladder \times 2 + (direction - 1) \tag{6.8}$$

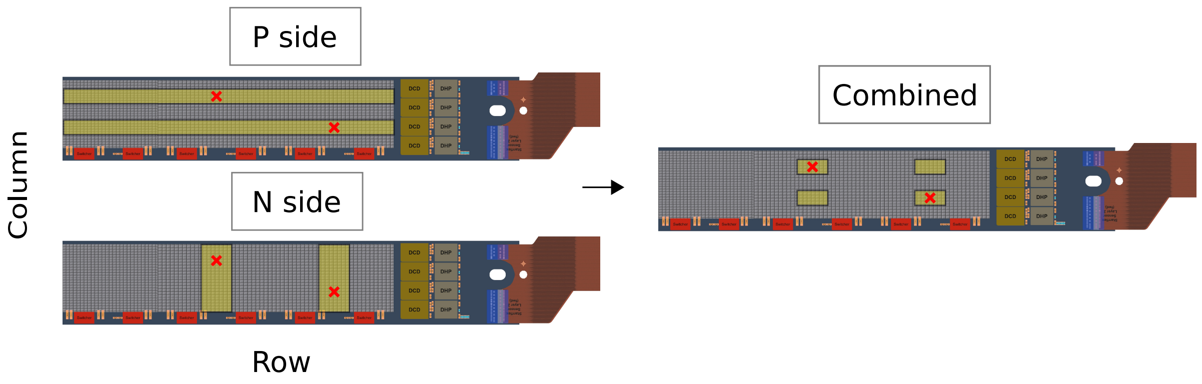


Figure 6.11: ROI merging principle with two track candidates. Two ROIs are found on each side creating four final ROIs. Half of them do not contain any relevant information.

The direction can be backward or forward and take the value 2 and 1, respectively. The layer can have the value 1 or 2 whereas the ladder belongs to the range [1:8],[1:12] for the inner and outer layer respectively. Backward direction correspond to a value of 2 while forward is 1. Every ROI merger module is instantiated with its specific combination of layer, ladder and

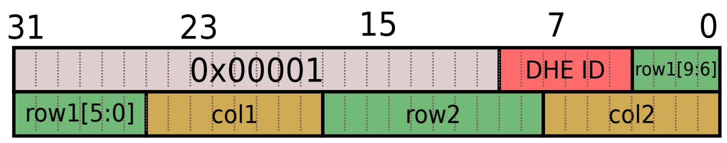


Figure 6.12: ROI format for ONSEN.

direction producing a unique DHE ID. For every ROI, the merger produces two 32-bit data words with the format depicted in fig. 6.12.
The 40 ROI mergers are connected to a final ROI sender that transmits the ROIs via a 3.125 Gbps single optical link based on the Xilinx Aurora protocol. The transmitter to ONSEN

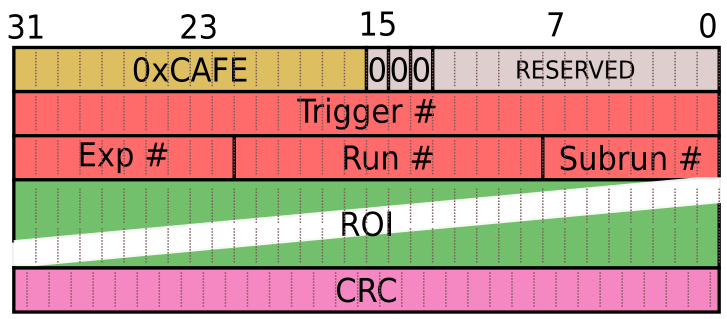


Figure 6.13: Frame format for ONSEN

requests the ROIs from every merger. This readout scheme and transmitting process to the ONSEN system is ordered in increasing DHE IDs. An event from DATCON always starts by

a known header followed by the trigger number which was first received by DATCON from the SVD inputs that received it from the FTSW. See section 2.3.2.
After the trigger information, run details are encoded. Only the 14-bit run number is of importance for ONSEN, but DATCON sends also the experiment and subrun number that were received from the FTBs. After those first words, the correctly formatted ROIs are sent out. The final word of an event frame is the CRC. The built packet, shown in fig. 6.13, is stored in a FIFO which is read as soon as the Aurora destination, ONSEN, is ready, thus preventing any data transmission loss.
The ONSEN can receive up to 128 ROIs in total and not more than 32 for a single PXD sensors. Chapter 7 discusses, in part, the number of ROIs that DATCON produces and how it can be limited.

6.8 DATCON Hardware Simulation

The simulation of the DATCON firmware was a large part for this work. Without extensively detailing the process, this section provides an short overview about simulation. Development of HDL design always necessitates the use of simulation to verify if it successfully fulfills its intended task, it was therefore built in parallel to the firmware. To run the simulation a testbench file is needed, in the current case Verilog is chosen as language. The testbench is simply a wrapper in which is instantiated at least one module to be simulated. The module's input ports, which normally would be physically connected onto the PCB or to another module, are steered by user defined stimulus. For DATCON, the only inputs which are asserted by the user are the clocks and the reset signals. On the AMCs the clocks are provided by external oscillators. In the testbench a waveform is generated based on the oscillator period and will serve as clock.
An example of stimulus given to the clock_i and the reset_i signals is shown in listing 6.1. The `‘timescale` compiler directive defines the time unit and precision for the current testbench. A delay of one time unit is represented by `#1`. In this example the base time unit is ns and simulation is done with a ps resolution.
In the `initial` statement, which is executed only once, the clock and reset are set to 0. After 200 ns written with the instruction `#200, reset_i` goes high for 100 ns before going back to its initial value. The clock `clk_i` is periodically and indefinitely reversed in the `always` block. The other ports of the DATCON boards are high speed link and can not be simulated by the same mean as incoming data which are the result of the Aurora protocol. The FTB emulation is therefore needed to complete the simulation. It includes a trigger, a data generator, a DATCON-link framer module, introduced in section 5.1, and the Aurora core. The data generator first loads RAM with FTB frames which are stored in .mif format and outputs them to the framer on received trigger. The framer is directly connected to the Aurora core itself

linked to the Aurora core on the DATCON side. The entire DATCON, as shown in fig. 4.1 can be simulated.
Important modules such as SVD decoder, FHT or HS clusterizer write their outputs into .txt files which are later used for visualization using Octave [46] or converted into Python numpy array for thorough analysis.
To run the testbench previously described, the Questasim simulator developed by Mentor Graphics was used. [47].

```
‘timescale 1 ns / 1 ps
parameter clk_period = 5;
reg clk_i,reset_i;

always begin
# clk_period clk_i =! clk_i;
end

initial begin
clk_i = 1'b0;
reset_i = 1'b0;
#200 reset_i = 1'b1;
#100 reset_i = 0;
end
```

Listing 6.1: Clock and reset stimulus.

7 DATCON Performance

In this chapter the performance of DATCON, governed by the ROI Finding Efficiency (RFE) and the Data Reduction Factor (DRF), is analyzed in various situations. Firs, by using the Belle Analysis Software Framework 2 (BASF2)[48] which allows simulation of the entire Belle II detector and production of the sub-detectors output signals. Simulation allows to generate any conceivable event on-demand with large freedom of settings. In comparison, physics data is constrained by event probability, detection and reconstruction efficiency.
Multiple simulation cases are used, from a single track without background up to a dozen of tracks generated by $B\bar{B}$ events. All along the chapter, the effects on the RFE and DRF are investigated and methods are introduced to improve those values as the event density increases. As a conclusion, DATCON ROIs obtained during data taking at KEK are compared with the ones from HLT.

7.1 RFE and DFR

DATCON is intended to perform track finding with the signal produced by the SVD system and generate ROIs on the PXD. The quality of a ROI can be quantified by the RFE and DRF values. The challenge is to ingeniously tune the tracking parameters to obtain the best possible combination of RFE and DRF. Naturally, by providing very large ROIs, the probability that they contain useful pixels is increased, as a consequence, the DRF diminished. On the other hand, drastically reducing ROI area increases the data reduction, therefore risking a PXD hit to be discarded. This section provides a clear definition and method of calculation for the RFE and DRF.

7.1.1 RFE

For the computation of the RFE the nuance between a ROI containing a PXD hit and a PXD hit being inside a ROI must be made. Demonstrated in section 6.7, the $\mathcal{P}$ and $\mathcal{N}$ ROIs are merged creating fake ROIs which do not contain any track-related PXD hits.
When a particle track passes through the PXD sensors, clusters of fired pixels are created. Hence, RFE is the number of PXD clusters found inside a DATCON ROI over the total number of clusters generated. The efficiency can be computed for a complete event and distributed

over range of p_t if available.

Error Calculation

Based on the efficiency ϵ, the probability of having k tracks found by a ROI out of n is defined by the binomial distribution:

$$P(k/\epsilon, n) = \binom{n}{k} \epsilon^k (1-\epsilon)^{n-k}, \tag{7.1}$$

Using the Bayes's theorem, the equation eq. (7.4) can be reversed as follow:

$$P(\epsilon/k, n) = \frac{P(k/\epsilon, n)P(\epsilon, n)}{C}, \tag{7.2}$$

where C is a normalization constant as detailed in [49]. The probability to obtain the efficiency ϵ knowing that k tracks are found by a ROI out of n becomes:

$$P(\epsilon/k, n) = (n+1)\binom{n}{k} \epsilon^k (1-\epsilon)^{n-k} \tag{7.3}$$

Based on the variance :

$$V(\epsilon) = \frac{(k+1)(k+2)}{(n+2)(n+3)} - \frac{(k+1)^2}{(n+2)^2}, \tag{7.4}$$

a 1-σ confidence interval around the mean efficiency is calculated. The interval is defined by $\overline{\epsilon} \pm \sigma$, where $\overline{\epsilon}$ is the mean efficiency and $\sigma = \sqrt{V(\epsilon)}$.
This method of error calculation is used for the rest of the chapter by replacing ϵ by the RFE.

7.1.2 DRF

Introduced in section 2.3.4 the PXD requires a data reduction to which DATCON is contributing. The DRF is calculated for an entire event and is expressed following the equation:

$$DRF = \frac{n_{PXD}}{n_{ROI}} \tag{7.5}$$

Where n_{PXD} is the total number of active pixels and n_{ROI} is the total number of active pixels inside a ROI. The DRF is independent of the RFE.

7.2 BASF2 Framework

To establish the quality of ROIs produced by DATCON it is necessary to have a viable reference for comparison. This is achieved by using BASF2 [48] which, like other High Energy Physics (HEP) software framework is intended to handle a large amount of data while being flexible. Just like its predecessor, Belle AnalysiS Framework (BASF) [50] which was successfully used over a decade with the Belle experiment, BASF2 can be used online and offline. For instance, the HLT relies on BASF2 for performing online analysis with real raw data while, offline applications use data stored after HLT decision. Additionally, it is possible to use BASF2 to generate MC simulated events that contain particles emitted from collisions or individual tracks from a predefined source, a feature which is referred as *particle gun*. For each event, the passage of the particles through the different layers of Belle II detector and their interaction with matter is simulated based on the Geant4 standard toolkit [51].
BASF2 is sub-divided into modules written in C++ which have their own specific task. For example, the event generator module is only responsible for particle collision production. In a Python steering file, the modules are arranged in a path and are executed one by one. Multiple paths can be instantiated and linked either linearly or conditionally. All the modules interact with a common storage, the DataStore, into which they can write and from which they can read. An overview of the framework organization is depicted in fig. 7.1. Within the framework, all

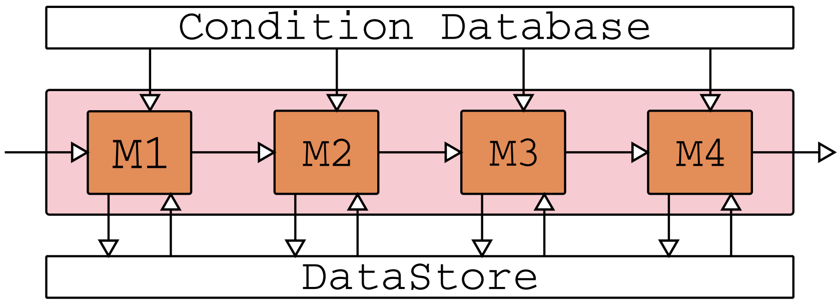


Figure 7.1: Diagram of the BASF2 steering process that includes four modules, M1 to M4, aligned in a path. Each module can read and write to the DataStore and is initialized by reading the Condition Database.

sub-detectors,described in section 2.2, have a digitizer module capable of producing a realistic output. The output can therefore be used for reconstruction or analysis. For the specific case of DATCON, the response of the SVD and more precisely the corresponding FTB frames are generated and later processed by a small-scale DATCON called DATCON-mini, detailed in the next section.

7.3 DATCON-Mini

During the course of this thesis, a compact version of DATCON was developed to simplify testing of the firmware without interfering with the system running at KEK. One of the main motivations for a small version of DATCON was the accessibility and availability of hardware. Almost the entirety of the DATCON AMCs are installed and running at KEK. The spares available are not sufficient to build a complete test-setup. Using the setup in Japan for testing is also not a viable solution as it is not only inconvenient but also means interrupting DATCON operation.

