DATCON: FPGA-based data reduction 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

Ralf Farkas

aus

Freiburg im Breisgau

Bonn, Oktober 2025

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

Gutachter/Betreuer:	Prof. Dr. Florian Bernlochner
Gutachter:	Prof. Dr. Jochen Dingfelder
Tag der Promotion:	02.03.2026
Erscheinungsjahr:	2026

Contents

Introduction

The current knowledge in the field of particle physics is summarized in the Standard Model (SM), which describes the fundamental particles and their interactions. The SM is the most successful theoretical description in physics. While its predictions have been verified with high precision, it is known to be incomplete. Many fundamental questions remain unanswered, like the nature of dark matter, the matter-antimatter asymmetry, or the unification of the fundamental forces. New physics beyond the Standard Model (BSM) aims to address these questions, and require sophisticated experiments to search for new particles and interactions and to evaluate theoretical models.

The Belle II experiment at the SuperKEKB accelerator at KEK in Tsukuba, Japan, is designed to study heavy flavour physics, with a primary focus on the decays of B mesons. By targeting the B meson pair production at the $\Upsilon(4S)$ resonance using asymmetric-energy e^--e^+-beams, Belle II can achieve high luminosities and precise measurements of the decay products of the B mesons.

To meet the physics requirements, an excellent vertex resolution is essential. The innermost subdetector of Belle II is specifically designed for this purpose. It is composed of the strip-based Silicon Vertex Detector (SVD) and the Depleted P-channel Field Effect Transistor (DEPFET) based Pixel Detector (PXD). During the first Long Shutdown 1 (LS1) in 2023, a new 2-layer PXD was installed, which now features 40 silicon modules with an active area of 250×768 pixels each. This will significantly enhance the accuracy of the reconstructed track parameters, but also presents significant challenges for the readout and data processing.

Because of the large number of pixels in the PXD, the high Level 1 trigger rate of 30 kHz, and the large background close to the beam line, the data rates of the PXD reach about 20 Gbit/s. This data rate needs to be reduced by a factor of $\sim$10, to allow transmission, processing and storage. Especially in high-occupancy events, important PXD information would be lost otherwise.

As part of the Belle II Data Acquisition (DAQ) system, **DATCON**, the Data Acquisition Tracking and Concentrator Online Node, is responsible for reconstructing the trajectories of particles as seen by the SVD, and to create regions of interest (ROIs) on the PXD sensors. Instead of the full PXD data, only the pixels contained in the ROIs will be processed and stored, making the data rates manageable.

This is the third thesis on DATCON, building on the work of Michael Schnell [1] and Bruno Deschamps [2]. It proposes an updated version of DATCON, which solves its previous shortcomings and provides a stable, tested system to be used throughout the next data runs of Belle II.

This thesis is structured as follows:

- *Chapter 1: The Belle II experiment* introduces the Belle II detector at the SuperKEKB accelerator. It covers the physics motivation, the SuperKEKB accelerator and the Belle II detector, with a particular focus on the PXD and the SVD, as their data are central to this work. The chapter concludes with an overview of the DATCON hardware system.
- *Chapter 2: Track reconstruction* describes the track reconstruction approach used by DATCON. It explains the principles of the Hough transformation, and details its specific application to reconstructing particle trajectories from SVD strip data.
- *Chapter 3: DATCON firmware* describes the firmware architecture of DATCON, following through the data flow from the SVD input to the final ROI output. Special emphasis is placed on the SVD segmentation, the major additions to DATCON, which is designed to suppress the creation of physically implausible ROIs.
- *Chapter 4: Performance* evaluates the performance of the updated DATCON system using simulated $\Upsilon(4S)$ events, by analysing the hit finding efficiency (HFE) and the data reduction factor (DRF) of various configurations.
- *Chapter 5: Conclusions and outlook* summarizes the results and gives an outlook on potential future enhancements.

Chapter 1

The Belle II experiment

1.1 Physics at the Belle II experiment

1.1.1 The Standard Model of particle physics

The current knowledge of particle physics is summarized in the Standard Model (SM) of Elementary Particle Physics. It describes the fundamental particles (quarks, leptons) and the interactions (weak, electromagnetic, and strong) among them. It is mathematically expressed through a Lagrangian density:

$$\mathscr{L}_{\text{SM}} = \mathscr{L}_{\text{gauge}} + \mathscr{L}_{\text{fermion}} + \mathscr{L}_{\text{Higgs}} + \mathscr{L}_{\text{Yukawa}} \quad ,$$

where

$$\mathscr{L}_{\text{gauge}} = -\frac{1}{4} G^{a}_{\mu\nu} G^{a\mu\nu} - \frac{1}{4} W^{a}_{\mu\nu} W^{a\mu\nu} - \frac{1}{4} B_{\mu\nu} B^{\mu\nu}$$

describes the massless gauge bosons (the eight gluons G, three weak isospin bosons W and the weak hypercharge boson B) and their self-interaction.

$$\mathscr{L}_{\text{fermion}} = \sum_{\text{quarks}} \mathrm{i}\bar{q}\gamma^{\mu} D_{\mu} q + \sum_{\psi_L} \mathrm{i}\overline{\psi_L}\gamma^{\mu} D_{\mu} \psi_L + \sum_{\psi_R} \mathrm{i}\overline{\psi_R}\gamma^{\mu} D_{\mu} \psi_R$$

describes the massless fermions (the quarks q, the left-handed doublets ψ_L and the right-handed singlets ψ_R) and their gauge interactions.

$$\mathscr{L}_{\text{Higgs}} = (D_{\mu}\Phi)^{\dagger} D_{\mu}\Phi + \mu^2 \Phi^{\dagger}\Phi - \lambda(\Phi^{\dagger}\Phi)^2$$

describe the dynamics of the Higgs field, a complex scalar doublet $\Phi = (\phi^{+}, \phi^{0})^{T}$. The final term,

$$\mathscr{L}_{\text{Yukawa}} = -Y_l \bar{L}\Phi \ell_R - Y_d \overline{Q}\Phi d_R - Y_u \overline{Q}\tilde{\Phi} u_R + \text{h.c.}$$

accounts for fermion mass generation.

The forces are mediated by the Gauge bosons: the photon γ for the electromagnetic interaction, the $W^{\pm}$ and Z^{0} bosons for the weak interaction, and the 8 gluons g for the strong interaction. The Higgs boson H^{0} is responsible for the mass of the particles. An overview of the fundamental particles is given in *Figure 1.1*.

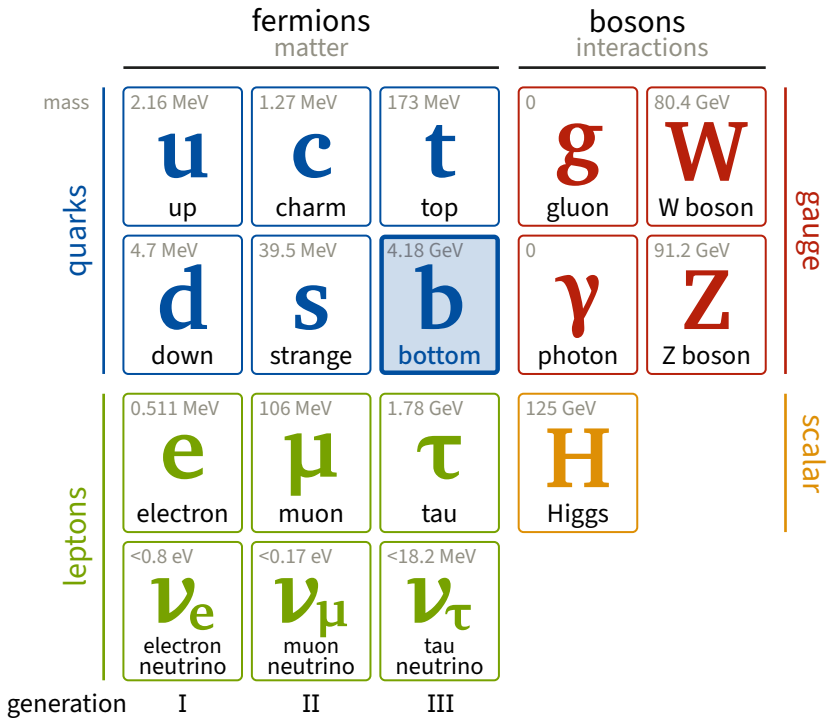


Figure 1.1 – Particles of the Standard Model. *The Standard Model classifies the fundamental particles into quarks, leptons and gauge bosons. Charged leptons participate in the electromagnetic (γ) and weak ($W^{\pm}$, Z^0) interactions, while the neutral neutrinos ν interact only weakly. Quarks participate in strong (g), electromagnetic and weak interactions. Inspired by [3].*

1.1.2 The CKM matrix

After electroweak symmetry breaking, the quark mass eigenstates differ from the weak interaction eigenstates, leading to mixing between generations. This mixing is encoded in the 3×3 unitary Cabibbo-Kobayashi-Maskawa (CKM) matrix [4, 5]

$$V_{\mathrm{CKM}} = \begin{pmatrix} V_{\mathrm{ud}} & V_{\mathrm{us}} & V_{\mathrm{ub}} \\ V_{\mathrm{cd}} & V_{\mathrm{cs}} & V_{\mathrm{cb}} \\ V_{\mathrm{td}} & V_{\mathrm{ts}} & V_{\mathrm{tb}} \end{pmatrix} \quad ,$$

where the matrix elements V_{ij} represent the coupling strength between the up-type quark i and the down-type quark j in charged-current weak interactions, mediated by the $W^{\pm}$ bosons.

The CKM matrix can be parametrized by four independent parameters, namely three real mixing angles ϕ_1, ϕ_2, ϕ_3 and one complex phase. The unitarity condition $V_{\mathrm{CKM}} V_{\mathrm{CKM}}^{\dagger} = \mathbb{1}$ leads to constraints on the matrix elements, most notably the unitarity triangle formed by

$$V_{\mathrm{ub}}^{*} V_{\mathrm{ud}} + V_{\mathrm{cb}}^{*} V_{\mathrm{cd}} + V_{\mathrm{tb}}^{*} V_{\mathrm{td}} = 0 \quad .$$

The unitarity triangle can be visualized in the complex plane, and its angles and sides can be measured experimentally *(Figure 1.2)*. At Belle II, the angle $\sin(2\phi_1)$ can be measured in time-dependent *CP* asymmetries in $b \to c\bar{c}s$ (like $B^0 \to J/\psi\, K_S^0$), ϕ_2 from $B^0 \to \rho^+\rho^-$ decays [6] and ϕ_3 from $B \to DK$ decays [7]. By over-constraining the CKM matrix elements, it is possible to test the consistency of the Standard Model: If the measured sides and angles would lead to a non-closing triangle, a direct hint for new physics (NP) would be given.