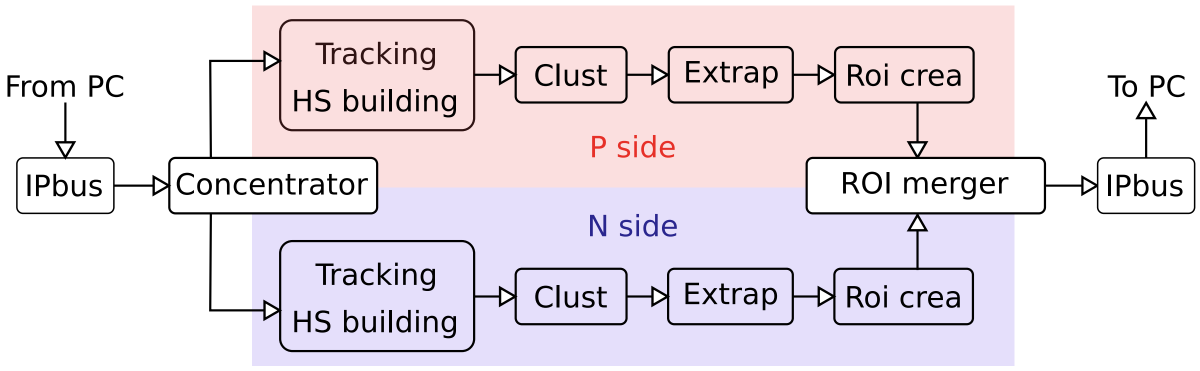


Figure 7.2: Logic diagram of the DATCON-Mini test setup structure. The essential modules previously introduced in Chapter 5 and 6 are included.

The idea behind DATCON-Mini is to include every part of the firmware described in Chapters 5 and 6 in a single FPGA. Every component related to event building or to interconnection (such as Aurora) is therefore removed. Verilog simulation of the DATCON-Mini is also built by mimicking the sequence previously described in section 6.8, with the exception of the IPbus part that was removed. The DATCON-Mini is built around a single Tracking board. A Python based control system is developed to transmit the FTB frames via Ethernet. On the DATCON-Mini side, an instanciation of the IPbus, described in more detail in section 8.3, is implemented and used to receive the SVD frames and to store them into FIFO before processing. A local counter is used to produce a trigger pulse that starts the readout of the IPbus FIFO which is connected to a compact version of the Concentrator. The frames of all the FTBs are received and pre-processed by a single Concentrator processing chain. Its output is spread out over $\mathcal{P}$ and $\mathcal{N}$ ROI finding chains, as explained in section 6.1. After the ROIs are merged, they are brought back to the IPbus core and sent to the PC. Figure 7.3 shows the DATCON-mini setup which uses simple μTCA carrier that does not require MCH or PM as introduced in Chapter 4. Just as in the KEK setup, the Tracking AMC requires the use of an extension board as a gateway between the PC and the FPGA.

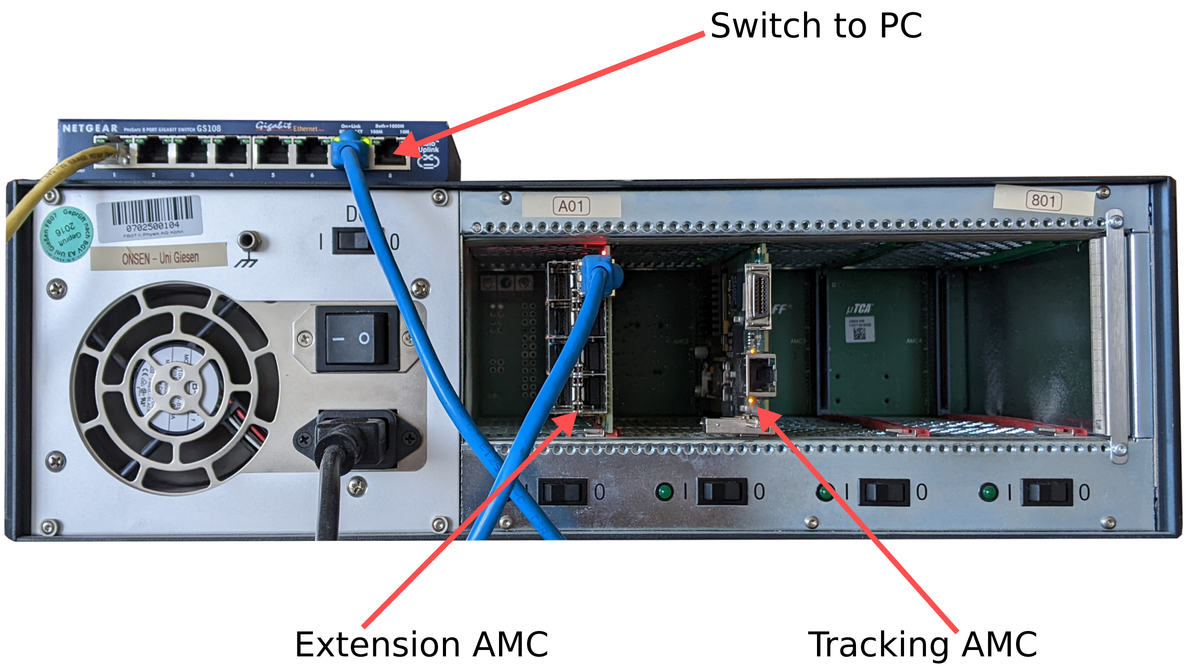


Figure 7.3: DATCON-Mini which includes only one Tracking AMC and one Extension AMC. From the PC, FTB frames are sent to DATCON-Mini which computes track finding and ROI. The ROIs are sent back via Ethernet.

7.4 Event Simulation

The goal of DATCON is to perform track finding and ROI calculation in the data recorded by Belle II , in particular for $B\bar{B}$ events and induced background. However, it is fundamental to focus on simpler cases first, for example single tracks without background.
The following sections presents the studies on DATCON tracking and ROI finding performance for various types of events generated by BASF2 and processed by DATCON-Mini . Simulated particles are not generated from e^-- e^+ collisions but from the particle gun module, which is added to the BASF2 path as described in section 7.2. The particle gun gives complete control over the event generation. It is possible to set the number of tracks, their range of momenta, their angles and also the type of particle.

7.4.1 Single Track Event

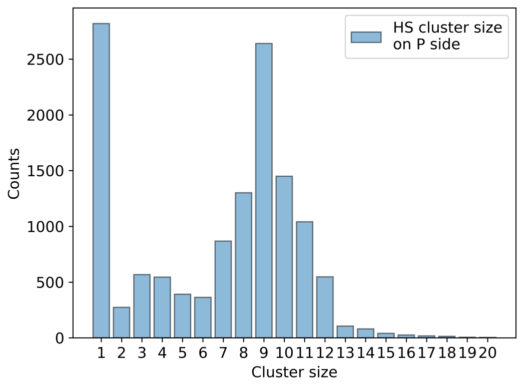
In this section, the particle gun is set to emit a single track for a set of events with user-defined parameters which are summarized in table 7.1. The features of the HS created by a single track is investigated to establish a base for the rest of the performance analysis.

Particle	e^+,e^-,μ^+,μ^-,π^+,π^-
Energy [GeV]	0 - 2
ϕ [deg]	0 - 360
θ [deg]	17 - 150
Source	Particle gun
Events	10^4
Magnetic field [T]	1.5
Origin [mm][mm]	(0,0)

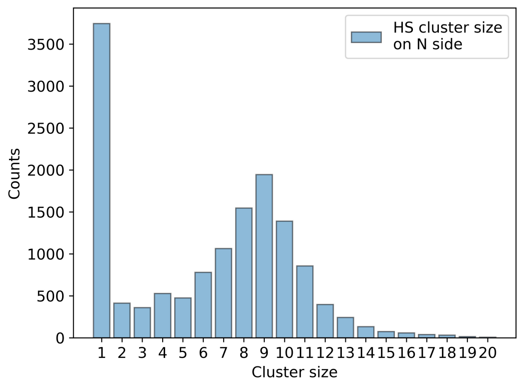
Table 7.1: Reference settings used for event simulation.

Hough Space Cluster Signature

After the HS building, introduced in section 6.3, the challenge for DATCON is to efficiently extract track candidates from the HS as described in section 6.4. It was also shown in section 6.4 that the clusters should have specific shapes to be valid. However, the number of cells per cluster needs to be evaluated. Originally, the clusterizer accepts every cluster whose size does not exceed the visited cell queue limit.
With the current particle gun settings, the typical HS cluster sizes for a single track event can be estimated. The number of cells per cluster obtained after processing the simulated event is depicted in fig. 7.4. Naturally, following the principle of separating $\mathcal{P}$ and $\mathcal{N}$ side, the results are shown for the two independent cases. It emerges that for both sides and particularly for the



(a) Cluster size on $\mathcal{P}$ side



(b) Cluster size on $\mathcal{N}$ side

Figure 7.4: Hough space cluster size distribution for single track events on $\mathcal{P}$ and $\mathcal{N}$ side.

$\mathcal{N}$ side, a large number of 1-cell clusters are found. As depicted in fig. 7.5, 1-cell clusters are often located in the direct vicinity of a larger one and are not centered around the intersection of lines created by the HT. Such clusters appear to simply be unconnected to the main one. The geometry of the HS defined by the cell dimension is directly causing this effect. Larger or

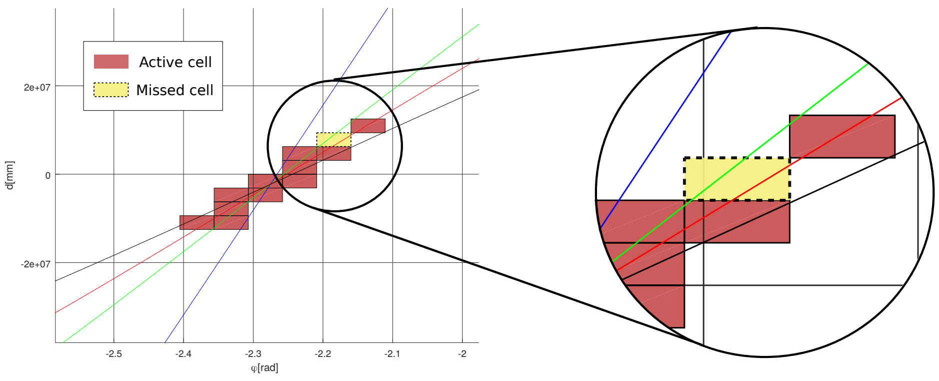


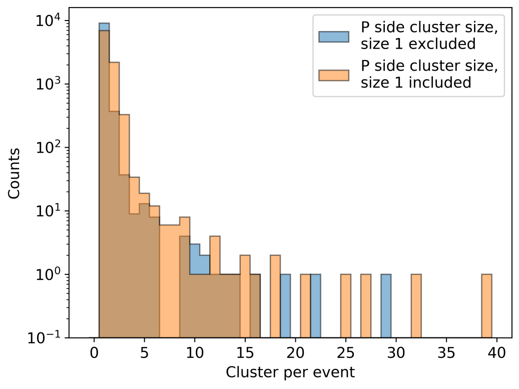
Figure 7.5: 1-cell cluster creation. The missed cell only contains HT from two SVD layers. A single active cell remains disconnected from the main cluster.

smaller cells could solve the issue for this particular case, but would not prevent it to happen again. A cell is active if at least three lines cross it. The detailed approach is formulated in eq. (6.3).
It is nevertheless required to comprehend if the 1-cell must be kept . In a perfect case, HT lines can intersect in a 1-cell cluster and the probability for it to happen can be estimated. The number of events that generated a HS containing a 1-cell cluster was computed and summarized in table 7.2. It shows that 1-cell cluster number is considerable but they are, in more than 99% of the cases, residual of the main HS clusters. The unique 1-cell cluster, meaning that there is only one cluster with only one cell in a HS, represent an extremely low fraction of all the 1-cell clusters. The clusterizer whose specifications were introduced in section 6.4, was then adapted to reject any 1-cell cluster as they can not improve DATCON performance and only result in a longer computation time and more importantly on a very large number of ROIs.

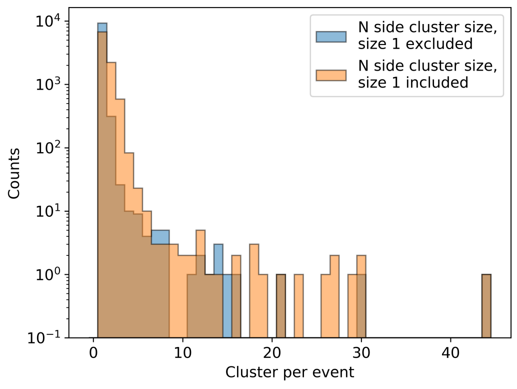
	$\mathcal{P}$ side	$\mathcal{N}$ side
Unique 1-cell clusters	3	25
1-cell clusters	2819	3744
Ratio of unique 1-cell clusters over 1-cell clusters	0.11%	0.67%
Total clusters	13156	14172

Table 7.2: Number of unique 1-cell clusters and total number of clusters with their corresponding ratios.

In the current simulation setup, the particle gun generates a unique track and it is expected to find only one track candidate. Nonetheless, as depicted in fig. 7.6, it appears that the DATCON HT creates two or more clusters for a non-negligible number of events. The residual single-cell clusters are the main contributor to multiple cluster finding and fake track building for single track events. Other effects are caused by multiple scattering caused by the particle interacting with detector material, which can generate secondary vertices. Low-momentum tracks that curl around the z-axis, produce additional SVD hits and more lines in the HS, consequently increasing the effective number of active cells and clusters. By discarding all single-cell clusters,



(a) $\mathcal{P}$ side HS cluster distribution



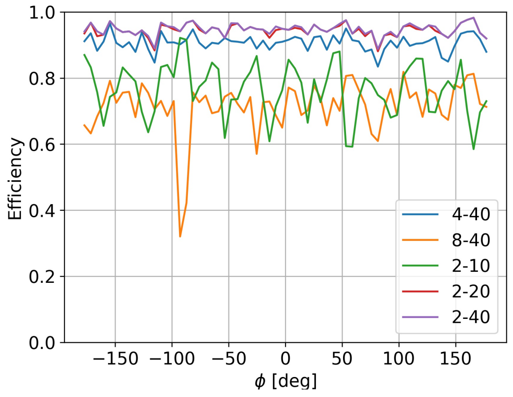
(b) $\mathcal{N}$ side HS cluster distribution

Figure 7.6: Number of Hough Space clusters per event. Discarding the single-cell considerably reduces the amount of clusters but also of track candidates and final ROIs.

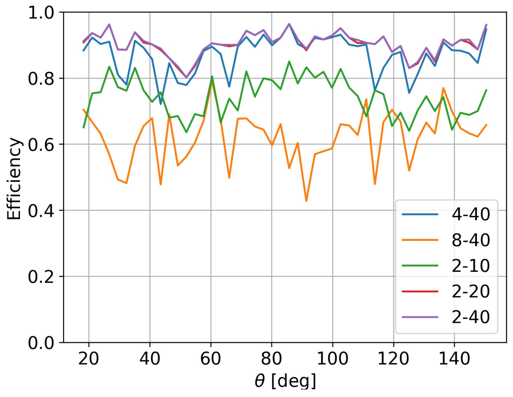
the final number of track candidates, obtained from HS clusters, is considerably reduced. The more potential tracks on both $\mathcal{P}$ and $\mathcal{N}$ side the higher is the number of final ROIs. A large number of ROIs risks to compromise the data reduction factor.

Effect on Cluster Size Selection

1-cell clusters are rejected for not improving DATCON performance while producing a large number of additional track candidates. The implementation of the HS clusterizer aims to find every possible cluster within a given cluster size range. A low threshold of 2 cells was admitted, but it is also interesting to investigate if other cluster size cuts are useful. Five sets of thresholds ranges are used with the widest going from 2 to 40 cells.
For each range, the remaining track candidates are used and their angle is compared with the initial track angle of the particle gun. If the absolute difference is below two HS cell size, 2,8125 ° for φ and 1,145,° for θ, the track is marked as found. The counting is done over the φ and θ range to compute the track finding efficiency. The efficiencies for the cluster size selections are shown in fig. 7.7. Linear interpolation between data points was applied for visibility. Narrowing the range of cluster size does not improve the efficiency. For the nominal range of [2:40] cells, the efficiency remains the highest. Reducing the high threshold only removes good track candidates.
In conclusion, for finding single tracks the most efficiently, the HS cluster size should be at



(a) $\mathcal{P}$ side track finding efficiency vs track angle



(b) $\mathcal{N}$ side track finding efficiency vs track angle

Figure 7.7: Track finding efficiency vs track angle for various HS cluster size ranges given on the legend.

least two cells, the upper limit should not be constrained. This settings is kept for the rest of the performance analysis.

7.4.2 Single Track Events with Background

The previous section 7.4.1 described the perfect case of a single track in a background-free environment, which is not representing realistic conditions. Indeed, during Belle II operation, a considerable amount of background is produced, inducing noise on both the SVD and PXD modules (see section 2.2.3).
BASF2 is again used to generate events with included background. There two kinds of backgrounds that can be added, the first one is obtained from simulation. Simulated beam backgrounds are estimated by extensive MC simulation. The second source is from randomly triggered events that do not contain any physical signal, consequently reflecting the effect of beam background and detector noise. Both background types can be included into BASF2 for MC simulation by using overlay files. The latter type of background, based on the early Phase3 luminosity is used for performance evaluation within this thesis.
The same set of events as the one of section 7.4.1 was reused with the described background sources included. The DATCON tracking settings remained unchanged, and as concluded already, the HS clusters with a size smaller than two are rejected.
In this section the performance of DATCON to find a single track surrounded by background hits is explored.

Data Reduction Factor

With the included background, it becomes possible for DATCON and DATCON-Mini to achieve their intended task, reduce the amount of PXD data by providing ROIs.
Throughout an event, pixels are activated either by the passage of a particle, the background or its inherent noise. Those pixels form PXD clusters whose number per event is shown in fig. 7.8. For a single track, it is expected to obtain one cluster on each PXD layer, yet the figure shows that on average, 429,95 clusters are produced, as a direct effect of the included background. The figure also shows the number of track candidates per event that are found by DATCON. To evaluate the DRF of DATCON the already mentioned PXD clusters are used. To compute the DRF, eq. (7.5) is used.
For the current case, events are dominated by a unique ROI on both PXD layer, more precisely 2.7 ROIs in total per event, resulting on an average DRF of 364.68.

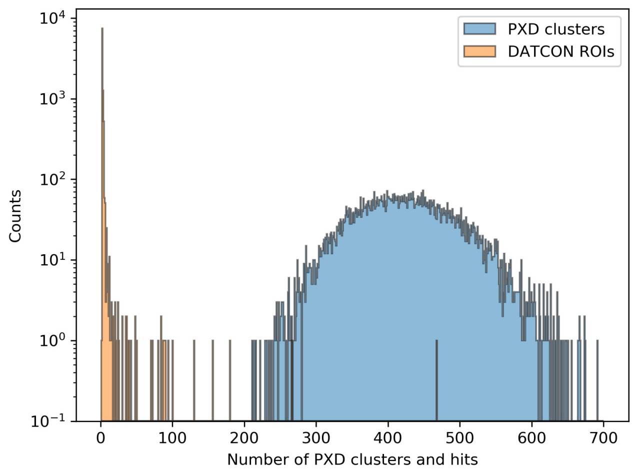


Figure 7.8: Number of PXD clusters per event, including background, compared to number of DATCON ROIs.

ROI Finding Efficiency

In addition to the DRF, the RFE is an important quantity to evaluate the performance. Indeed, the DRF does not indicate whether the data that was preserved is of any interest. To estimate the ROI's quality, the PXD_{true_hit} is introduced.
When a MC particle passes through the PXD layer, its creates a so called PXD_{true_hit} which is a single pixel. This data is only available by simulating events, in this case with the particle gun module. A PXD_{true_hit} is used as reference for investigating the ROIs quality.
Every PXD_{true_hit} is compared to every DATCON ROI if they both are on the same PXD

sensor. If a PXD_{true_hit} happens to be inside a ROI, the hit is counted as found. A ROI that does not contain a PXD hit is not included in the efficiency estimation. Every PXD_{true_hits} provides track information, such as momentum. For each p_t bin, the efficiency is computed by calculating the ratio of the number of PXD_{true_hit} inside a ROI over the total number of PXD_{true_hit}.
The efficiency was computed based on a nominal size of the DATCON ROIs of 80×80 pixels and is depicted in fig. 7.9 where the total efficiency is shown, as well as the independent $\mathcal{P}$ and $\mathcal{N}$ efficiencies. The $\mathcal{P}$ and $\mathcal{N}$ side RFE can be individually calculated by using either the

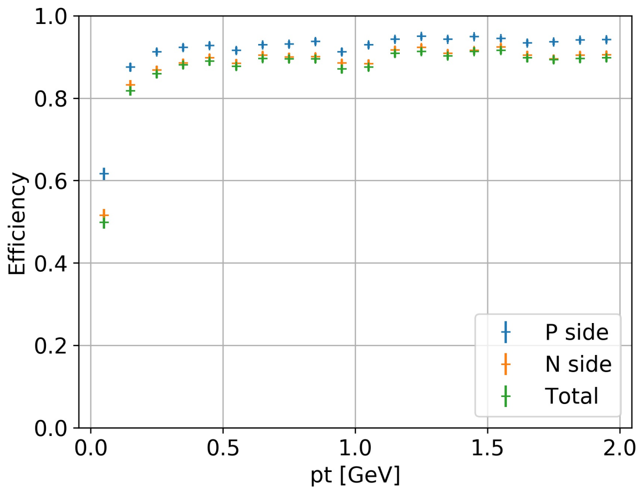


Figure 7.9: ROI finding efficiencies over p_T for single track event and early Phase3 background

column or the row information of a ROI only and the column or row value of a PXD_{true_hit}. The overall efficiency is above 88 % and is brought down by the $\mathcal{N}$ side RFE as explain bellow. There are different reasons explaining that a hit can not be found by any ROI and justifies the efficiency limitation:

- Tracks that do not originate from the IP. For reconstructing ϕ the conformal mapping, introduced in section 3.2, assumes a constant IP location at (0,0). Within the simulation condition, the IP position is known and fixed, however due to the strips coordinate system, used and described in section 3.4, tracks appear to be shifted from the origin, resulting in an inaccurately extrapolated hit. This issue is happening only when processing events on the $\mathcal{N}$ side. Every hits on a ladder are translated onto a single ladder per layer view (see fig. 3.9b) therefore causing displaced tracks, fig. 3.10.

- Causing a similar obstacle, tracks produced by multiple scattering have unknown origin. On the $\mathcal{P}$ side, the conformal mapping of a track not originating from the IP can not be used, as explained in section 3.2. On the $\mathcal{N}$ side, the hit found is shifted.

- Additionally for $\mathcal{N}$ side, the linear approximation of low momentum trajectories causes increasing deviation per layer eventually leading to missed hits. See section 3.3.1.
- Very low momentum tracks ($p_T < 35$ MeV), which cannot reach the third SVD layer, cannot be found nor reconstructed.

7.4.3 Multiple Tracks Events with Background

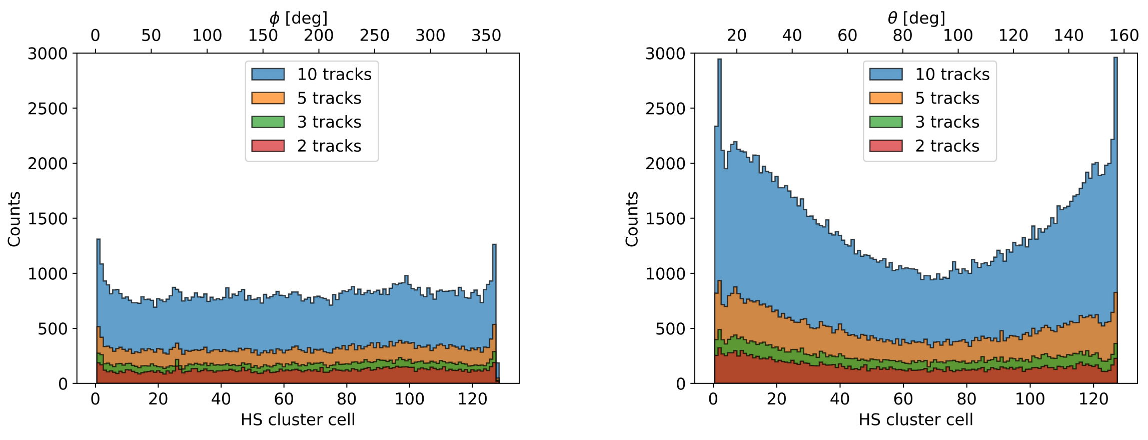
For single-track event, the high ROIs finding efficiency and the data reduction beyond requirements, confirm DATCON capabilities to efficiently find the track and to produce ROIs. However, DATCON is required to be able to work with $\Upsilon(4S)$ events, which produce around 11 tracks on average. For this reason and in a similar manner as for the single track tests, various events, with 2, 3, 5 and 10 tracks were generated with BASF2 to be later injected and processed by DATCON. Gradually augmenting the number of track is an appropriate approach to benchmark DATCON implementation.
The exact same DATCON settings as in the previous section were used at first but they quickly showed their limitations due to the much higher number of clusters detected in the HS and, therefore, the number of ROIs. If a high ROI finding efficiency is obtained, it results in many track candidates with a diminished DRF.
In this section the attention is first brought to the HS cluster distribution followed by different investigations made to improve the DRF while keeping a high RFE.