Further information about the CKM matrix and quark mixing can be found in [8].

1.1.3 *CP* symmetry violation

CP symmetry implies that all laws of physics should remain unchanged if the charge conjugation *C* (transforms a particle to its antiparticle), and the parity operation *P* (inverts the spatial coordinates / handedness) are applied to a particle, simultaneously. *CP* violation refers to the violation of the combined *C* and *P* symmetry.

While the CKM mechanism allows for *CP* violation, its magnitude is not sufficient to explain the observed baryon asymmetry in the universe. Other sources of *CP* violation thus must exist beyond the SM's CKM mechanism.

The first observation of *CP* violation occurred in 1964 in the decay of neutral kaons, $K_L^0 \to \pi^+\pi^-$, in an experiment led by Cronin and Fitch [10]. *CP* violation in the B-meson system was observed for the first time in 2001 by the Belle experiment at KEKB [11] and the BABAR experiment at PEP-II [12]. To date, *CP* violation has also been observed in D meson [13] and Λ_b^0 baryon [14] decays.

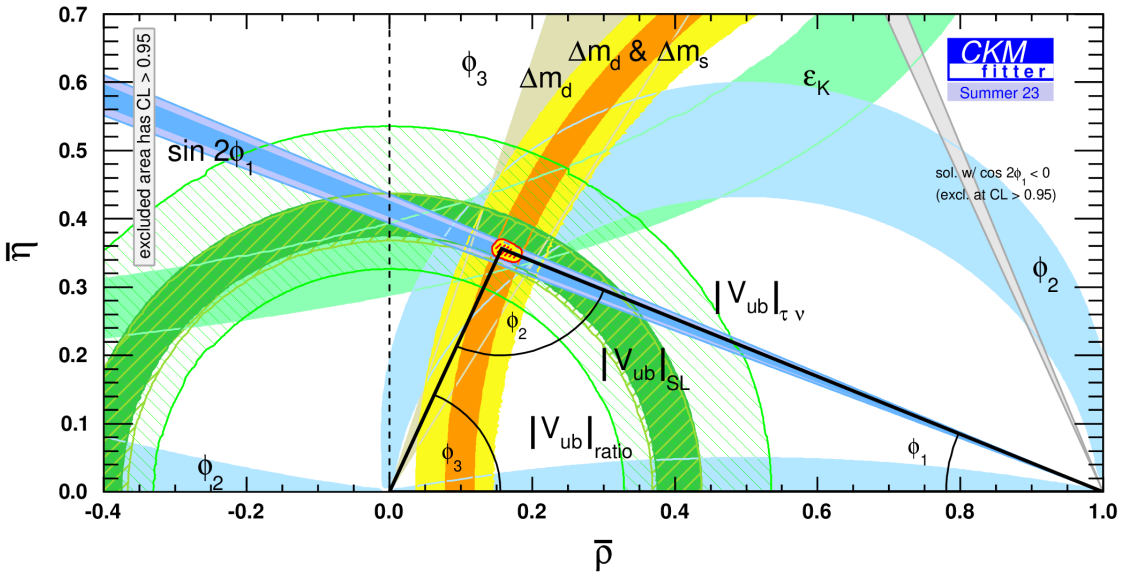


Figure 1.2 – Unitarity Triangle in the $\bar{\rho}$-$\bar{\eta}$-plane with current constraints. *Belle II constrains* $\sin 2\phi_1$ *from time-dependent CP violation,* ϕ_3 *from* $B \to DK$ *decays [7] and* ϕ_2 *from* $B^0 \to \rho^+\rho^-$ *[6]. Image from [9].*

1.1.4 B mixing

The neutral B^0 meson can oscillate into its antiparticle $\overline{B}^0$ (and vice versa) through weak interactions. This is known as *B mixing*. The mass eigenstates, the heavy B_H and the light B_L, are linear combinations of the flavour eigenstates B^0 and $\overline{B}^0$:

$$|B_{L,H}\rangle = p\,|B^0\rangle \pm q\,|\overline{B}^0\rangle \quad .$$

CP violation is reflected in an asymmetry $A_{CP}(\Delta t)$ in the time-dependent decay rates of the B^0 and $\overline{B}^0$ mesons to the same final state f, which is defined as

$$A_{CP}(\Delta t) = \frac{\Gamma(\overline{B}^0 \to f) - \Gamma(B^0 \to f)}{\Gamma(\overline{B}^0 \to f) + \Gamma(B^0 \to f)} = -\xi_f \sin 2\phi_1 \sin \Delta m_d \Delta t \quad ,$$

with ξ_f being the *CP* eigenvalue of the final state and Δm_d the mass difference of the mass eigenstates, which has been determined experimentally [8] as

$$\Delta m_d = 0.5069 \pm 0.0019\,\mathrm{ps}^{-1} \quad .$$

Measuring the decay rates Γ for specific final states, like $B^0 \to J/\psi\, K^0_S$, allows the determination of the CKM angle ϕ_1 and was used to establish the existence of *CP* violation in the B meson system [11].

1.2 The SuperKEKB accelerator

Two complementary experimental approaches can be pursued in particle physics in order to obtain a better understanding of the Standard Model and to search for phenomena beyond it. The first approach, pursued by the LHC experiments ATLAS and CMS, is at the *energy frontier*. By colliding protons at very high centre-of-mass energies of up to ∼14 TeV, these experiments can directly produce new heavy particles such as dark matter candidates. The scope of the new particles is limited by the beam energy.

The second approach is to pursue the *intensity frontier*, as Belle II at SuperKEKB and LHCb at the LHC do, where large data samples allow searches for rare processes. Small deviations from theoretical predictions can be interpreted as indirect evidence for new physics, as new heavy particles can contribute to these processes in higher-order corrections. The limiting factor is the data sample size.

The SuperKEKB accelerator is an upgrade of the KEKB accelerator, which was operated from 1998 to 2010 [15]. It is an electron-positron collider with asymmetric beam energies of 7 GeV for electrons and 4 GeV for positrons, with a centre-of-mass energy at the $\Upsilon(4S)$ resonance of ≈10.58 GeV. The $\Upsilon(4S)$ is a bound, excited state of $b\overline{b}$, and decays with a branching fraction of >96 % into a pair of B mesons, which can be either charged (B^+B^-, 51.4 %) or neutral ($B^0\overline{B}^0$, 48.6 %).

The asymmetric beam energies result in a Lorentz boost of the $\Upsilon(4S)$, with $\beta\gamma = 0.28$. The mass of the $\Upsilon(4S)$ is just slightly above the mass of a B meson pair, which are thus produced nearly at rest in the $\Upsilon(4S)$ frame, and boosted along the e^- beam ($+z$) in the lab frame. This boost creates a spacial separation Δz of the decay vertices of the two B mesons, which directly translates to Δt via

$$\Delta t = \frac{\Delta z}{\beta\gamma c} \quad .$$

The SuperKEKB accelerator complex, shown schematically in *Figure 1.3*, consists of several stages to produce and collide the e^+ and e^- beams. The process begins with the generation of electrons from a photocathode radio-frequency (RF) gun, an upgrade from the thermionic gun used at KEKB that yields a significantly lower beam emittance ϵ. The electron beam can be directed onto a fixed tungsten target to produce positrons. Both electron and positron beams are subsequently accelerated by the linear accelerator (LINAC). For SuperKEKB, a new Damping Ring (DR) was included for the positron beam, which reduces its emittance before being injected into the main rings. From the LINAC, the e^- are injected into the High Energy Ring (HER), and the e^+ into the Low Energy Ring (LER), having both a circumference of $\approx$3 km. After acceleration, the beams are brought to collision at the interaction point (IP) at the centre of the Belle II detector.

A primary objective of the SuperKEKB upgrade is to achieve a substantial increase in luminosity. The instantaneous luminosity can be expressed as [16]

$$L = \frac{\gamma_\pm}{2er_e}\left(1 + \frac{\sigma_y^*}{\sigma_x^*}\right)\frac{I_\pm \xi_{y\pm}}{\beta_y^*}\,\frac{R_L}{R_{\xi_y}} \quad ,$$

where $\gamma_\pm$ is the Lorentz factor for the beams, $I_\pm$ the beam current, $\xi_{y\pm}$ the vertical beam-beam tune-shift parameter, and β_y^* the value of the beta function at the IP. σ_x^* and σ_y^* represent the horizontal and vertical beam sizes at the IP. The parameters R_L and R_{ξ_y} are geometric correction factors, r_e the electron radius, and e the elementary charge. The beam size depends on the emittance ϵ_y and the beta function β_y^* at the IP:

$$\sigma_y^* = \sqrt{\epsilon_y \beta_y^*} \quad .$$

Compared to KEKB, the beam energies were changed from 8 GeV to 7 GeV for the e^-, and 3.5 GeV to 4 GeV for the e^+. The reduced Lorentz boost from $\beta\gamma = 0.42$ to 0.28 is necessary to implement the *nano-beam scheme* [17], in which the vertical size of the beams at the IP is squeezed by strong focusing to $\sigma_y^* \sim 50$ nm. The increased crossing angle θ between the beams (from 22 mrad to 83 mrad) further decreases their effective overlap region. With the low emittance particles, assured by the RF gun and the DR, a reduction of β_y^* by a factor of 20 is achieved (from $\sim$6 mm to $\sim$0.3 mm). By also increasing the beam currents $I_\pm$ by a factor of 2, one can expect an increase in luminosity by a factor of 30 for SuperKEKB compared to KEKB.

In December 2024, SuperKEKB reached a peak luminosity of $5.1 \times 10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$, setting a new world record for electron-positron colliders [18]. With the current accelerator design, SuperKEKB is expected to reach $2 \times 10^{35}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ [19]. For the targeted luminosity of $6 \times 10^{35}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$, further improvements are necessary, like a redesigned Interaction Region (IR). The final focusing magnets (Quadrupole Cryogenic System, QCS [20]), which are responsible for the squeezing of the beams close to the IP, will be upgraded, along with other improvements to the accelerator and detector. This is expected to take place around 2032, during the Long Shutdown 2 (LS2).