ϕ and θ Distribution

For the simulation, the tracks are uniformly generated over the angle range presented in table 7.1. The tracks found by DATCON are, however, not following the same distribution. The horizontal cluster positions, or x component, of the HS on the $\mathcal{P}$ and $\mathcal{N}$ side along with the corresponding ϕ and θ are represented in fig. 7.10. On each side, the cluster can have a position within the range [1:128]. Except on the edges of the HS, the distribution for ϕ tends to be uniform. The peaks are caused by a track angle close to 75° or 360° which creates two HS clusters at the edges. The effect observed is even more amplified for θ where extreme track angles lead to an increased multiple scattering. In these regions the background is also more important. Because more clusters are created at the edge of the HS, they can frequently have similar x and therefore θ value causing overlapping ROI. setting

HS Clusters and Track Candidates

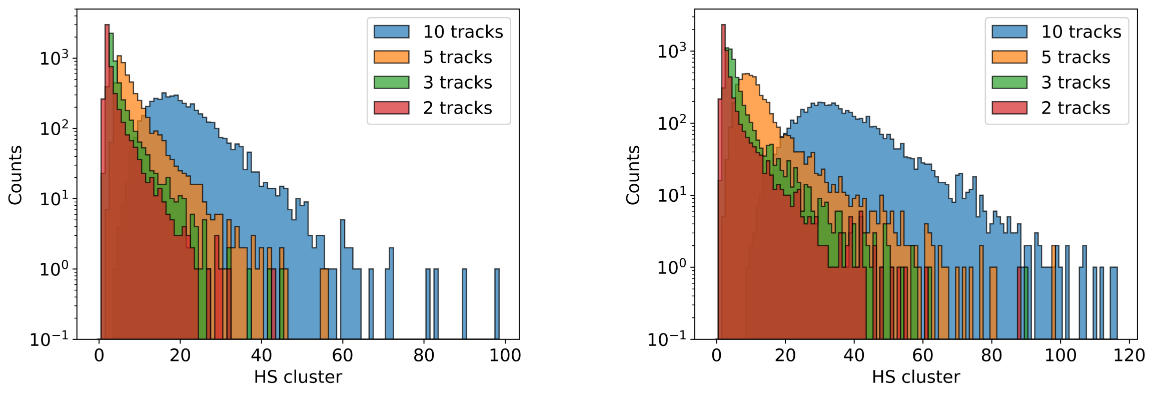
Figure 7.11 shows the number of clusters detected in the HS depending on the number of trakcs and on the side. Intuitively, n tracks should lead to n clusters, yet the number of clusters is



(a) ϕ distribution on the $\mathcal{P}$ side. (b) θ distribution on the $\mathcal{N}$ side.

Figure 7.10: Cluster position in the DATCON HSs and their corresponding angles for $\mathcal{P}$ and $\mathcal{N}$ side.

higher than the number of tracks and considerably increases with the quantity of track candidates. For instance, 2 tracks generate, on average, 3 and 4 clusters on $\mathcal{P}$ and $\mathcal{N}$ side while 10 tracks produce 21 and 37. Every cluster becomes a track candidate which potentially can lead to a ROI. A high number of candidates produces all the more ROIs during merging which is accompanied by a large number of fakes. The final mean number of ROIs is computed for



(a) Number of clusters found on the $\mathcal{P}$ side. (b) Number of clusters found on the $\mathcal{N}$ side.

Figure 7.11: Number of Hough space clusters found for the $\mathcal{P}$ and $\mathcal{N}$ side with 2, 3, 5 and 10 tracks events with early phase3 background.

the four test conditions on each event and is summarized in table 7.3. The number of ROIs containing a PXD_{true_hit}, not to confuse with the fake ROIs, are also shown in the table. The large number of fakes that comes with an increased number of tracks are not compatible with ONSEN specifications which support up to 128 ROIs. This limitation causes the over-

flowing ROIs to be discarded regardless of their possible quality.
Fake ROI generation is directly due to the way ROIs are merged from the $\mathcal{P}$ and $\mathcal{N}$ side as explained in section 6.7. For n track candidates that actually generate a ROI on both sides, DATCON produces $n(n-1)$ fake ROIs. This statement is only valid when more than one track candidate is extrapolated, otherwise, no fake can be created. The mean RFE oscillates

	2 tracks	3 tracks	5 tracks	10 tracks
Number of ROI	56	110	335	2078
Number of ROI containing a PXD_{true_hit}	4	8	19	89
Fraction of ROI containing a PXD_{true_hit}	7.1 %	7.3 %	5.7 %	4.3 %
Fraction of fake ROI	92.9 %	92.7 %	94.3 %	95.7 %

Table 7.3: Average number of DATCON ROIs per event for different number of tracks. The overall number of ROI can be compared with the ones that contain a PXD_{true_hit}.

between 88.74 % for 2 tracks event and 91.65 % for 10 tracks. Because an increased number of tracks increases the number of HS clusters and ROI, the probability to have a PXD_{true_hit} inside a ROI also increases. With the current techniques used to translate strips into coordinates and to perform tracking and extrapolation, the efficiencies can not be improved without increasing the ROI size.
As described in this section, an increased number of tracks also results in an increased number of fake ROIs, which directly affects the data reduction. The DRF distribution for 2, 3, 5 and 10 tracks was computed and is depicted in fig. 7.12. If for the 2-tracks test condition, DATCON

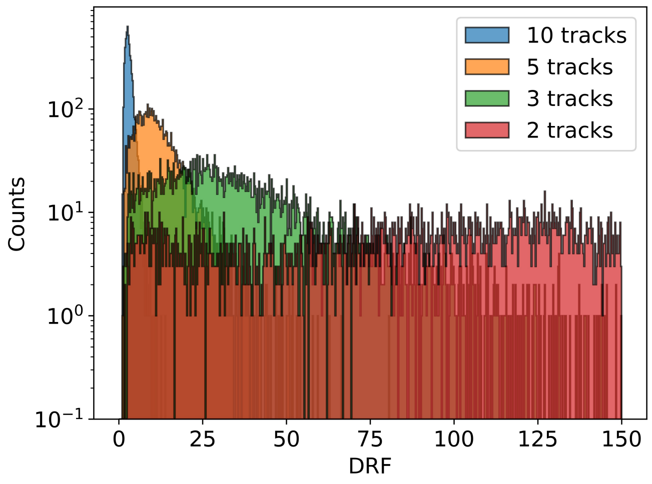


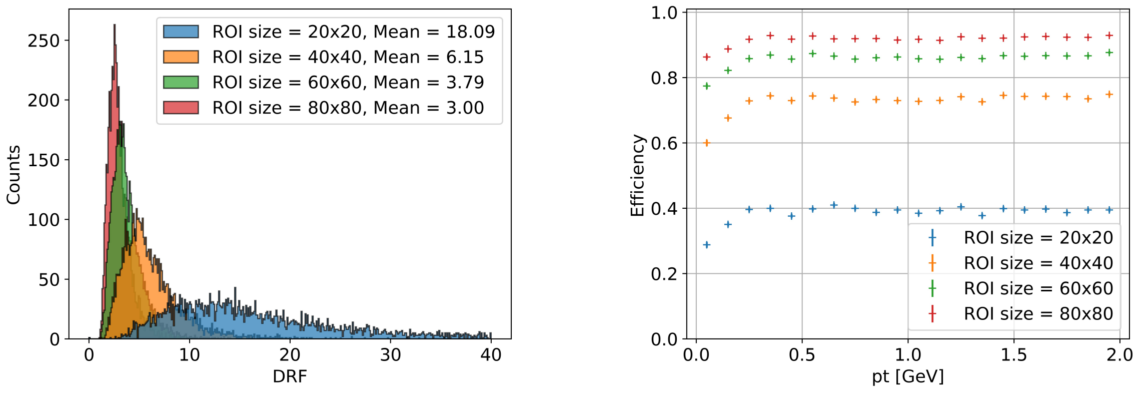
Figure 7.12: DRFs obtained for 2, 3, 5 and 10 track events with early Phase3 background.

is capable of removing a large part of background hits, providing a DRF of 93.36, the achieved

value for 10 tracks is quite far from the requirements. On average at 2.85.
Two ways of improvement were investigated. The first one is to reduce the size of the ROIs. Secondly the number of track candidates reduced directly by tuning the HS cluster selection. The following paragraphs describe the effects of the two approaches on the final DRF along with the global RFE with 10-tracks events which is roughly representative of the number of tracks expected in $\Upsilon(4S)\rightarrow B\bar{B}$ events.

ROI Size Reduction

So far the ROI size was fixed to 80×80 pixels but in order to find a good compromise between data reduction and ROI finding efficiency the ROI sizes are reduced to 60×60, 40×40 and 20×20. The input for DATCON generated by MC events remains unchanged, just as the rest of the DATCON settings. The DRFs is computed for each case as described in section 7.4.2 and is shown in fig. 7.13a. Figure 7.13b shows the different ROI efficiencies in bins of p_T



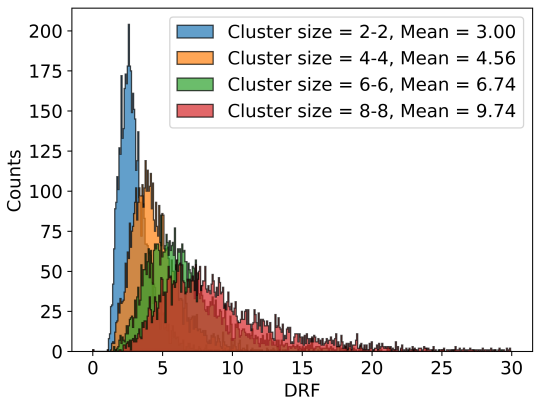
(a) DRF obtained for various ROI sizes. (b) Roi finding efficiencies for various ROI sizes.

Figure 7.13: DRF and corresponding RFE for different ROI sizes.

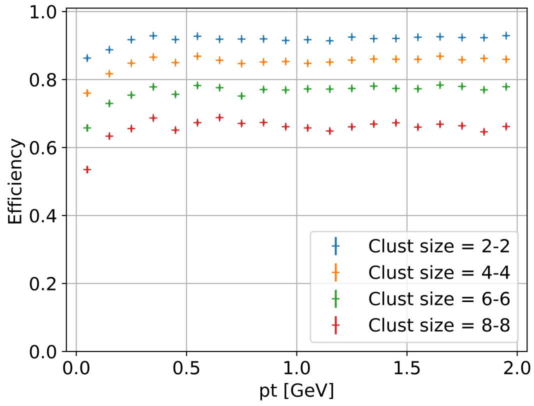
calculated for the four ROI sizes. Reducing the ROI size significantly improves the DRF but at the cost of a large efficiency drop. For instance, the only mean DRF greater than 10 is obtained for a ROI size of 20×20 pixels that result in an efficiency averaging at only 38.6 %. Reducing the size of the ROIs can only be done with caution and therefore a stricter selection on HS clusters must be made.

HS Cluster Size Threshold Increase

In addition to reducing the ROI size in order to improve the DRF, there is also the possibility to reduce the number of track candidates directly by modifying the required HS cluster size. Based on the single-track study, and developed in section 7.4.1, every cluster with a size of at least 2 cells is accepted. Such a loose selection may not adapted for increased number of tracks. Logically, indefinitely increasing the minimum threshold removes every possible track candidate and results in a zero RFE. For this reason the minimum HS cluster size of 2, 4, 6 and 8 is studied. The effect on DRF and RFE is observed and the results are presented in fig. 7.17a and fig. 7.17b. The ROI size was set back to 80×80 used in the previous section. The improvements on the DRF are less significant than by simply reducing the ROI size but



(a) Data Reduction Factors



(b) ROI finding efficiencies

Figure 7.14: DRF and corresponding RFE for different HS cluster sizes.

at comparable overall DRF it shows a better RFE. After exploring possibilities to improve DRF the conclusion is that with the current settings, reducing the HS cluster size or reducing the ROI size are not a solution as they result in a large RFE drop.

Merge the Overlaping ROIs

It often can happen that two HS cluster horizontal projections intersect, leading to two extrapolated hits on the PXD plane whose distance is below the defined ROI side length. It is already established that, to comply with ONSEN, the number of ROI must stay below 128 in total, and below 32 for a PXD sensor. During track parameter extraction from the HS

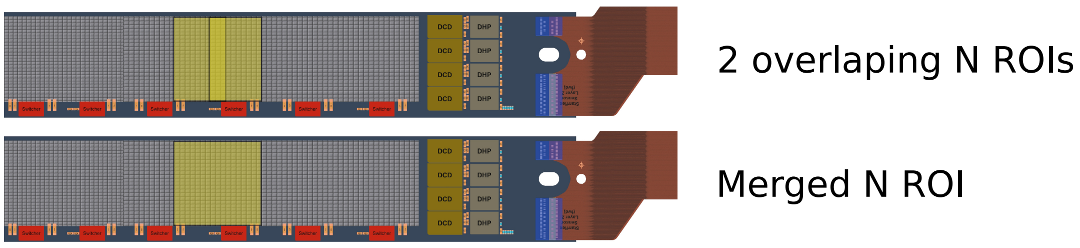


Figure 7.15: Illustration of the merging of two overlapping ROIs on the $\mathcal{N}$ side.

and especially for high-multiplicity events, multiple clusters can be close to each other, thus creating nearby extrapolated hits on the PXD plane. From those hits, a surrounding ROI is built which can often already contain the neighboring hit. The result is the production of numerous intersecting ROIs that can be merged into larger ones, thus reducing the total number of ROIs. This poblem is more noticeable on the $\mathcal{N}$ side, where the extrapolation is made only on two ladders on each layer, compared to the eight and twelve ladders on the $\mathcal{P}$ side.
During ROI computation the boundaries of the neighboring ROI are compared with the current ones to find a possible overlap. This is done only for ROIs belonging to the same PXD sensor. Since the creation is independently made on $\mathcal{P}$ and $\mathcal{N}$ side, the merging of the columns and the rows is also independent. Combining the overlapping ROIs has a considerable effect

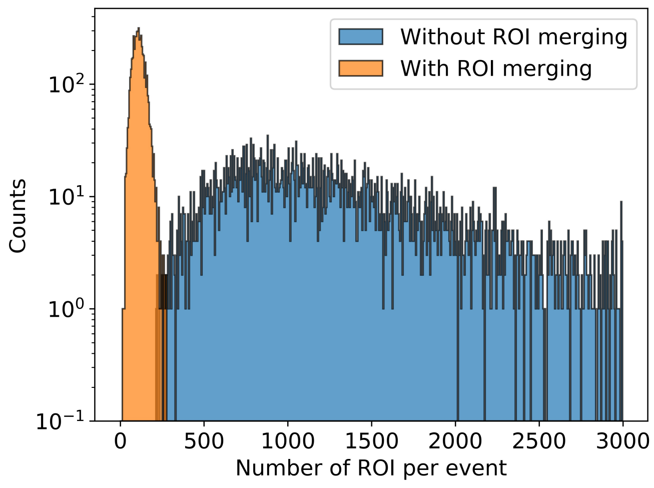
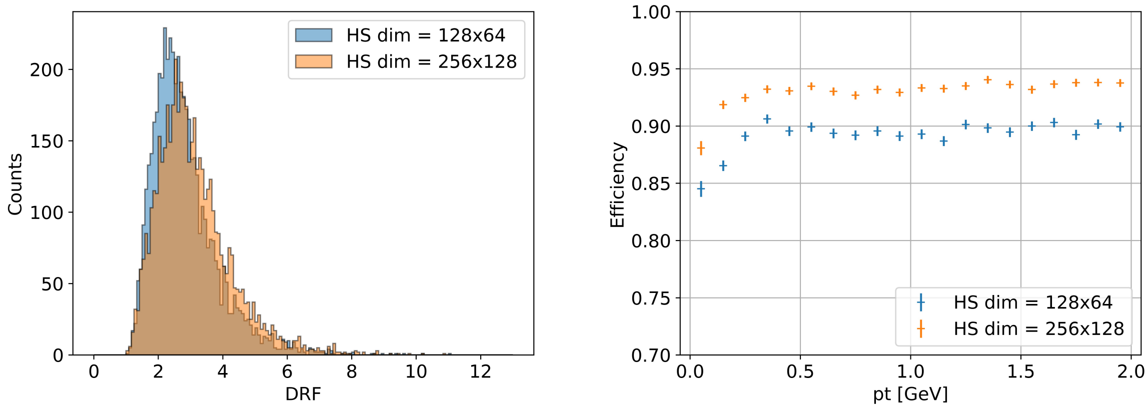


Figure 7.16: Number of produced ROI with and without ROI merging.

on minimizing the produced number of ROI but also the fakes. The fraction of fakes compared to the total number of ROIs was 95.7 % in table 7.3. By merging the overlapping ROI the fraction of fakes is brought down to 86.5 %. The most important effect of the merging is the number of ROIs which is brought down by 93.2 %, corresponding to an average number of ROI per event of 141. Without merging the was at 2078. It turns out that merging the ROIs when they intersect is an absolutely necessity in order for ONSEN to not randomly reject them when their number exceeds the buffers limitation. However, the mean number of produce ROIs still exceeds the requirement of 128 requested by ONSEN.

HS Dimensions Increase

The HS size used from section 7.4.1 set to 128×64 cells for both $\mathcal{P}$ and $\mathcal{N}$ side. The last step is to introduce larger HS by extending it to 256×128 cells. As detailed in section 6.3, the Verilog code of the state machine responsible to generate an HS is pre-built on PC by user defined parameters. The size was updated but the number of grouping steps was unchanged, see fig. 6.3a. The ranges used for ϕ and θ but also track radius are identical. The clusterizer was set up to accept only cluster with at least two cells. The ROI finding efficiency as a function of transverse momentum and the DRF per event were computed with the new HS size and the results are shown in fig. 7.17.The gain obtained for the DRF is on average about 11.07 % while the efficiency is increased by 3.78 %.



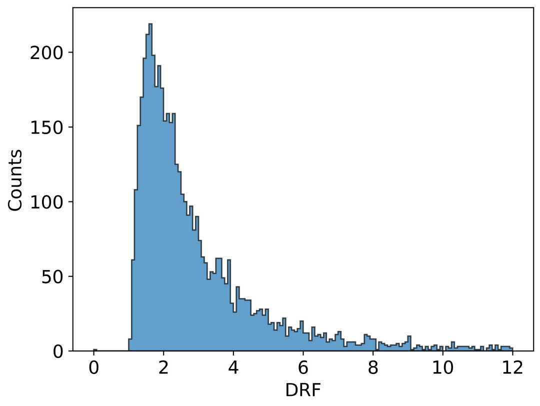
(a) Data Reduction Factors

(b) ROI finding efficiencies

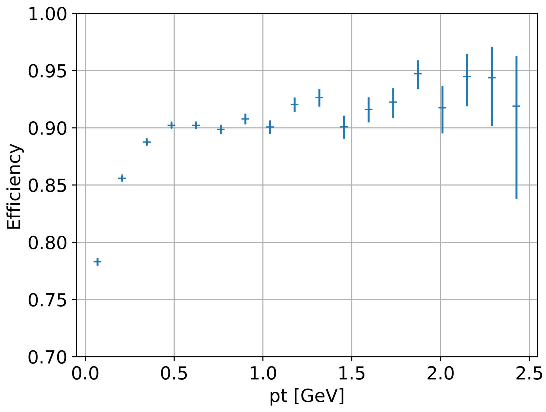
Figure 7.17: RFE over p_T and DRF per event for HS dimensions extended to 256×128 cells.

7.4.4 Υ(4S) Events with Background

Simulated events were so far generated with the particle gun and use to optimize the ROI and HS cluster sizes in order to maximize the efficiency and the DRF.
The particle gun used to generate tracks is removed and replace by the event generator module, e^+e^- collision is used to generate $\Upsilon(4S)$. It is therefore not possible anymore to influence settings such as track momentum, number or angle. MC events were generated for early Phase3 background conditions, as described in section 7.4.2.
$B\bar{B}$ events come as a logical conclusion for investigating DATCON performances.
Similar to the previous sections, the RFE and DRF are computed for each event and are shown in fig. 7.18. The momentum distribution is not uniform as the previous simulation and the lower number of tracks in the upper range induces error. The error shown is obtained by following the method introduced in section 7.1.1. Over all the events, a mean RFE above 85 % and mean DRF of 3.5 is achieved. So far, RFE and DRF are computed independently from



(a) DRF



(b) RFE

Figure 7.18: DRF and corresponding RFE for Υ4S events.

each other hence, a correlation plot is made.
For each event, combining all the PXD sensors, the DRF value is calculated . At the same time, the overall RFE of the event is determined. The details about their computation are given in section 7.1. For the RFE and DRF, the common denominator is therefore the event number. For each DRF bin, the corresponding events are selected and the mean RFE is calculated. From the sample of selected events, the standard error calculated. The resulting plot is depicted in fig. 7.19.

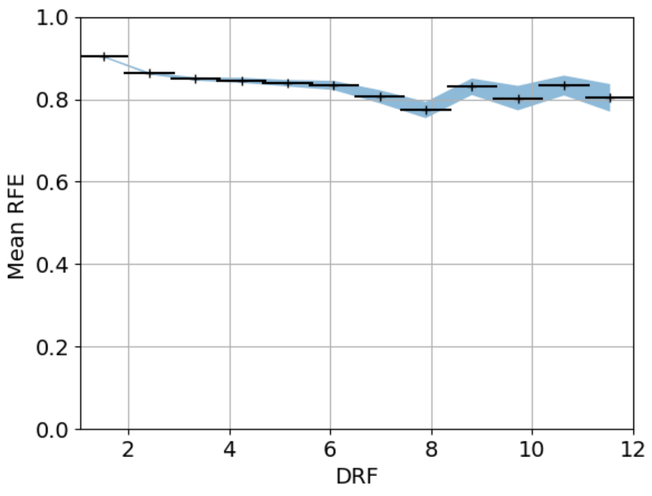


Figure 7.19: Mean RFE vs DRF.

7.5 Belle II Physics Runs

DATCON has been running and taking data during Phase3, hence to conclude this section, a summary of its functioning and the difficulties encountered during operation is given. To also include the various improvements previously described in section 7.4.3 that were not included to the running hardware,such as HS cluster size selection or ROI merging, collected SVD data from physics runs was reused and processed by DATCON-Mini . The results are compared to what was obtained from HLT. With run data, a different definition of the RFE must be used since naturally PXD_{true_hit} are not available.
It is however possible to perform an alternative analysis to better understand the produced ROIs. The first step to study the two types of ROIs by comparing the distance between their centers on $\mathcal{P}$ and $\mathcal{N}$ side or, in other words, the residuals between the positions of the rows and columns. For this analysis 2×10^4 events were taken from run 1553 of experiment 18.

7.5.1 Summary of Belle II Runs and Encountered Problems

The running conditions at KEK are more complex compared to operating DATCON-Mini , but they also include a considerable amount of parameters over which limited actions are available. Multiple problems were encountered during data taking that are related to the FTB, to DATCON itself or to the connections with other systems.

Issues with FTB

One of the main issues with data received from FTB is the occurrence of broken frames. The FTB encapsulates FADC data into a specific protocol for transmission to DATCON in accordance with fig. 5.2. As described in section 4.3, frames are sent via Aurora link. On the receiver side, they are read out when SOF is detected. EOF, on the other hand, is asserted with the last received word. The DATCON receiver's state machine relies on these signals to process the incoming data, but also on the header and trailer.
A broken frame is a frame which starts correctly but was not entirely transmitted because it is truncated. This means that without the trailer word and without the EOF signal, the DATCON receiver interprets any incoming data to belong to the current event. If the receiver automatically recovers, the event data that overlaps is lost. It is suspected that very large events, by the number of strips, can cause this issue. In case of a missing event, the event builder waits infinitely, causing the entire DATCON to stop. A safety mechanism was implemented to artificially create an event based on the last known event, allowing to partially overcome this condition.
There are debugging tools such as ChipScope [1][52] which can help identify a problem in a firmware. For instance by monitoring the FIFO on the FTBs to detect overflow and therefore investigate its cause. It is, however, impossible to debug the FTBs due to their location on top of Belle II and to their operation at KEK.

Issues with DATCON

DATCON'running stability at KEK was tested over long period of data taking and found not to be flawless. It appears that one of the event builders on the $\mathcal{P}$ or $\mathcal{N}$ Tracking board, described in section 6.1, fails, thus paralyzing DATCON. The reason for the event builder to freeze is that one of the circular buffers has a different event ID than the others. Due to the rarity of such a case, finding the cause for an event mismatch is intricate. The already mentioned ChipScope can be helpful but is not sufficient to gain a complete overview of the status of the system.
After a run, available SVD data was used to be re-injected into DATCON-Mini to compare its output with the obtained ROIs during data taking. It appears that the ROIs obtained by the two systems are not entirely identical. Because DATCON-Mini does not include all the component of the final firmware, for instance the Aurora links, a full simulation of the entire system was made and confirms the difference between the results. Understanding the origin of ROI disparity is challenging for two reasons. The first one is that the SVD data are only available for events that were selected by the HLT. Therefore, it is impossible to re-run a

[1]Xilinx ChipScope tool inserts logic analyzer, and virtual I/O software cores directly into the design, allowing one to monitor internal signals.

one-to-one equivalent to a previous run. Secondly, DATCON is a complex system. With 15 boards running, three different firmwares and many more sub-modules, it becomes challenging to identify a cause of miscalculation. Since the only information available are the raw SVD data and the produced ROIs, many intermediate outputs are missing. Implementing the ROI calculation solution detailed in section 6.6 could discard possible sources of error.

Interconnection

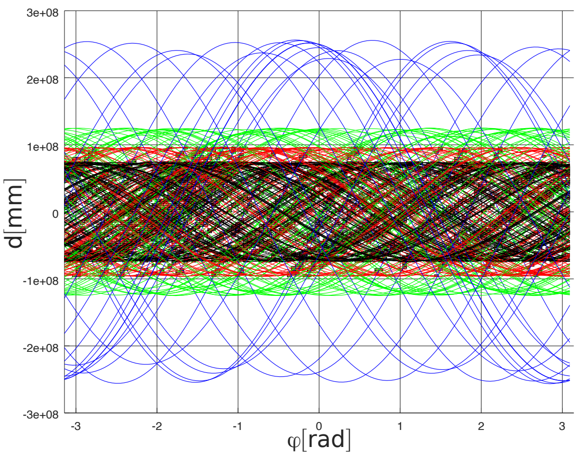
One of the major issues that can prevent DATCON from starting is if one of the Aurora links is down. Indeed, to ensure proper data transmission, every link has to be up. This includes the ones from all the FTBs, the ones between the $\mathcal{P}$ and $\mathcal{N}$ chassis and finally the one to ONSEN.
When starting a new run, these systems and their links are reset. It can happen that links with FTB or ONSEN remain down after the reset was deasserted. A manual action easily fixes the situation but requires constant attention. The DATCON slow control, detailed in Section 8.4.4, was adapted to automatically reset the link until it is up.