During Run 1 from March 2019–June 2022, Belle II recorded $424\,\mathrm{fb}^{-1}$ of integrated luminosity, of which $363\,\mathrm{fb}^{-1}$ are at the $\Upsilon(4S)$ resonance [21]. This is about half what Belle recorded during its entire runtime of 11 years ($\sim 1\,\mathrm{ab}^{-1}$, $711\,\mathrm{fb}^{-1}$ at $\Upsilon(4S)$). For Run 2, which started in 2024 and is expected to end in 2032, the projected luminosity is $\sim 10\,\mathrm{ab}^{-1}$. The final dataset of Belle II is expected to contain $\sim 50\,\mathrm{ab}^{-1}$ of data [22], which corresponds to approximately 10^{11} recorded B mesons.

1.3 Belle II detector design

The Belle II detector is an upgrade of the Belle detector, carried out with the SuperKEKB accelerator upgrade. Like its predecessor, the Belle II detector features an asymmetric design, adapted to the asymmetric beam energies and the forward boost of the $\Upsilon(4S)$ particle. Most components of the Belle detector were replaced or upgraded. An overview of the Belle II detector is shown in *Figure 1.4*.

The central part of the Belle II detector, closest to the IP, is the Pixel Detector (PXD). To achieve a higher resolution of the vertex reconstruction compared to Belle, which is both necessary because of the lower forward-boost of SuperKEKB, and desirable to improve on the track reconstruction, two DEPFET-based silicon pixel layers were constructed at distances of 14 mm and 22 mm from the beamline, now being much closer than Belle's innermost silicon layer at ~ 30 mm [25]. The PXD is described in detail in *Section 1.4: The Pixel Detector (PXD)*.

Just outside the PXD is the Silicon Vertex Detector (SVD), which is built from double-sided silicon strips. The SVD is described in detail in *Section 1.5: The Silicon Vertex Detector (SVD)*. Both the PXD and SVD subdetectors are responsible for the vertex and track reconstruction, and are also referred to as Vertex Detector (VXD) or *silicon tracker*. They detect charged particles and measure their trajectories.

The Central Drift Chamber (CDC), a large-volume wire chamber containing 14 336 sense wires, surrounds the VXD. It provides momentum measurement, contributes to particle identification via $\mathrm{d}E/\mathrm{d}x$, and supplies trigger information.

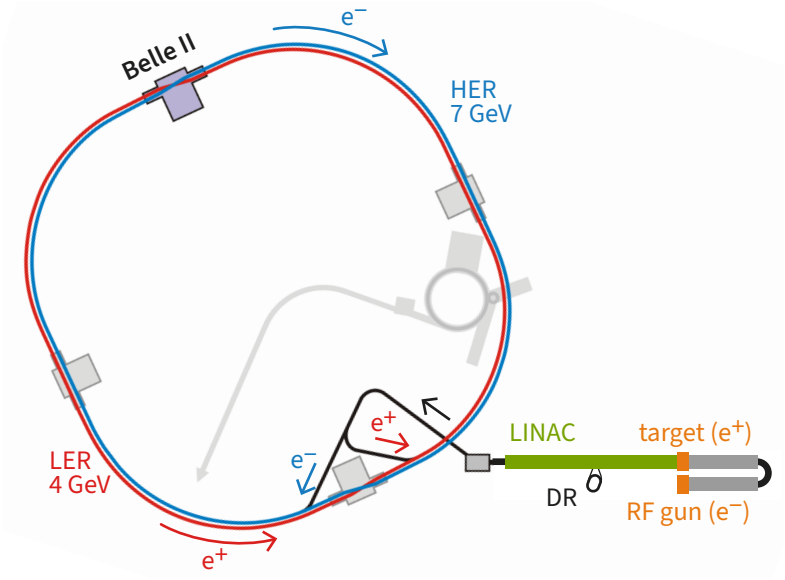


Figure 1.3 – Schematic layout of the SuperKEKB accelerator at KEK. *The previous KEKB accelerator has been upgraded, among others, by adapting the beam energies and adding a Positron Damping Ring (DR). Adapted from [23].*

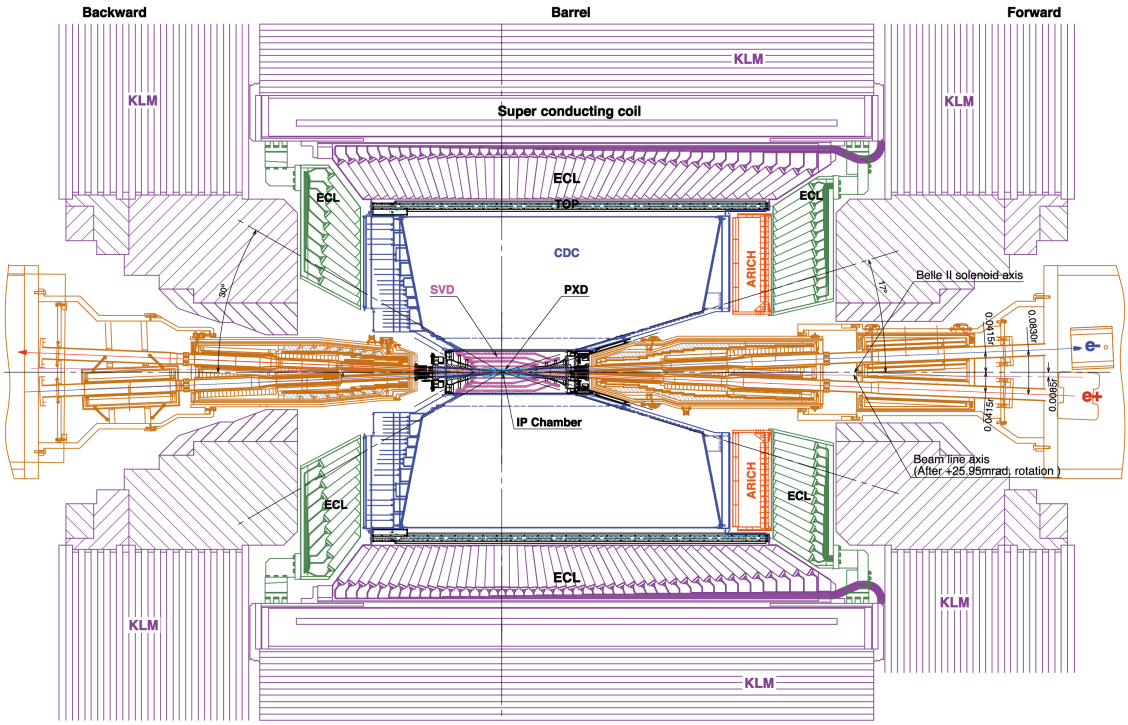


Figure 1.4 – Technical drawing of the Belle II detector. *Shown in a top-down view, the crossing angle between the e^+ and the e^- beams is visible. Most subdetectors are adapted to the asymmetric beam energies. Adapted from [24].*

The CDC is surrounded by the Electromagnetic Calorimeter (ECL), which is built from CsI(Tl) crystals in the barrel region and pure CsI crystals in the forward region. The high density scintillator crystals have a radiation length of $16X_0$, which allows the detection of neutral particles like γ and π^0 from B meson decays and the determination of their energy.

The outermost subdetector is the K_L^0 and μ detector (KLM), outside the superconducting solenoid. It is built of alternating layers of iron plates and sensitive material. It extends the ECL's detection ability to long-lived, high-energy particles, mainly μ and K_L^0. Depending on their energy, K_L^0 mesons create hadronic showers in the ECL or the KLM. μ and non-showering charged hadrons are stopped with electromagnetic interactions, or traverse the KLM entirely.

To discriminate between charged K and π tracks, the Belle II detector features two Particle Identification (PID) systems based on Cherenkov radiation: The Time-Of-Propagation counter (TOP) is a thin layer between the CDC and the ECL. It measures the time of propagation of Cherenkov photons induced by charged particles inside a quartz radiator. In the forward endcap, the Aerogel Ring-Imaging Cherenkov Detector (ARICH) detector is deployed. It is built of a silica aerogel radiator and an array of photon detectors to record the Cherenkov rings.

1.4 The Pixel Detector (PXD)

The Pixel Detector (PXD) is the innermost subdetector of Belle II, and is optimized for precise reconstruction of B-meson decay vertices. It is built with Depleted P-channel Field Effect Transistor (DEPFET) technology, which allows for a sensor thickness of 75 µm [26]. This contributes to a low material budget, corresponding to a radiation length of 0.21 % X_0 [27] per layer. The PXD consists of 40 modules, each containing a 250×768 pixel matrix with sizes varying from 50×55 µm to 50×85 µm.

The PXD sensors are based on a matrix of DEPFET structures [28, 29]. A cross section of a single DEPFET pixel is shown in *Figure 1.5*. Each device is fabricated on a thin silicon bulk that is fully depleted via a reverse-bias voltage. A highly n-doped region is embedded in the substrate, and forms the *internal gate*. At the surface, a field-effect transistor (FET) senses the charge stored in the internal gate and provides amplification. When a charged particle passes through the depleted bulk, it generates electron-hole pairs. While the holes drift to the p+ back contact, the electrons are collected in the internal gate. The stored charge modulates the FET channel current as it is read out. After readout, the electrons are removed from the internal gate by applying a positive pulse to the *clear gate*, resetting the pixel for the next measurement.

The DEPFET pixel matrix is read out in a rolling shutter mode, with four rows processed simultaneously. On each of the 40 modules, the pixel currents are amplified and digitized to 8-bit resolution by four Drain Current Digitizer (DCD) chips. To reduce

this data volume, the output is processed on-module by Data Handling Processor (DHP) ASICs. The DHP performs common-mode correction and zero-suppression, then buffers the resulting data until a hardware trigger requests their transmission. The data are then passed on to the Data Handling Hybrid (DHH), an FPGA-based backend readout system located at the Top-of-Belle (ToB). This area also houses the power supplies necessary for PXD operation. The DHH system is composed of several specialized boards. Five Data Handling Engines (DHEs) receive the data streams from the PXD modules. A separate Data Handling Insulator (DHI) is responsible for the Slow Control and synchronization [30]. Finally, a Data Handling Concentrator (DHC) gathers the data from the DHEs, builds sub-events, and sends them to the Online Selection Node (ONSEN) [31].