7.5.2 SVD Noise Filtering

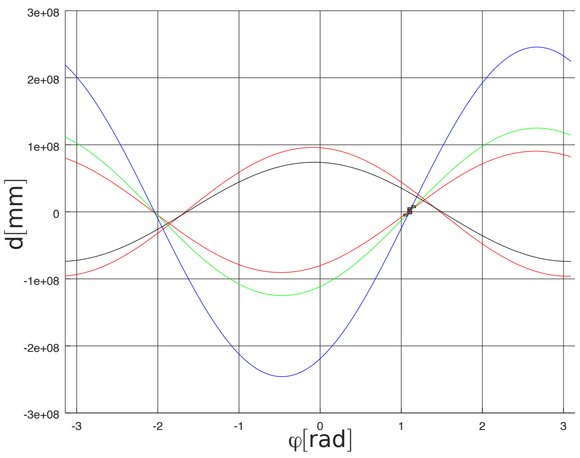
With the simulations used in section 7.4.2, the included background was lacking the electric noise of the strips. Nevertheless, for the Belle II runs, the strip filtering, introduced in section 5.2.3, based on the recorded noise has to be used. From previous runs, the list of noisy strips is obtained. Based on the noise, a threshold value corresponding to an SNR of 5 is pre-calculated and written into a Verilog LUT. Compared to the simulated event, such a filtering is absolutely necessary . An example of $\mathcal{P}$ HS before and after filtering is shown in fig. 7.20.
Without the SNR filtering the HS becomes extremely populated and clusters are found all over the φ range, resulting in extrapolated hits and ROIs on every PXD ladder. From this state, no valuable tracking can be achieved nor any DRF obtained. The event shown in fig. 7.20a had only one track to be detected which only appears after the SNR filtering was applied. The strips that passed the filter are used to build the HS depicted in fig. 7.20b. In the current example, the number of strips was reduced from 328 to 5.

7.5.3 ROI Residuals

Having applied the necessary noise filtering in the Concentrator firmware, it is possible to perform the tracking and ROI calculation using the parameters introduced in section 7.4. Both HSs have a size of 128×64 cells, covering a φ range of [-180°,180°] and θ range of [20°,157°]. Every single HS cell cluster is rejected and the ROI size is set to 80×80 pixels. If column or



(a) $\mathcal{P}$ Hough space without SNR filtering.



(b) $\mathcal{P}$ Hough space with SNR filtering

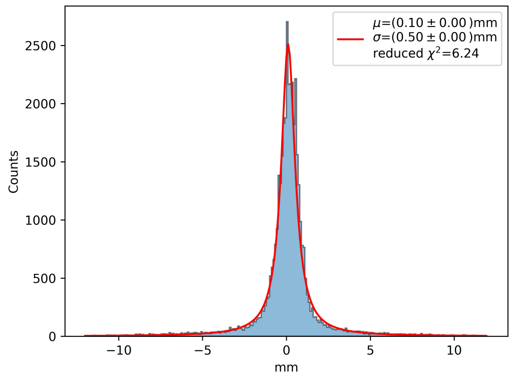
Figure 7.20: $\mathcal{P}$ Hough space generated for event 69 with and without the use of SNR strip filtering which shows to be indispensable. Most of the noisy strip do not reach a SNR of 5 and are discarded leaving a workable HS.

row components of two ROIs are overlapping they are automatically merged.
Out of the $2{\times}10^4$ events, only a selection had at least one ROI found by HLT, by DATCON or by both systems. The different quantities are summarized in table 7.4 For each HLT ROI, the

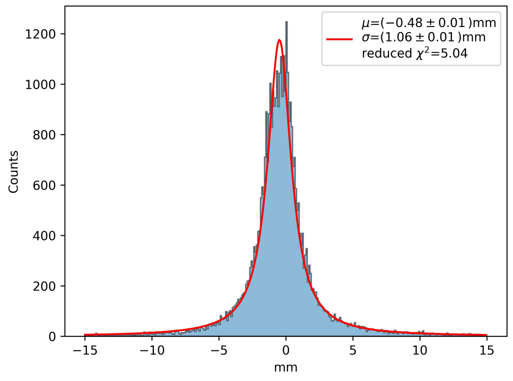
Events selected	20000
Events with DATCON and HLT ROIs	14307
Events with only HLT ROIs	2337
Events with only DATCON ROIs	633

Table 7.4: Reference settings used for event simulation.

column and row-wise comparisons are made with every DATCON ROI at the only condition that they are on the same PXD sensor. The ROI centers are computed thus creating a mean column value as well as a mean row value. Those mean values simply correspond to the $\mathcal{P}$ and $\mathcal{N}$ extrapolated hit coordinates as the ROI is built around them. The closest distance is saved as the best match between two ROIs. The ROI center residuals in mm are plotted in fig. 7.21 for $\mathcal{P}$ and $\mathcal{N}$ sides. The distributions are fitted based on the Cauchy model. Afterwards, the reduced χ^2 is calculated and for both the columns and the rows, reasonable values are obtained considering the low number of statistics. The Cauchy model is therefore qualified to extract the properties of the distributions. The residuals were extended to every PXD module and are depicted in fig. 7.22. In Belle II , only 20 PXD modules are installed. The layer 1 is complete with 8 ladders for a total of 16 modules, however on layer 2, only ladder 1 and 2 are present. The DHE IDs of the PXD modules are calculated based on eq. (6.8) are used as abscissa on the plot. Forward sensors have an even ID while the backward ID is odd, the latter appears



(a) Residual distribution of the columns center position of HLT and DATCON ROIs.



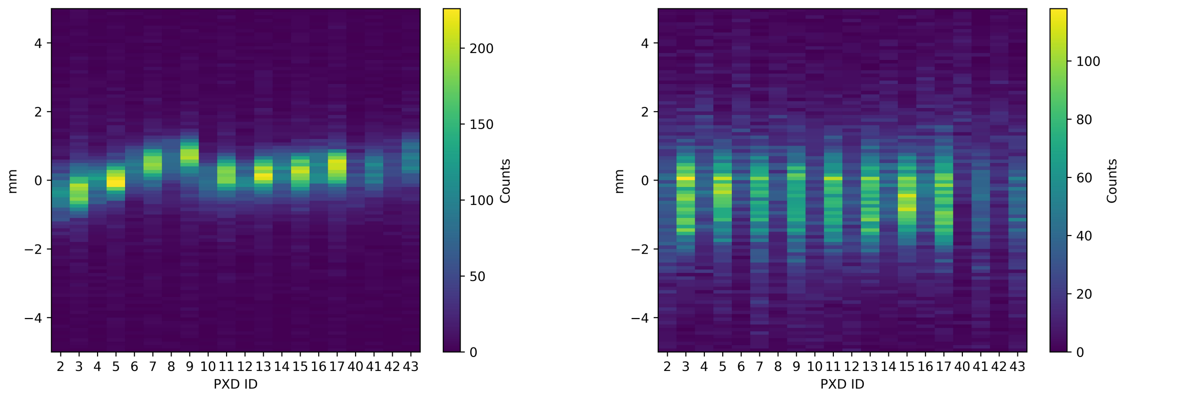
(b) Residual distribution of the rows center position of HLT and DATCONROIs.

Figure 7.21: ROI position residuals between DATCON and HLT.

to be the direction where most of the ROIs are created. It is a direct effect of the PXD being shifted along the z-axis towards the forward region.

7.5.4 Achieved Data Reduction

After investigating the ROI center position residuals, the DRFs from DATCON and HLT are compared. The method of calculation is detailed in section 7.4.2. In fig. 7.23, the DRF distribution of the two system is shown. A large difference in the distribution profiles is visible, DATCON has a peak. The first difference to be noticed is that the minimum DRF provided by DATCON is 1.03 whereas HLT is 6.94. The second is the compliance with the requirement of a DRF at 10 which is respected for HLT with 99.99% of the events (with at least one ROI, see table 7.4), while DATCON has only 84.26%. It should be noted that the events DATCON could process are only the ones that were stored on EB2 (see section 2.3.2). It implies that many events for which DATCON could have potentially found a ROI are already discarded. For the selected run, 82.23% of the events did not pass the HLT decision. Based on the ROI residuals and the DRF in the order of magnitude of the requirement, the capabilities of the DATCON algorithm are validated. Figure 7.24 shows the number of ROIs produced by DATCON against the number of ROIs produced by HLT for the selected events. Even with the ROI merging, introduced in section 7.4.3, the number of DATCON ROIs, which includes a considerable number of fakes as described in section 7.4.3, is large and contributes to lowering the DRF.



(a) Residual distribution of the columns center position of HLT and DATCON ROIs.

(b) Residual distribution of the rows center position of HLT and DATCON ROIs.

Figure 7.22: Residual distribution of the DATCON ROI centers with the HLT ROI centers on $\mathcal{P}$ and $\mathcal{N}$ side for every PXD module.

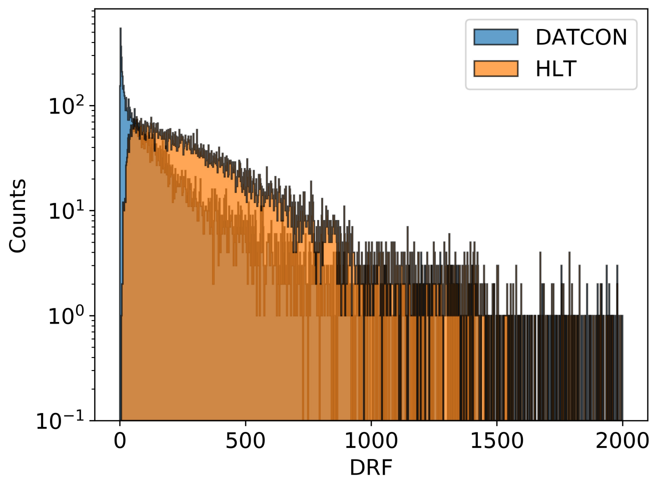


Figure 7.23: DRF distribution for HLT and DATCON.

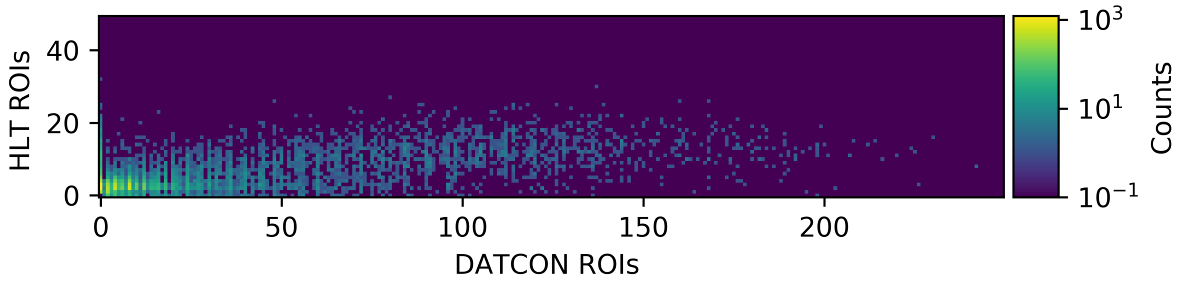


Figure 7.24: Number of HLT ROIs vs the number of DATCON ROIs

7.5.5 Adapted RFE

With the Belle II data it is not possible anymore to compute the RFE as it was done in section 7.1.1. A new approach to estimate the RFE is made with the following definition. Considering the HLT ROIs as the reference, for each HLT ROI on a PXD sensor, the possible

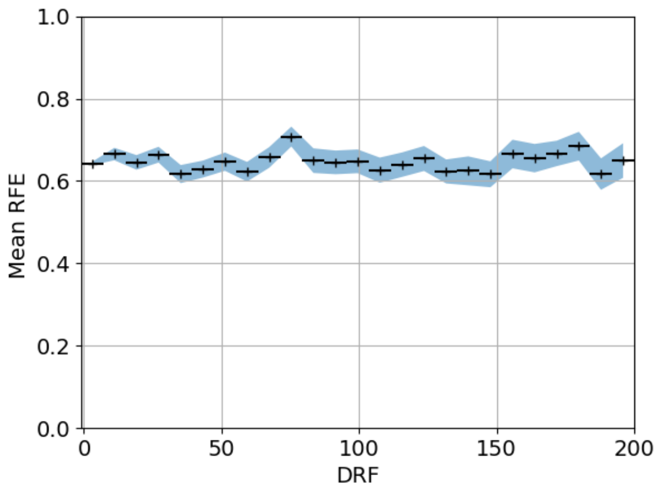


Figure 7.25: Mean RFE vs DRF.

overlapping with a DATCON ROI is calculated. If an overlap exists, the HLT ROI is counted as found. The ratio of overlapping ROIs over the total number of HLT ROIs becomes an estimation of the RFE. This adapted RFE is combined with the DRF using the same method as in section 7.4.4 and is shown in fig. 7.25. The DRF bins stops at 200 since the number of events after are limited. The distribution of the mean RFE slightly oscillates around 64.5 %. Two factors can explain the low value. First, it is possible that DATCON has not produced any ROI on the current PXD sensor. Secondly, DATCON can produce a ROI that do not overlap with an HLT. Finally, the case where DATCON finds a ROI but HLT does not also exist, as shown in table 7.4. These ROIs are of great interest and thorough studies are necessary to characterize them.

8 Slow Control

As introduced in Chapter 2, Belle II is a complex detector system composed of multiple sub-detectors that all have to operate in harmony. For this reason, every subsystem including DATCON has its specific SC which provides monitoring and control over a network by reading and writing Process Variable (PV)s. This chapter describes the key components used for the Belle II SC system focusing on the development and functioning of DATCON SC.

8.1 EPICS

A unified slow control mechanism, Experimental Physics and Industrial Control System (EPICS), was adopted to ease communication between sub-systems. The EPICS [53] is a collection of software and tools used to create Distributed Control System (DCS). It provides necessary features of DCS, for example remote control and monitoring, sequencing of operations or alarm detection, reporting and logging.
The EPICS tool set allows for the creation of server and client applications where servers, for instance the DATCON hardware, provides data to the client. One of the key component of an EPICS based DCS is the IOC.

8.2 CSS

The Control System Studio (CSS) [54] is a set of tools and applications used to monitor and operate a DCS. It offers multiple data connection layers and supports the EPICS channel access and pvAccess protocols. The CSS allows the users to build their own user interface which provides easy monitoring and configuration of the different PVs.

8.3 IPBus

For a long time physics experiments were based on the VMEbus standard. Over time it has been intensively replaced by ATCA and μTCA that includes industry communication standards, such as Gigabit Ethernet. Ethernet is a good solution for control communication with

the disadvantage of not having a dedicated hardware control protocol. From the idea of controlling hardware via Ethernet with a low resource impact, the IPbus protocol and associated firmware was developed and introduced by J. Mans et al. in 2009. Such a solution offers a high scalability which can therefore be ported on a large number of boards as often seen in HEP experiment.
Respecting the requirement of a light implementation on hardware, User Datagram Protocol (UDP) has been chosen as the transport protocol as an alternative to Transmission Control Porotocol (TCP). While the latter has profitable features, for instance error check or re-transmission of lost packet, the fact remains that it is a complex protocol to implement on hardware. The IPbus protocol [55] is a simple transactional protocol between a client (computer) and an host (hardware). Every request comes with a specific response allowing for request confirmation. The adoption of confirmations compensates the low reliability of UDP compared to TCP. The protocol relies on read or write transactions of FIFOs as well as adding or modifying values at a specific address location.
The IPbus suite is composed of the firmware that can interprets the protocol transaction on FPGA and a software part that includes the μHAL which is a Hardware Access Library (HAL) that offers a C++ and Python Application Programming Interface (API) for the read and writes operations.

8.4 DATCON Slow Contol

In parallel to the development of DATCON firmware for tracking and data reduction, the entire dedicated slow control was developed and implemented. Only the IOC, taken from the PXD DAQ cards, could directly be reused. Building the slow control can be divided into three distinct parts. First, by defining the PVs and building the corresponding database for the running IOC. Secondly by including IPbus to the firmware with the required connections. Finally by designing a Graphical User Interface (GUI) used to monitor the DATCON system.

8.4.1 IOC and Database

The core of an EPICS IOC is the process memory resident database, not to mix up with common relational database like SQL. The IOC functionality is directly given by the database which is composed of multiple records. A record, with a unique name, has a specific behavior defined by its type. There are a few types of interest for DATCON. The **longin** retrieves a 32 bit long integer, identical to IPbus registers. Type **calc** is used to computes functions from other records. For instance, the logical `or` of Aurora link is used to set the DATCON state.

8.4.2 SC firmware

The implementation of the SC firmware for DATCON is divided into two distinct parts, one for the Concentrator boards and one for the Tracking. They, however share a large amount of similarities thus no distinction is made in this section.
Crucial FPGA signals to monitor or assert are firstly listed and then connected to IPbus registers. These registers can later receive the read or write transactions. DATCON is an autonomous system, in the sense that it automatically starts processing SVD data as they arrived. Except for sending various resets signals, the main task of the SC is to read registers containing relevant information about the system's state. The most common internal signals are the optical link status or, for instance, the link data rate.
For each firmware the IPbus core is implemented and slaves are instantiated, for that purpose, simple 32-bit registers for elementary monitoring. To implement the IPbus protocol a set of addresses are necessary, the Media Access Control (MAC) and the IP address. Each DATCON AMC thus has its unique pair of addresses. The upper bits of the addresses are fixed but the 8 last ones of the IP address are given by the User Access Register[56] during the bitstream file generation. This allows to change address of the FPGA without having to recompile the entire source code (see section 4.2).

8.4.3 Integration to PXD Slow Control

DATCON is accessible via the PXD SC, itself being part of the Belle II SC, therefore it needs to respect some conventions. Firstly, every PV has to have the form :

$$PXD : T : XY : PV_{name}, \tag{8.1}$$

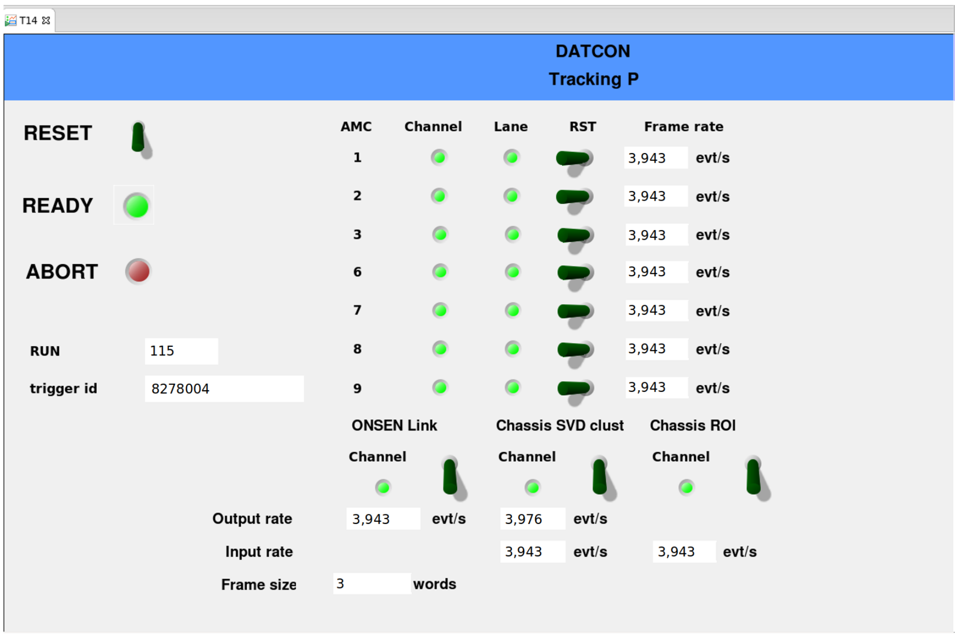
where X is the chassis identification number, 1 for $\mathcal{P}$ side and 2 for $\mathcal{N}$ side chassis. The slot number within the shelf [1:9], is referred to by Y. The position of each AMC in the chassis is depicted in fig. 4.9. From the value of X and Y, a unique IP address is also assigned to every DATCON AMC as described in section 8.4.2. Each IP is assigned by following:

$$10.16.5.XY \tag{8.2}$$

The different PVs are only of interest when looking at DATCON specifically. In the large structure of the Belle II SC, sub-systems, including DATCON, are running their Run Control (RC). The RC reflects the global status of the system based on PV condition. There are three possible states for the RC: `NOT-READY`, `READY` and `RUNNING`.

8.4.4 Slow Control Interface

Every AMC also has its own dedicated user Operator Interface (OPI) which is a general GUI but with extra features to directly connect to live data. There are 3 models of OPI. One for the Concentrator, one for the $\mathcal{P}$ Tracking and one for the $\mathcal{N}$ Tracking. Figure 8.1 shows the interface of the $\mathcal{P}$ side Tracking AMC during data taking. Each of the tree OPI shows the status of every Aurora optical or electrical link. On fig. 8.1, the seven links to the Concentrators, the ones to exchange SVD strips and ROIs with the $\mathcal{N}$ chassis are shown. In addition to the link status, their respective rate is shown.
The state of the board is computed as the logical `AND` of all the link status and controls the `READY` PV. Any down link, or reset activated will activate an `ABORT`. Only the `READY` and `ABORT` PVs of an AMC are necessary for the DATCON RC.


T14

DATCON
Tracking P

RESET

READY ●

ABORT ●

RUN	115
trigger id	8278004

AMC	Channel	Lane	RST	Frame rate	
1	●	●		3,943	evt/s
2	●	●		3,943	evt/s
3	●	●		3,943	evt/s
6	●	●		3,943	evt/s
7	●	●		3,943	evt/s
8	●	●		3,943	evt/s
9	●	●		3,943	evt/s

	ONSEN Link		Chassis SVD clust		Chassis ROI	
	Channel ●		Channel ●		Channel ●	
Output rate	3,943	evt/s	3,976	evt/s		
Input rate			3,943	evt/s	3,943	evt/s
Frame size	3	words				

Figure 8.1: OPI of the $\mathcal{P}$ side Tracking board

9 Conclusion

On February 2016, the first beam of positrons was circulated in the low-energy ring of the SuperKEKB collider. It has since reached a world record luminosity of $3.81 \times 10^{34} cm^{-2} s^{-1}$. However, the level of beam related backgrounds is considerable

Located at 14 mm from the interaction point, the DEPFET Pixel Detector (PXD) is highly sensitive to those backgrounds which are expected to be responsible for up to 3 % of its occupancy. The resulting data rate of about 20 Gbps, well beyond the capabilities of the storage system, therefore needs to be significantly reduced.

An online data reduction system, DATCON, running exclusively on FPGA, which uses the hits produced by the Silicon Vertex Detector (SVD), was developed in this thesis. The strips are referred as $\mathcal{P}$ and $\mathcal{N}$ depending on their implantation type. A pre-processing was necessary to removed noisy strips, thereby reducing the computational load to perform tracking. The tracking method is based on a Hough Transformation, capable of detecting lines from a subset of 2D coordinates. It generates a so-called Hough Space from which the lines parameters appear. Two distinct track reconstructions are performed, one with the $\mathcal{P}$ stips to reconstruct ϕ and track radius, the second with $\mathcal{N}$ strips to find θ. In this thesis, the emphasis on separating $\mathcal{P}$ and $\mathcal{N}$ is made and is respected through the whole processing chain. In the form of a binary matrix, the Hough Space is searched to extract track parameters using a graph search algorithm. The track candidates are then extrapolated back onto the PXD sensors. Around the intersection points of the extrapolated tracks on the PXD layers, a region of interest, or ROIs are built. Multiple mathematical calculations are necessary for the extrapolation, and are, by far more complex to implement on an FPGA than they would have been on a computer. A mix of hard-coded function output or the use of Xilinx cores were hence included.

The full DATCON system relies on the μTCA standard and consists of two sorts of AMCs with embedded FPGAs dispatched over two chassis. Connected with optical fibers, thirteen Concentrator AMCs are used to pre-process the SVD data coming from 52 FTBs. The FTB cards are used to transmit the strips signals digitized by the FADC boards. Two Tracking AMCs perform track finding, extrapolation and ROI building. The ROIs are sent to the ONSEN system which merges them with High Level Trigger (HLT) ROIs and applies the selection to the PXD data. Considering the large number of connections involved, event building and synchronization of the data inside DATCON was essential during development.

To ease testing and validation of the various modules used by DATCON, a small-scale version,

the DATCON-Mini , was built in parallel to the main project. It contrasts to the full system by using a single pipelined chain without all the interconnections, and can be run in only one AMC. Data injected to the DATCON-Mini were taken, either from previous Belle II runs or from MC simulations. A significant part of the thesis has been devoted to test the DATCON performance under various conditions provided by MC simulation in order to improve the currently running firmware. DATCON capabilities are mainly characterized by two parameters : the Data Reduction Factor (DRF), and the ROI Finding Efficiency (RFE) which naturally depend on each other. Finally, data from Belle II was re-processed to compare DATCON and HLT outputs.
To run in the Belle II experiment, a dedicated slow control was implemented to allow DATCON to follow the Belle II operations but also to transmit its state. During data taking, various issues were observed, some of which could be solved whereas some need further investigation. With simulated $\Upsilon 4S$ events, DATCON is capable of reducing the amount of PXD data by an average factor of 3.5 while the RFE stays above 85 %. While the obtained DRF and RFE are not sufficient for a standalone data reduction by DATCON, it was shown for the first time in this thesis that an FPGA-based online data reduction works and that there is potential for improvement. For instance, a better space point representation of the strips, a variable ROI size depending on the track momentum, 3D track reconstruction or to favor pre-calculation for the extrapolation of tracks with usage of LUTs to improve the efficiency.
Currently, DATCON is capable of performing track finding and ROI creation in addition the the HLT. Additional studies will be made in the future to characterize the DATCON ROI in comparison with the ones from HLT.