The PXD detector itself is composed of two cylindrical layers which are placed at 14 mm and 22 mm from the beamline. Each layer is subdivided radially into module-pairs called ladders. The ladders are tilted with respect to the beam line, to form a windmill structure. This allows to have a full ϕ coverage with the PXD, eliminating gaps that would occur when placing the ladders next to each other. The first layer (L1) is composed of 8 ladders, while the second layer (L2) contains 12 ladders. Each ladder contains a pair of half-modules with a sensor area of 250×768 pixels each, glued together on the short side. The sensors on one ladder are referred to as forward-sensor (along the boost, in $+z$ direction), or backward-sensor. The gaps between the forward- and the backward-sensor are shifted from L1 to L2, so that the PXD covers the full θ detector acceptance of ∼17° forward and ∼150° backward. While each sensor has the same amount of pixels, the sizes of the pixels vary. For each half-module, the 250×256 pixels close to the IP are smaller (50×55 µm for L1, 50×70 µm for L2) than the 250 ×512 pixels in the outer regions, next to the DCD/DHP electronics (50×60 µm for L1, 50×85 µm for L2).

Each of the 40 sensors are identified by the sensor ID, which is composed of [33]:

- **layer.** 1 for the innermost layer L1, 2 for the second layer L2.
- **ladder.** 1–8 for L1, 1–12 for L2. Ladder 1 is defined as intersecting the $+x$-axis, pointing outwards the accelerator ring, with an associated azimuthal angle $\phi = 0$. Increasing ladder numbers are counted anticlockwise.
- **sensor.** The sensor number is counted along the ladder, 1 for forward, 2 for backward.

During Run 1 from March 2019 – June 2022, the PXD had to be operated in a reduced configuration, composed of the full inner layer L1 with 8 ladders, and only two out of 12 ladders in the second layer, covering about 15 % of the azimuthal ϕ detector acceptance. The other sensors were damaged during the ladder assemblies. During the Long Shutdown 1 (LS1) in 2023, a renewed PXD2 was installed, now containing 40 working modules as originally envisaged [27, 34, 35]. While the inner layer L1 is the most important for precise vertexing, the outer layer L2 facilitates track reconstruction for low-p_T tracks and provides fallback capabilities in case of failures in L1.

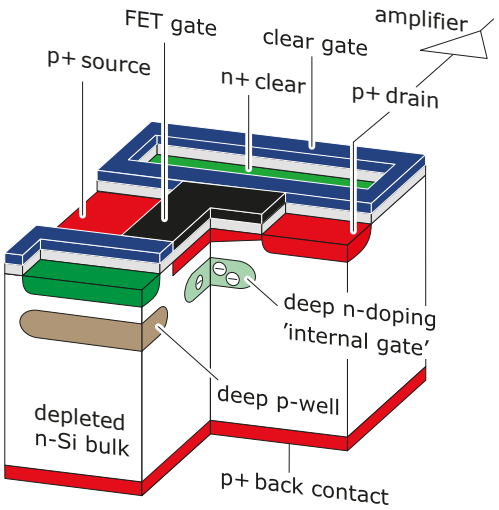


Figure 1.5 – Cross section of a DEPFET sensor. *The DEPFET sensor operates by detecting the charge deposition in the internal gate using an external FET. Image from [32].*

1.5 The Silicon Vertex Detector (SVD)

The Silicon Vertex Detector (SVD) is built as double-sided strip detector (DSSD), where each sensor has silicon strips on the front side and the back side, placed orthogonally to each other. On the front side of the sensor (facing the IP), the longer P-strips are oriented parallel to the beam axis z. On the back side of the sensor (facing outwards), the N-strips are oriented orthogonal to the z-axis.

The DSSD sensors are made of n^--doped silicon, thus the P-type strips can simply be realized by implanting p^+-doped material in the substrate (*junction side*). For the N strips on the other side of the sensor (*ohmic side*), the n^+-doped electrodes need to be isolated from each other, often realized by implanting additional p-doped elements (*p-stops*) between them. The SVD uses atoll-type p-stops to reduce the crosstalk and the noise [36].

The double-sided design allows each sensor to obtain 2D information, by considering the crossing of the active P- and the N-side strips. In case two particles traverse the sensor at the same time, an ambiguity arises: in addition to the two actual intersections of the particle track with the sensors, two additional *ghost* hits are seen, as the 2 P-strips and the 2 N-strips create four crossings (see *Figure 1.6*). Based on the information from a single sensor, these *ghost* hits are indistinguishable from *true* hits. This effect amplifies with increasing number of (simultaneous) tracks, which needs to be considered in the track reconstruction:

$$n_{\text{ghosts}} = n_{\text{tracks}} \cdot (n_{\text{tracks}} - 1) \quad .$$

Overall, the SVD is composed of 172 DSSD sensors in four layers (L3, L4, L5 and L6). Like the PXD, the SVD sensors are arranged in tilted ladders distributed on the azimuthal angle ϕ (L3: 7 ladders, L4: 10 ladders, L5: 12 ladders, L6: 16 ladders). Each ladder contains multiple sensors in z direction (L3: 2 sensors, L4: 3 sensors, L5: 4 sensors, L6: 5 sensors per ladder). To reduce the amount of sensor material, while still covering the full Belle II polar angular acceptance of 17° forward and 150° backward, the forward-most sensors in the outer layers L4, L5 and L6 are tilted towards the interaction point, so that the layers follows a pencil-shape (lantern shape) rather than a cylindrical shape as Belle's SVD. Those sensors have a trapezoidal shape. The naming of the sensors follows the same scheme as the PXD, so sensor *6.10.1* would be on the outermost SVD layer L6, on the 10th ladder out of 16 (roughly in $-x$ direction), and at the forward-most position (using a trapezoidal geometry). The sensors of the SVD are shown in *Figure 1.7*.

The DSSDs are read out using the APV25 [37], a low-noise 128-channel amplifier chip. On the central modules, the *Origami chip-on-sensor scheme* [38] was applied: the APV25 are thinned down to 100 µm and are only mounted on the N-side of the DSSD. A flexible circuit is bent around the short edge of the sensor to reach the P-side strips. This simplifies the cooling of the electronics. A second type of readout chip,

the *hybrid board*, is used on the edges of each ladder, outside the sensitive volume. They read out either the P or the N side.

The 1748 APV25 are connected to 48 junction boards, located close to the CDC. The junction boards figure as patch panel and provide the power supply for the APV25 chips. A copper wire carries the analogue signals to 52 Flash Analog-to-Digital Converters (FADCs). The signals from the junction boards corresponding to the hybrid APV25 of L3 are split and serve 2 FADC boards each, to avoid readout bandwidth limitations of the high-occupancy inner layer, thus the higher FADC count. The FADCs are placed in four crates at ToB, two at the backwards side (with 32 FADCs, where most cables exit the SVD), and two at the forward side (20 FADCs). Each crate features one FADC Controller, which receives the trigger signal and distributes it to the FADCs.

The FADCs digitize the analogue APV25 signals, and apply various data processing steps, like re-ordering strips, common-mode correction and zero-suppression. The FADCs can operate in four modes: raw, transparent, zero-suppressed and hit-time-finding. The data format of the zero-suppressed mode is shown in *Figure 1.8a*. Each FADC is connected to one FINESSE Transmitter Board (FTB), which is based on a Xilinx Spartan-6 XC6SLX100T FPGA. It wraps the FADC data into the FTB data format (*Figure 1.8b*), and sends it via optical fibres to the common readout system and to DATCON. The communication with DATCON is described in *Section 3.1: FTB-DATCON link*.

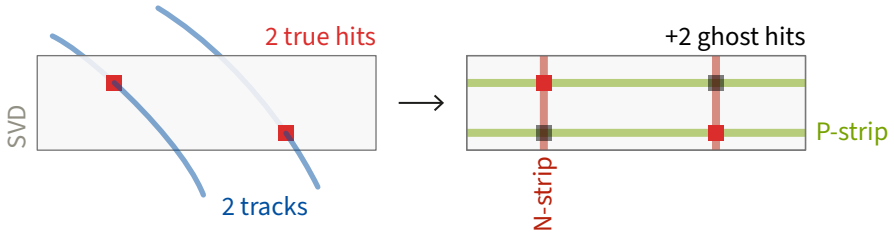


Figure 1.6 – Ghost hits on a double-sided strip sensor. *Double-sided silicon strip detectors (DSSD), like the SVD, read out the hits coordinate by two perpendicular strips, the* P-strips *and* N-strips*, mounted on the front- and backside of the sensor. This creates an ambiguity when multiple particles cross a single sensor, leading to* ghost hits *which are not distinguishable from* true hits *during readout. Additional information is required, e.g. from other layers, to resolve this ambiguity.*