Acronyms

ADC	Analog-to-Digital Converter
ADU	Analog Digil Unit
AMC	Advanced Mezzanine Card
ASIC	Application Specific Circuit
ATCA	Advanced Telecommunications Computing Architecture
API	Application Programming Interface
ARICH	Aerogel Ring Imaging Cherenkov
BASF	Belle AnalysiS Framework
BASF2	Belle Analysis Software Framework 2
BRAM	Block Random Access Memory
CBM	Compressed Baryonic Matter
CDC	Central Drift Chamber
CERN	European Organization for Nuclear Research
CLB	Configurable Logic Block
CM	Common Mode
CMC	Common Mode Correction
CMOS	Complementary Metal Oxide Semiconductor
CMS	Compact Muon Solenoid
CoG	Center of Gravity
COPPER	Common Pipeline Platform for Electronics Readout

CPP	C++
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSS	Control System Studio
DATCON	Data Acquisition Tracking and Concentrator Online Node
DAQ	Data Acquisition
DCD	Drain Current Digitizer
DCM	Digital Clock Management
DDR2	Double Data Rate 2
DESY	Deutsches Elektronen SYnchroton
DEPFET	DEpleted p-channel Field Effect Transistor
DHP	Data Handling Processor
DHH	Data Handling Hybrid
DHHC	Data Handling Hybrid Controler
DIMM	Dual Inline Memory Module
DRF	Data Reduction Factor
DSP	Digitial Signal Processor
DSSD	Double-Sided Strip Detector
DQM	Data Quality Monitoring
DFS	Depth First Search
DHE	Data Handling Engine
DR	Damping Ring
DCS	Distributed Control System
DHI	Data Handling Isolator
DHC	Data Handling Concentrator

EEPROM	Electrically Erasable Programmable Read-Only Memory
EPICS	Experimental Physics and Industrial Control System
EB	Event Builder
EB0	Event Builder 0
EB1	Event Builder 1
EB2	Event Builder 2
ENC	Equivalent Noise Charge
ECL	Electromagnetic Calorimeter
EOF	End Of Frame
FADC	Flash Analog to Digital Converter
FEE	Front End Electronics
FF	Flip Flop
FIFO	First-In First-Out
FPGA	Field Programmable Gate Arrays
FPU	Floating Point Unit
FSM	Finite State Machine
FTB	Finesse Transmitter Board
FTSW	Fast Timing Switch
FHT	Fast Hough Transformation
FIR	Finite Impulse Response filter
GDL	Global Decision Logic
GPIO	General Purpose Input/Outputs
GUI	Graphical User Interface
HDL	Hardware Description Language
HER	High Energy Ring

HEP	High Energy Physics
HLT	High Level Trigger
HS	Hough Space
HT	Hough Transformation
HTTP	HyperText Transfer Protocol
HV	High Voltage
IBL	Insertable B-Layer
IEEE	Institute of Electrical and Electronics Engineers
IHEP	Institute of High Energy Physics
IOB	Input/Output Buffers
IP	Interaction Point
IPMI	Intelligent Platform Management Interface
IC	Integrated Circuit
IOC	Input/Output Controllers
JTAG	Joint Test Action Group
JSM	JTAG Switch Module
KEK	High Energy Accelerator Research Organization
KLM	KLM
LHC	Large Hadron Collider
LER	Low Energy Ring
LVDS	Low Voltage Differential Signal
LUT	Lookup Table
LINAC	Linear Accelerator
MIP	Minimum Ionizing Particle
MOSFET	Metal Oxide Field Effect Transistor

MPH	Most Probable Hit
μTCA	Micro Telecommunications Computing Architecture
HAL	Hardware Access Library
MGT	Multi-gigabit Transceiver
MCH	MicroTCA Carrier Hub
MAC	Media Access Control
MC	Monte Carlo
MMC	Module Management Controller
NIM	Nuclear Instrumentation Standard
OPI	Operator Interface
RAM	Random Access Memory
RBB	Radiative Bhabha
ROI	Region of Interest
RPC	Resistive Plate Chamber
ONSEN	ONline SElector Node
OSI	Open Systems Interconnection model
PC	Personal Computer
PCB	Printed Circuit Board
PCIe	Peripheral Component Interconnect express
PICMG	PCI Industrial Computer Manufacturers Group
PXD	Pixel Detector
PM	Power Module
PF	Photon Factory
PF-AR	PF Advanced Ring
PID	Particle Identification

PV	Process Variable
ROPC	Readout PC
RTL	Register Tranfer Level
ROM	Read Only Memory
RPC	Resistive Plate Chambers
RFE	ROI Finding Efficiency
RC	Run Control
SM	Standard Model
SFP	Small Form Plugable
SVD	Silicon Vertex Detector
SM	Standard Model
SNR	Signal-to-Noise Ratio
SR	Synchrotron Radiation
SPS	Super Proton Synchroton
SUSY	Supersymmetry
SC	Slow Control
SCM	Sector Check Module
SOF	Start Of Frame
SiPM	Silicon PhotoMultiplier
TC	Track Candidate
TCP	Transmission Control Porotocol
TLU	Trigger Logic Unit
TTD	Trigger Timing Distribution
TOP	Time of Propagation
TTL	Transistor to Transistor Logic

TAP	Test Access Port
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus
VME	Versa Module Europa
VXD	Vertex Detector
VHDL	Very High Speed Integrated Circuit Hardware Description Language

Bibliography

[1] Georges Aad et al. “Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC”. In: *Physics Letters B* 716.1 (2012), pp. 1–29.

[2] Andrei D Sakharov. “Violation of CP-invariance, C-asymmetry, and baryon asymmetry of the Universe”. In: *In The Intermissions... Collected Works on Research into the Essentials of Theoretical Physics in Russian Federal Nuclear Center, Arzamas-16*. World Scientific, 1998, pp. 84–87.

[3] James H Christenson et al. “Evidence for the 2 π Decay of the K 2 0 Meson”. In: *Physical Review Letters* 13.4 (1964), p. 138.

[4] Makoto Kobayashi and Toshihide Maskawa. “CP-violation in the renormalizable theory of weak interaction”. In: *Progress of theoretical physics* 49.2 (1973), pp. 652–657.

[5] Tetsuo Abe et al. “Achievements of KEKB”. In: *Progress of Theoretical and Experimental Physics* 2013.3 (2013), 03A001.

[6] A Abashian et al. “The belle detector”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 479.1 (2002), pp. 117–232.

[7] Bernard Aubert et al. “The BABAR detector”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 479.1 (2002), pp. 1–116.

[8] Michael Schnell. “Development of an FPGA-based Data Reduction System for the Belle II DEPFET Pixel Detector”. In: (2015).

[9] Tetsuo Abe et al. “Belle II technical design report”. In: *arXiv preprint arXiv:1011.0352* (2010).

[10] P. Raimondi. “New developments in super B-factories”. In: *2007 IEEE Particle Accelerator Conference (PAC)*. 2007, pp. 32–36. DOI: `10.1109/PAC.2007.4440329`.

[11] J Kemmer and Gerhard Lutz. “New detector concepts”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 253.3 (1987), pp. 365–377.

[12] Andreas Moll et al. “The vertex detector numbering scheme”. In: *Belle II Note* (2016).

[13] Michel Raymond et al. “The APV25 0.25 /spl mu/m CMOS readout chip for the CMS tracker”. In: *2000 IEEE Nuclear Science Symposium. Conference Record (Cat. No.00CH37149)* 2 (2000), 9/113–9/118 vol.2.

[14] Dehui Sun, Jingzhou Zhao, Hao Xu, et al. “Belle2Link: A global data readout and transmission for Belle II experiment at KEK”. In: *Physics Procedia* 37 (2012), pp. 1933–1939.

[15] Kenji Inami and II Belle. “TOP counter for particle identification at the Belle II experiment”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 766 (2014), pp. 5–8.

[16] R Pestotnik et al. “Aerogel RICH for forward PID at Belle II”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 732 (2013), pp. 371–374.

[17] M Yamaga et al. “RPC systems for BELLE detector at KEKB”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 456.1-2 (2000), pp. 109–112.

[18] T Aushev et al. “A scintillator based endcap KL and muon detector for the Belle II experiment”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 789 (2015), pp. 134–142.

[19] M Nakao. “Timing distribution for the Belle II data acquistion system”. In: *Journal of Instrumentation* 7.01 (2012), p. C01028.

[20] SY Suzuki et al. “The three-level event building system for the Belle II experiment”. In: *IEEE Transactions on Nuclear Science* 62.3 (2015), pp. 1162–1168.

[21] R. Itoh et al. “Data Flow and High Level Trigger of Belle II DAQ System”. In: *IEEE Transactions on Nuclear Science* 60.5 (2013), pp. 3720–3724. DOI: 10.1109/TNS.2013.2273091.

[22] Dmytro Levit et al. “FPGA based data read-out system of the Belle II pixel detector”. In: *IEEE Transactions on Nuclear Science* 62.3 (2015), pp. 1033–1039.

[23] Stefan Huber et al. “Performance of the Data-Handling Hub Readout System for the Belle II Pixel Detector”. In: *IEEE Transactions on Nuclear Science* 68.8 (2021), pp. 1961–1967. DOI: 10.1109/TNS.2021.3083720.

[24] Thomas Geßler et al. “The ONSEN data reduction system for the Belle II pixel detector”. In: *IEEE Transactions on Nuclear Science* 62.3 (2015), pp. 1149–1154.

[25] Paul VC Hough. “Machine analysis of bubble chamber pictures”. In: *Proc. of the International Conference on High Energy Accelerators and Instrumentation, Sept. 1959.* 1959, pp. 554–556.

[26] Richard O. Duda and Peter E. Hart. “Use of the Hough transformation to detect lines and curves in pictures”. In: *Communications of the ACM* 15.1 (Jan. 1972), pp. 11–15. DOI: 10.1145/361237.361242.

[27] C.Wessel. *Private communication.*

[28] *Virtex-5 Familly Overview.* URL: https://www.xilinx.com/support/documentation/data_sheets/ds100.pdf.

[29] *Virtex-6 Familly Overview.* URL: https://www.xilinx.com/support/documentation/data_sheets/ds150.pdf.

[30] *Spartan-3 FPGA FamilyData Sheet.* URL: https://www.xilinx.com/content/dam/xilinx/support/documentation/data_sheets/ds099.pdf.

[31] *ISE In-Depth Tutorial.* 2011. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx13_3/ise_tutorial_ug695.pdf.

[32] Abhijit Athavale. *High-Speed Serial I/O Made Simple A Designers’ Guide, with FPGA Applications.* 2021.

[33] *8B/10B Protocol Specification*. 2014. URL: https://www.xilinx.com/support/documentation/ip_documentation/aurora_8b10b_protocol_spec_sp002.pdf.

[34] *Virtex-5 FPGA RocketIO GTP Transceiver User Guide*. 2009. URL: https://www.xilinx.com/support/documentation/user_guides/ug196.pdf.

[35] *Virtex-6 FPGA GTX Transceivers User Guide*. 2011. URL: https://www.xilinx.com/support/documentation/user_guides/ug366.pdf.

[36] *Micro TelecommunicationsComputing ArchitectureShort Form Specification*. URL: https://www.picmg.org/wp-content/uploads/MicroTCA_Short_Form_Sept_2006.pdf.

[37] *microTCA overview*. URL: https://www.vadatech.com/media/pdf_MicroTCA_Overview.pdf.

[38] I.Konorov. *Private communication.*

[39] "IEEE Standard for Floating-Point Arithmetic". In: *IEEE Std 754-2019 (Revision of IEEE 754-2008)* (2019), pp. 1–84. DOI: 10.1109/IEEESTD.2019.8766229.

[40] Thomas H Cormen et al. *Introduction to algorithms*. MIT press, 2009.

[41] *Microblaze*. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/mb_ref_guide.pdf.

[42] SF Obermann and Michael J Flynn. *An Analysis of Division Algorithms and Implementations*. Tech. rep. Technical Report CSL-TR-95-675, Stanford University, 1995.

[43] D Kumar, P Saha, A Dandapat, et al. "Hardware implementation of methodologies of fixed point division algorithms". In: *International Journal on Smart Sensing and Intelligent Systems* 10.3 (2017), pp. 630–645.

[44] Jack Volder. "The CORDIC computing technique". In: *Papers presented at the the March 3-5, 1959, western joint computer conference*. 1959, pp. 257–261.

[45] John S Walther. "A unified algorithm for elementary functions". In: *Proceedings of the May 18-20, 1971, spring joint computer conference*. 1971, pp. 379–385.

[46] *GNU Octave*. URL: https://www.gnu.org/software/octave/index.

[47] *QuestaSim*. URL: https://www.microsemi.com/document-portal/doc_view/131619-modelsim-user.

[48] Andreas Moll. "The software framework of the Belle II experiment". In: *Journal of Physics: Conference Series*. Vol. 331. 3. IOP Publishing. 2011, p. 032024.

[49] Tino Ullrich and Z Xu. "Treatment of errors in efficiency calculations". In: *arXiv preprint physics/0701199* (2007).

[50] Ryosuke Itoh. "BASF-Belle AnalysiS Framework". In: *Proceedings, 9th International Conference on Computing in High-Energy Physics (CHEP 1997)*. 1997.

[51] Sea Agostinelli et al. "GEANT4—a simulation toolkit". In: *Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 506.3 (2003), pp. 250–303.

[52] *ChipScope Pro Software and Cores*. 2012. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/chipscope_pro_sw_cores_ug029.pdf.

[53] *Experimental Physics and Industrial Control System*. URL: https://epics.anl.gov/about.php.

[54] Kay Kasemir et al. "Control system studio applications". In: *Proc. 11th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'07)*. 2007, pp. 692–694.

[55] C Ghabrous Larrea et al. "IPbus: a flexible Ethernet-based control system for xTCA hardware". In: *Journal of Instrumentation* 10.02 (2015), p. C02019.

[56] *Bitstream Identification*. URL: https://www.xilinx.com/support/documentation/application_notes/xapp497_usr_access.pdf.

List of Figures

ACKNOWLEDGMENT

First of all, I would like to thank Prof. Dr. Jochen Dingfelder for supervising my thesis and for providing continuous support.
I would also like to thank Dr. Carlos Marinas for introducing me to the world of Belle II and offering his great help.
Thank to Prof. Dr. Florian Bernlochner for proposing to continue developing the project.
I wish to thank Christian Wessel who worked in parallel on a software implementation of the Hough transform based tracking. His comments and ideas considerably helped me through this project.
I would also like to thank my colleagues of the Silab, Patrick, Georgios, Jannes and Botho for helping during the writing of my thesis. I would like to thank in particular Pascal, for his precious advice and for all his time that he has given.
I wish to thank my friends from Paris, Charles, Léa and Robin for the moral support they have offered during the repeated lock-downs.
Finally I would like to thanks my family, my parents and grand-parents who kept supporting me all along this challenging period.