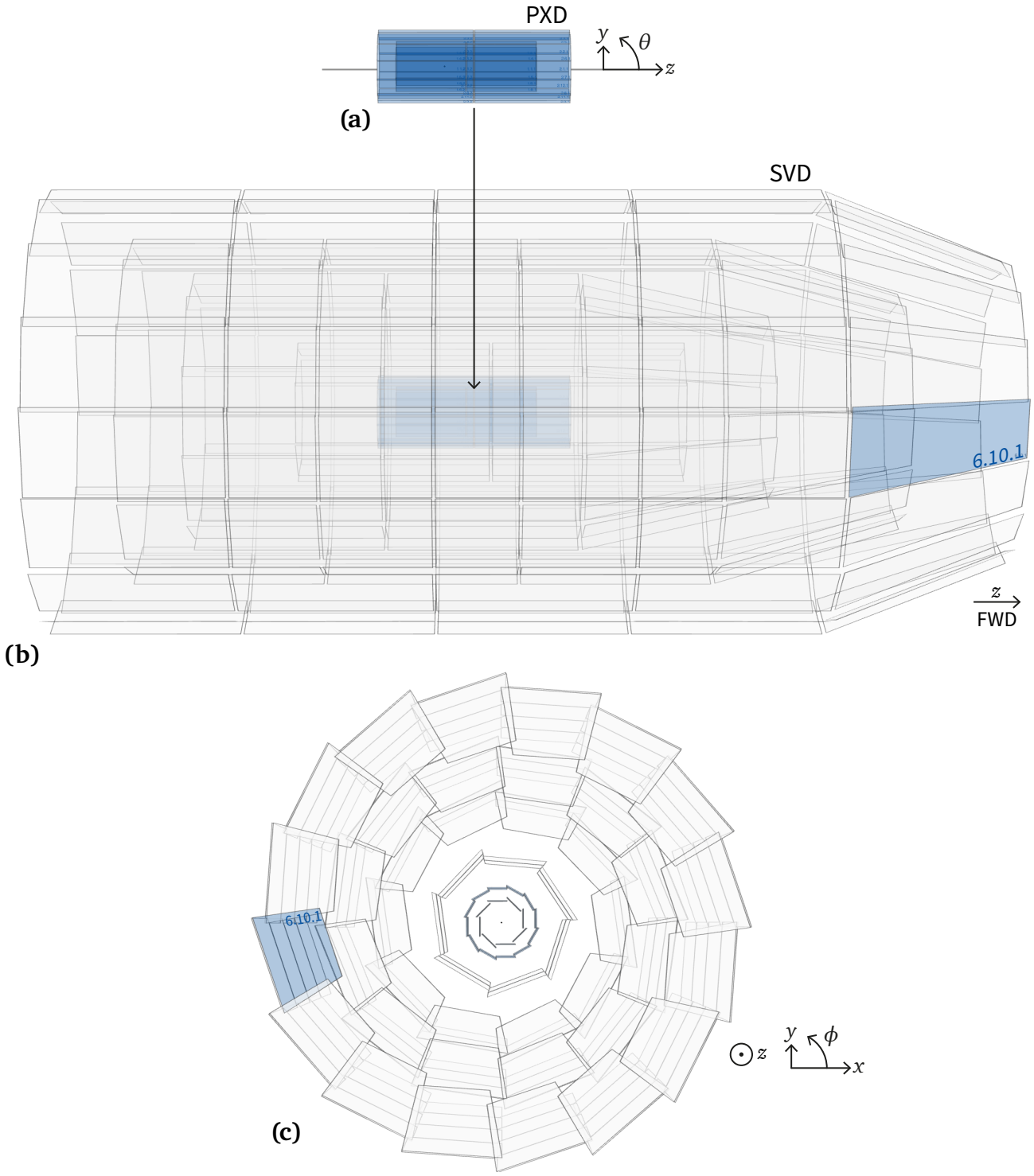


Figure 1.7 – Detector layout of the PXD and SVD subdetectors. **(a)** *shows the active sensor areas of the 40 PXD sensors.* **(b)** *shows the full VXD with the SVD around the PXD. The PXD and the exemplary SVD sensor* ■ 6.10.1 *are highlighted. The slanted trapezoidal forward sensors in SVD layers L4, L5 and L6, forming the pencil-shape of the SVD, are visible on the right.* **(c)** *shows the VXD from the front, in the xy- / $r\phi$-view, the forward direction $+z$ facing the reader, highlighting the windmill structure of the ladders.*

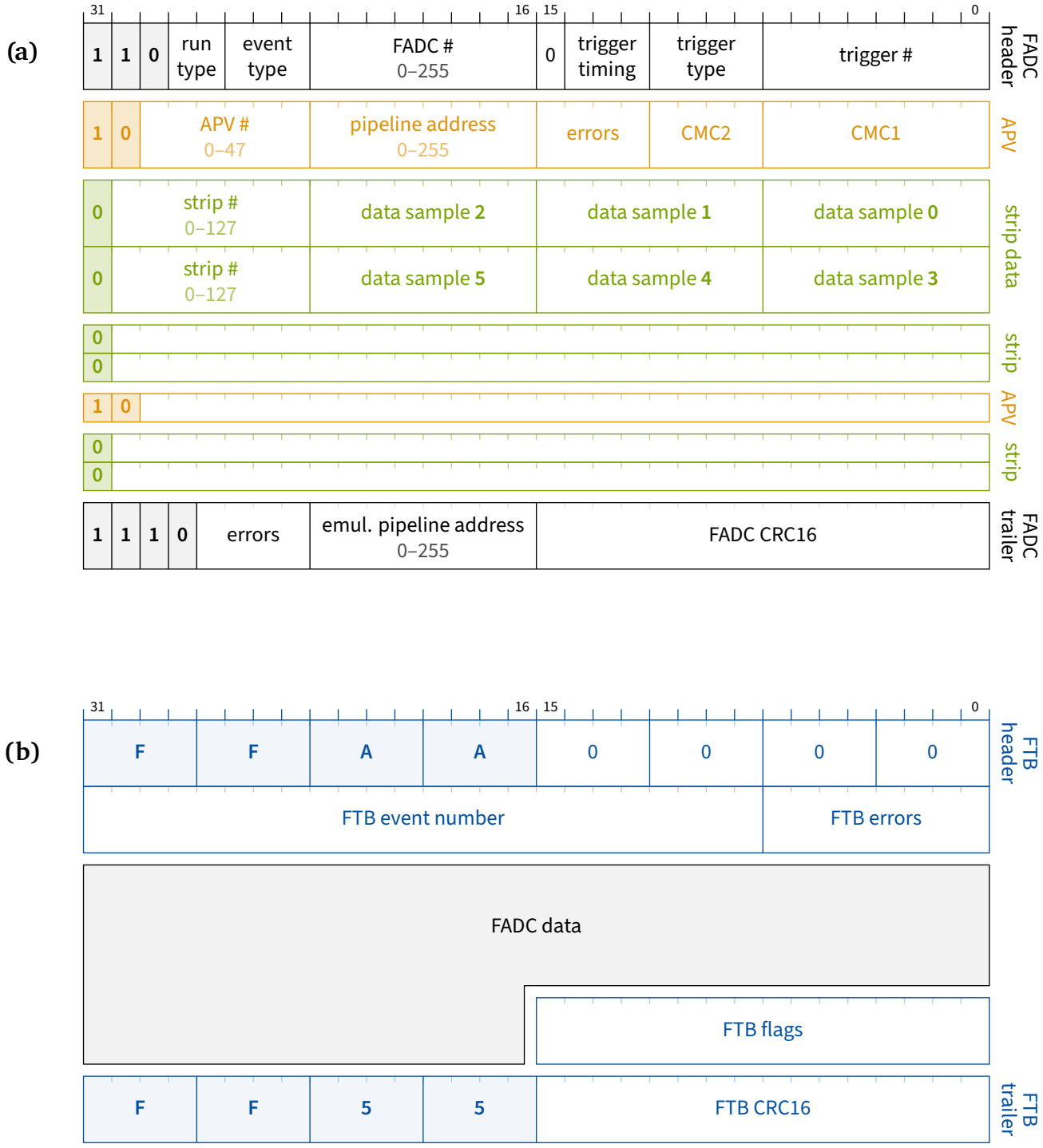


Figure 1.8 – FADC and FTB data format as received by DATCON. **(a)** *shows the FADC data format for zero-suppressed 6-sample mode. It contains data from multiple* ■ APVs *with multiple* ■ strips *each.* **(b)** *shows the FTB data format, which wraps the FADC data.*

1.6 Data acquisition (DAQ)

The Belle II data acquisition (DAQ) system is responsible for the readout of the detector signals, the data processing, and the data storage. An overview of the DAQ chain is shown in *Figure 1.9*.

The first trigger stage, the Level 1 trigger, has a rate of 30 kHz [39]. The trigger signal is distributed to the subdetectors and readout electronics through a tree structure of Frontend Timing Switches (FTSW) modules [40].

The subdetectors of Belle II share a common readout scheme. The front-end electronics, inside or close to the detector, amplify and digitize the analogue signals. They are then transmitted via optical fibres, using the b2link [41] protocol, to the PCIe40 [42] back-end electronics (previously COPPER) in the Electronics Hut (E-Hut). A readout PC (ROPC) then forwards the data via Ethernet to the event builders and the High-Level Trigger (HLT).

The HLT [43, 44] is a computer farm of 20 units, with about 5000 CPU cores [45]. It performs the online event reconstruction using the Belle II Analysis Software Framework (basf2) [46–48], which is also used for the offline reconstruction of the stored data later on. Based on the physics criteria in the reconstructed event, the HLT decides if it is to be stored (triggered) or discarded. The HLT works in a highly parallel fashion, with each node processing one event at a time. The time to reconstruct an event varies depending on the event complexity, but takes on average 0.5 s.

Because of the large data rate of the PXD, it has a special path in the DAQ chain and does not follow the b2link / PCIe40 readout scheme. Instead, the PXD data are sent to the ONSEN, which is designed to buffer the incoming PXD data for up to five seconds. During that time, it receives regions of interest (ROIs) from DATCON and the HLT. After merging them, ONSEN reduces the PXD data to only the pixels inside the ROIs. If the HLT didn't trigger on the event based on the physics, the ONSEN discards the PXD data entirely. The reduced PXD data are then sent to the event builder 2, which merges them with the HLT output of the other subdetectors, to be stored on disk for offline usage.

The creation of ROIs for the PXD is a crucial task, which is why two systems, DATCON and the HLT, were developed to reduce the data stream with distinct approaches. While the HLT generally yields better performance in terms of ROI quality and data reduction due to its access to the full basf2 track reconstruction algorithms as shown in *Section 4.7: Comparison with HLT*, DATCON's FPGA-based architecture can provide ROIs with very low latency. This capability is beneficial for future high-luminosity operations and ensures continuous data processing under various running conditions.

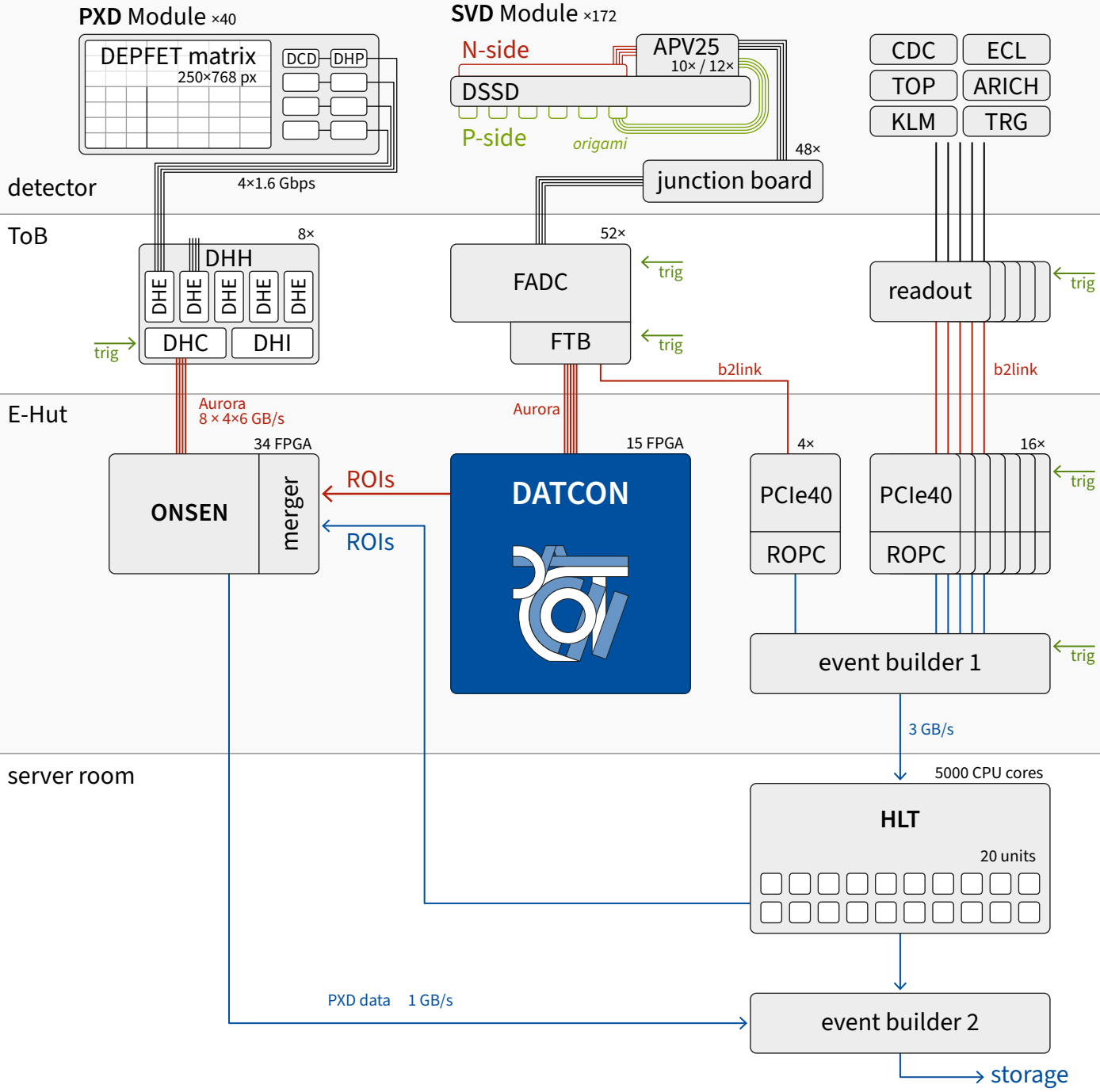


Figure 1.9 – Belle II DAQ chain. *The front-end read out electronics are located directly on the sensors (DCD/DHP, APV25) and on ToB. All subdetectors follow the common b2link protocol to transmit their data to PCIe40-based FPGAs in the E-Hut. The PXD being the exception, its data is sent to ONSEN for buffering. DATCON receives the FTB data and sends its ROIs to ONSEN, where they are merged with the HLT ROIs. ONSEN reduces the PXD data and forwards it to the final event builder.*

1.7 The DATCON hardware

At KEK, the DATCON hardware is located in a 19-inch rack in the E-Hut, next to the Belle II detector. An overview of the rack, with all interconnections, is provided in *Figure 1.10*.

MicroTCA shelves. In the rack, two 8U MicroTCA shelves are installed. Each shelf has 2+9+2 slots. The cards in the slots can communicate electrically over a backplane printed circuit board (PCB), which was custom designed for DATCON. The nine central slots can be used for Advanced Mezzanine Cards (AMCs): The slots 1–3 and 6–8 (resp. 6–9, for the bottom shelf) are equipped with Concentrator AMCs. Slot 4 holds the Tracking AMC, slot 5 is used for the Extension AMC. The two leftmost slots are reserved for a 400W Power Module (PM) and a MicroTCA Carrier Hub (MCH). The MCH manages the power and cooling for the AMC cards. The two rightmost slots are not used, but could host redundant PM and MCH modules. The AMC cards of DATCON are shown in *Figure 1.11*.

The MCH is connected via Ethernet to the DATCON virtual LAN (VLAN) for management via Intelligent Platform Management Interface (IPMI). It is also connected via serial USB connection to the DATCON-control PC.

In each shelf, the FPGAs on the AMCs are daisy-chained with a JTAG connection, so only one JTAG adapter is required per shelf. The two JTAG adapters are connected to the DATCON-control PC for direct communication and flashing of the FPGAs.

Each AMC also has an Ethernet connection to the DATCON VLAN `10.16.5.0/24` in the PXD network, for Slow Control. The IP addresses for the AMC cards are assigned based on the shelf (`.2_` for the upper, N-side shelf, `.1_` for the lower, P-side shelf) and slot number (`._1`–`._9`), so the P-side Tracking AMC is reachable as `10.16.5.14`, while the last Concentrator in the N-side shelf is `10.16.5.28`.

DATCON-control PC The DATCON-control PC is located below the two shelves. It is used to download the latest firmware from the CI/CD servers, and flash them onto the FPGAs via JTAG. Additionally, the DATCON-control PC can be used to run the IOCs for Slow Control during development, and to inspect ChipScope signals from the FPGAs for debugging.

Concentrator AMCs The 13 Concentrator AMCs receive data from the SVD FTBs, preprocess it, and forward it to the Tracking AMCs. Each is equipped with a Xilinx Virtex-5 XC5VLX50T FPGA. The firmware of the Concentrator FPGAs are described in *Section 3.2: Concentrator*.

The 52 SVD FTBs are located at ToB, in the area directly above the Belle II detector. Each FTB has its own optical fibre in one of the 3 multi-fibre MPO cables running ≈50 m through the ceiling of the E-Hut to the optical patch panel in the DATCON rack,

from where they are connected to the 13 Concentrator cards. Each Concentrator has four optical transceivers, receiving signals from four FTBs. As the FADC and APV25 IDs are included in the data stream (*Figure 1.8*), the Concentrators can identify the source of the data without relying on a specific cable mapping of the FTBs to the Concentrator transceivers.

Tracking AMCs The two Tracking AMCs perform the track reconstruction and the ROI finding. They are equipped with a Xilinx Virtex-6 XC6VLX240T FPGA, which is more powerful than the Concentrator FPGAs. The Tracking firmware is described in *Section 3.3: Tracking firmware*.

As the Tracking AMCs don't have SFP cages for optical/Ethernet transceivers, an Extension AMC in slot 5 of the shelf is connected to the Tracking AMC via electrical backplane connection. The Extension AMC does not contain an FPGA, and is only designed for routing the electrical signals from the Tracking FPGAs to additional SFP cages: two optical SFP modules for the communication between the two Tracking AMCs, and one Gigabit Ethernet module for Slow Control. The Tracking AMC in the P-shelf uses an additional optical connection for sending the final ROIs to the ONSEN.

Interconnection Concentrator and Tracking AMCs After the FPGAs on the Concentrator AMCs preprocessed the data, they send their data to the Tracking cards in the respective shelf via electrical backplane connection.

The Tracking firmware on both P and N Tracking AMCs separate the P- and N-strip data, which are then either kept or forwarded to the other, corresponding Tracking card, so all P-side strips will be processed by the P-side Tracking AMC in the bottom shelf, and all N-side strips are processed by the N-side Tracking AMC in the top shelf.

Connection to ONSEN After both Tracking AMCs produced their 1D ROIs, they are collected by the P-side Tracking AMC, where they are merged to per-sensor 2D ROIs and sent to the ONSEN via a single optical link. The ONSEN accepts up to 128 ROIs per event.

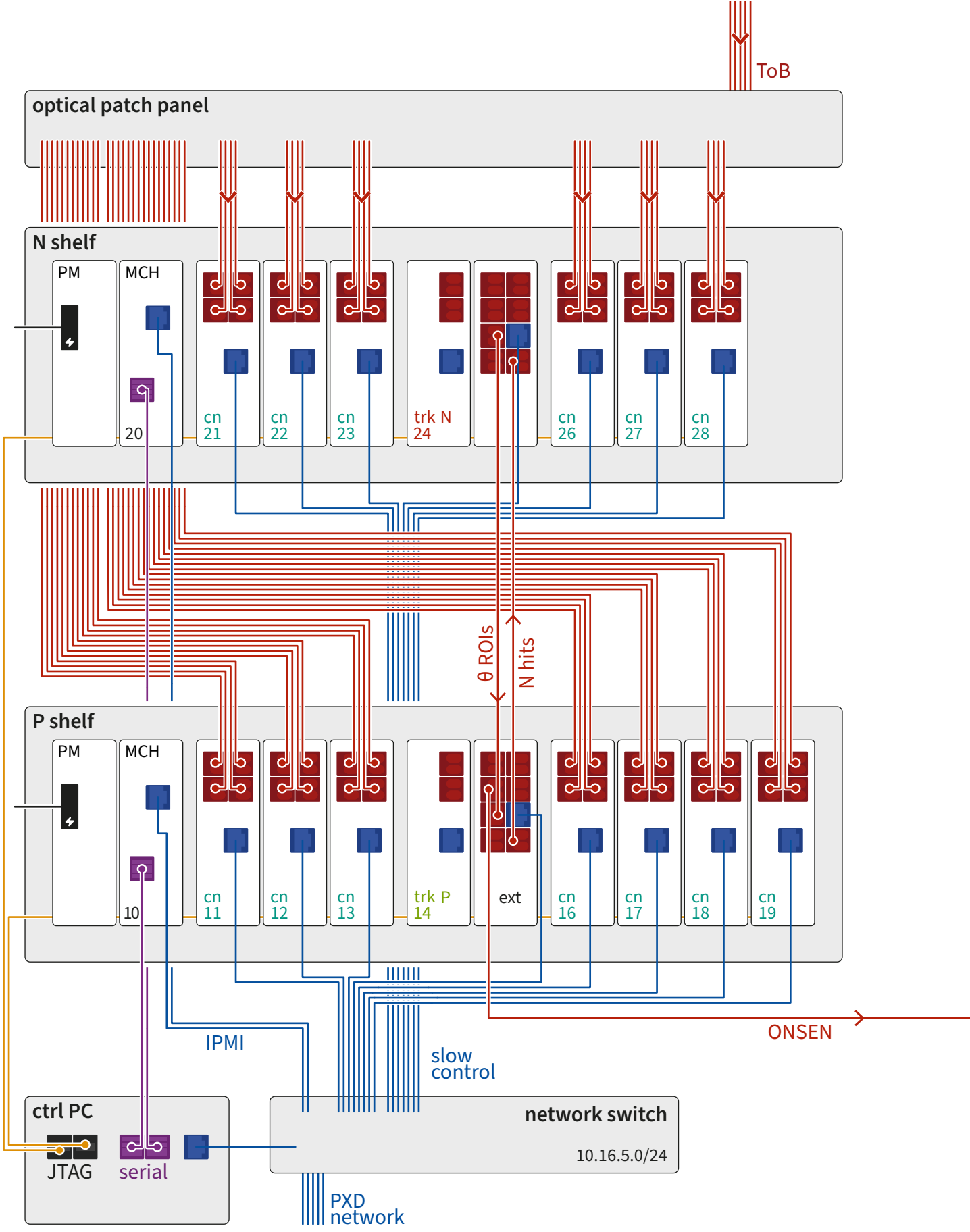


Figure 1.10 – DATCON rack with interconnections. *Optical cables shown in red, Ethernet cables in blue.*

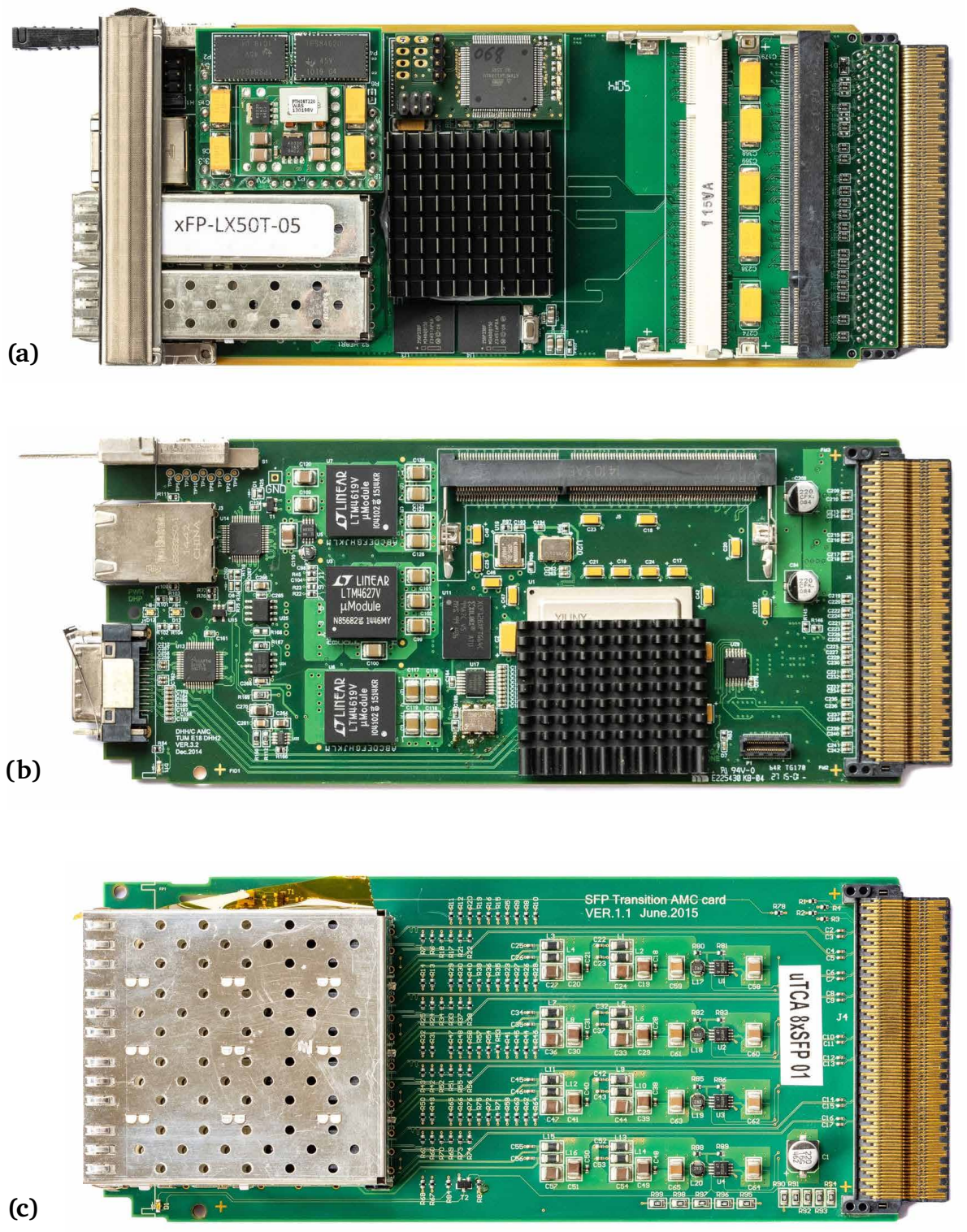


Figure 1.11 – DATCON AMC cards. **(a)** *shows the AMC of the Concentrator, equipped with a Xilinx Virtex-5 FPGA.* **(b)** *shows the Tracking AMC, with the more powerful Xilinx Virtex-6. As the Tracking AMC has no SFP cages, the Extension AMC* **(c)***, connected electrically via the backplane in the shelf, is used for optical communication.*

1.8 FPGAs

Field-programmable gate arrays (FPGAs) are programmable integrated circuits (ICs) that can be reconfigured by the developer after manufacturing. They are similar to application-specific integrated circuits (ASICs) in performance and scope, but they are more flexible, as the internal logic can be changed at any time by loading a different configuration file.

The fundamental building block of an FPGA is the look-up table (LUT), which stores a truth-table of 6 input bits to one output bit. As a LUT is technically a small storage, the contents of the LUTs can be configured to any Boolean function, allowing it to implement common gates like `AND` and `XOR`, but also more complex combinatorial logic. The contents of the LUT, as well as the interconnections between all components, are defined by the bitfile. LUTs are combined with flip-flops (FFs) as storage elements, which enables the FPGA to implement complex digital circuits such as registers, multiplexers, and arithmetic units.

LUTs and FFs are physically grouped together to form a slice, which are arranged in a regular pattern across the silicon. In a Virtex-6 device, for example, a slice contains four LUTs, eight FFs, dedicated multiplexers, and arithmetic carry logic [49]. Two slices are grouped together to form a configurable logic block (CLB). The number of slices and CLBs is specified on each FPGA's data sheet, and provides a measure of its logic capacity.

Additionally, FPGAs also include special-purpose components, like Digital Signal Processing (DSP) slices for efficient arithmetic operations, or block RAM (BRAM) for memories and storage of precomputed data. FPGAs also feature configurable input/output blocks, which can be programmed to interface with optical transceivers or Ethernet links.

FPGAs are typically mounted on PCBs, which provide the necessary support circuitry for integration into the target environment. This includes oscillators for clock generation, connectors for optical fibres or other data links, status LEDs, and additional memory chips. For DATCON, the PCBs conform to the AMC specification.

The firmware for an FPGA is written in a hardware description language (HDL) such as Verilog or VHDL. The source code is organized into self-contained modules with defined input and output signals. Those modules can be nested and interconnected using wires to form the required data flow. In addition to user-written modules, pre-verified IP cores can be used as building blocks. Those are often provided by the FPGA vendor for common functionality like memories, FIFOs, arithmetic units and high-speed transceiver interfaces (like Aurora [50]). Other IP cores can be licensed from third-party vendors, or obtained freely (like IPbus [51]).

After the HDL fully specifies the design, the following steps are necessary:

- **synthesis.** The synthesis tool translates the HDL code to a list of logic elements (LUTs, flip-flops, memory blocks) and their interconnections. It applies optimizations to minimize resource utilization like removing unused logic (dead code elimination), and by inferring device-specific primitives like BRAM and DSPs instead of generic LUTs.
- **implementation (map, place & route).** Mapping packs the synthesized logic into slices. It further optimizes the resource utilization by merging LUTs, deduplicating registers and folding constants. Placement determines the exact tile coordinates for the packed slices on the FPGA, optimizing for timing constraints. Routing connects the placed components.
- **bitstream/bitfile generation.** The design is then converted to a binary file, native to the FPGA model.
- **configuration/flashing.** The bitstream is loaded onto the FPGA, typically via a JTAG interface. The FPGA is now functioning exactly as specified by the original HDL design.

Testing and verification is typically done at the module level using a simulation (testbench). Various stimuli are applied to the inputs, and the outputs are checked for expected values. Traditionally, the test benches are written in the same HDL as the design, but modern tools like cocotb [52] allow writing testbenches in Python. By interfacing with a Simulator like Mentor Graphics (now Siemens) Questa Sim, cocotb can apply input vectors, observe signal changes, and assert expected outcomes, while at the same time interface with Python's data-processing and reporting tools like numpy/pandas and pytest.

DATCON's codebase was extended with an extensive test suite using cocotb, which is automatically run in a CI/CD pipeline to prevent regressions.

Chapter 2

Track reconstruction

The link between the spatial coordinates (hits) in the detector subcomponents and the trajectory (track) a particle followed while propagating through the detector is made by the **track reconstruction**, or **tracking**. The track reconstruction can be divided into two parts:

- **track finding.** The job of the track finding is to group together hits which belong to the same track.
- **track fitting.** The track fitting takes a set of hits from the track finding, and determines the track parameters that best describe the particle's trajectory. For helicoidal trajectories in a uniform magnetic field, there are five track parameters: the curvature q/p, the azimuthal and polar angles ϕ and θ, and the transverse d_0 and longitudinal z_0 distances of the point of closest approach (POCA) to the origin.

Numerous algorithms exist for both track finding and track fitting, each with specific advantages and performance characteristics.

Track reconstruction can be performed either *offline*, after data have been fully recorded, or *online*, during data acquisition. While offline reconstruction can leverage significant computing resources to achieve the highest possible precision for physics analysis, online tracking must operate under strict time constraints to keep up with the detector's data rates. Its primary purpose is to quickly decide what data are worth keeping, reducing the volume of data written to permanent storage. For the PXD detector, which is highly susceptible to background noise as it is close to the beam line, the data must be filtered in near real-time.

While the High-Level Trigger (HLT) reconstructs tracks by applying a Kálmán filter [53, 54] to hits from both the Central Drift Chamber (CDC) and Silicon Vertex Detector (SVD), DATCON employs a Hough-transformation-based method using only SVD hits, as illustrated in *Figure 2.1*.

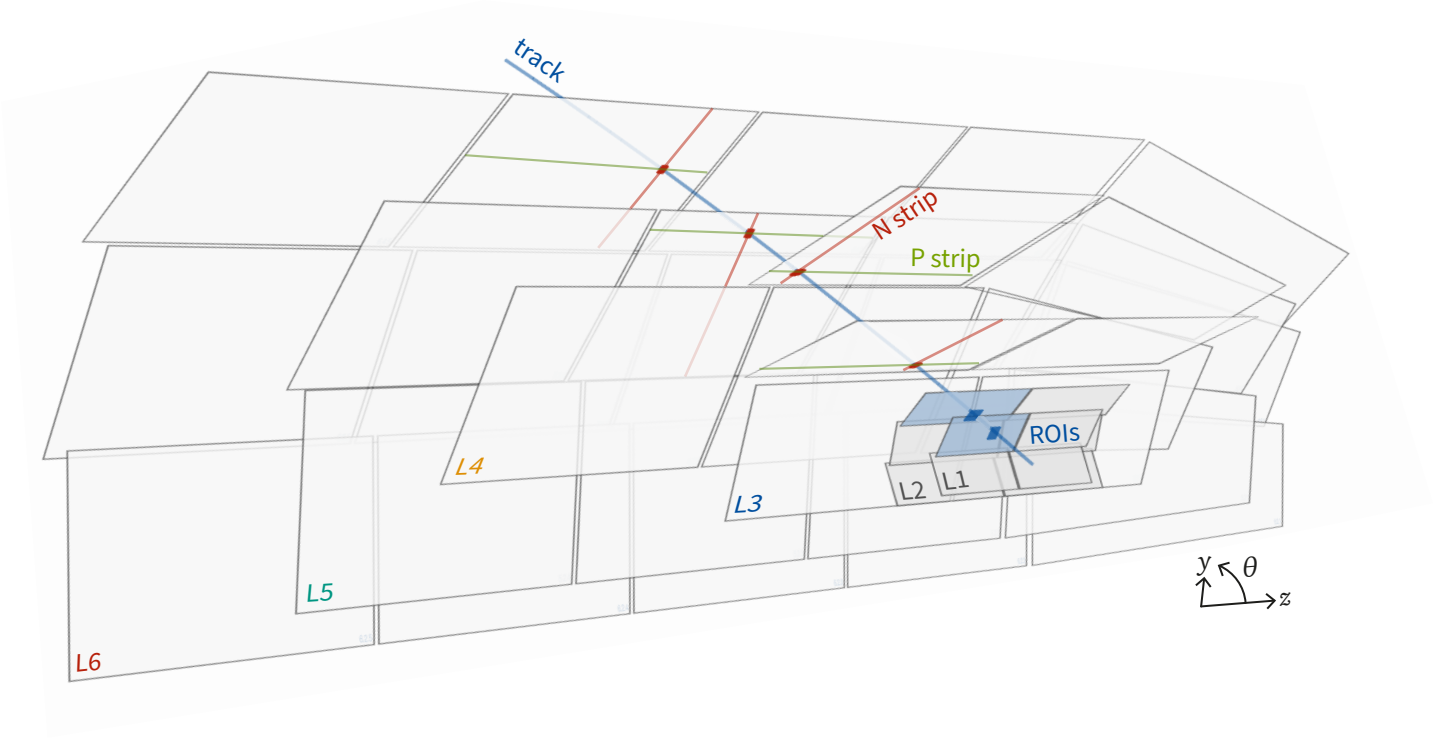


Figure 2.1 – Track reconstruction and ROI creation in the SVD and PXD detectors. *A particle leaves* ■ hits *in the SVD, their corresponding* ▬ P-strips *and* ▮ N-strips *are recorded and passed to DATCON. That strip information is then used to reconstruct the* ▬ track *and extrapolate it towards the IP. Around the intersections of the extrapolated tracks with the PXD layers L1 and L2, a* ■ ROI *of 80×80 pixels is created.*

2.1 Track reconstruction with Hough transformation

The Hough transformation method can be used to find a common line *(track)* through a set of points *(hits)*. Because of its computational efficiency, a Hough transformation based track reconstruction was chosen for DATCON.

The method applies a transformation to the hit coordinates that replaces the problem of finding a connecting line with that of finding the intersection of a set of curves. This transformation maps points from the *figure space* (x, y) into a *parameter space*.

In general, a straight line can be described by the slope m and the intercept c:

$$y = m\,x + c \quad ,$$

as originally proposed by Hough [55]. For lines parallel to the y-axis, this parameterization would pose problems, as m needs to become infinite. Also, the parameter c could take infinitely large values. An improved parameterization proposed by Duda and Hart [56] uses the Hesse Normal Form to define a line by the angle θ_0 of its normal and the distance ρ_0 from the origin. The line can then be expressed as

$$x\cos\theta_0 + y\sin\theta_0 = \rho_0 \quad .$$

The angle θ_0 is usually restricted to the interval $[0, \pi)$, which makes the transformation unique. Each line in the xy figure space corresponds to exactly one point in the $\theta\rho$-plane, as illustrated in *Figure 2.2a*. The $\theta\rho$-plane is called Hough space (HS).

The same idea can now also be used to describe points (x_i, y_i) in the figure space instead of lines, which corresponds to the track finding problem. Applying the same transformation, the points will become sinusoidal curves in the HS:

$$\rho = x_i\cos\theta + y_i\sin\theta \quad .$$

If now the points (x_i, y_i) are collinear in the figure space, their corresponding sinusoidal curves in the HS intersect at the single point (θ_0, ρ_0). To find the track which passes through all the points (x_i, y_i), it is thus sufficient to apply the Hough transformation to obtain a set of sinusoidal curves in the HS, and to search for their intersections. An intersection point (θ_0, ρ_0) describes the parameters of the original line, as shown in *Figure 2.2b*.

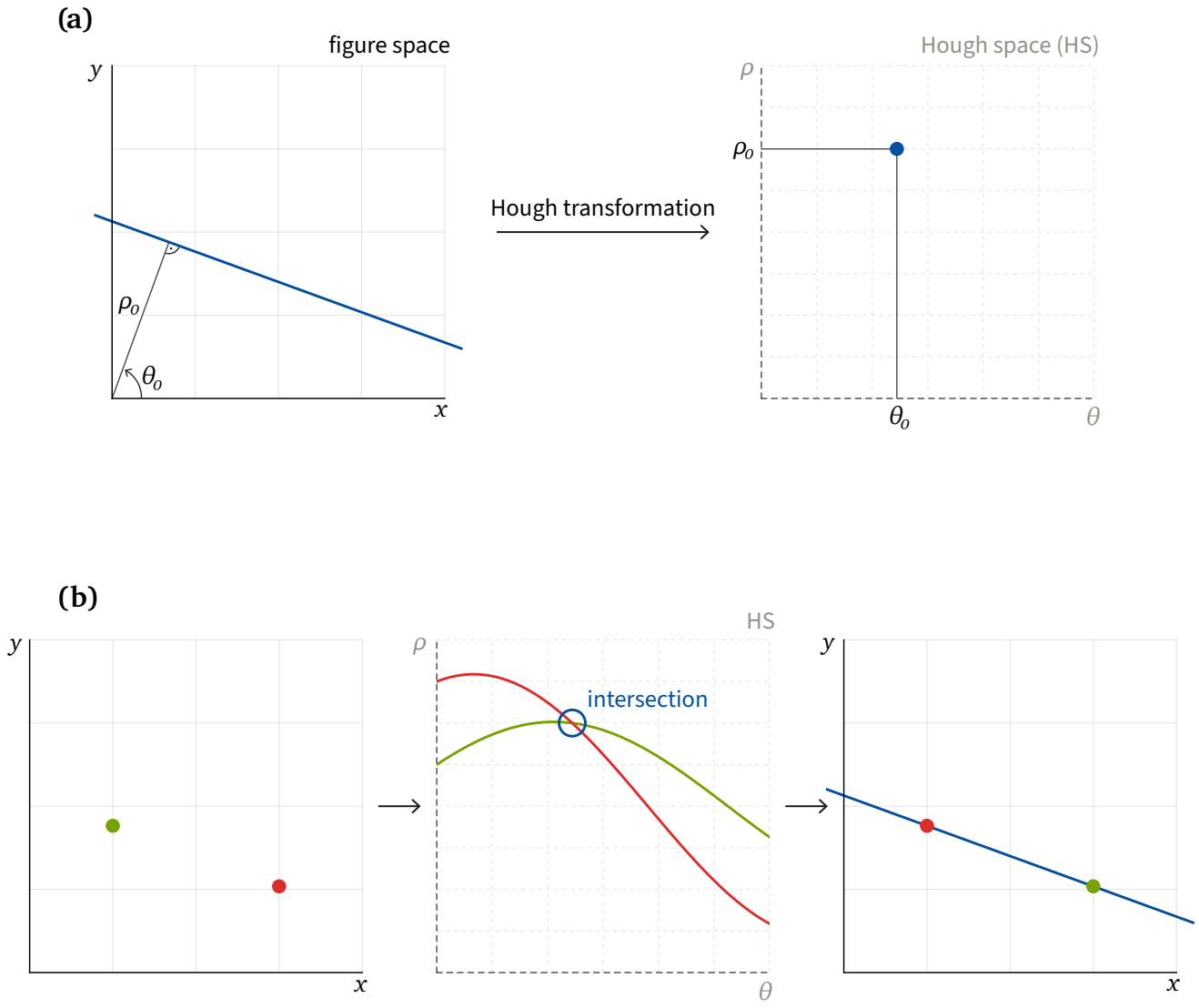


Figure 2.2 – Hough transformation. *In* **(a)** *the Hough transformation is applied to a line in the figure space. In* **(b)***, the Hough transformation is used to find a common line through two points. The two sinusoidal curves — and — in the HS each describe all possible lines through their respective point in figure space. The ○ intersection of the two sinusoidal curves corresponds to the parameters of the target line.*

2.2 Track reconstruction from SVD strips

The SVD data are already separated into P-strips and N-strips when received by DATCON. The track reconstruction process is thus split into two independent 2D track reconstructions, called *projections*.

The P-strips, parallel to z, measure the azimuthal angle ϕ. The N-strips measure the z coordinate, from which the polar angle θ is derived. The two projections are therefore called the φ projection and the θ projection.

The track reconstruction for each projection follows these steps:

- **detector readout.** Obtain the sensor ID and strip number from the SVD.
- **coordinate lookup.** Calculate the 2D coordinates in the corresponding projection (figure space), based on the 3D position of the SVD sensors and strips.
- **transform.** Apply the Hough transformation to the hit coordinates to obtain sinusoidal curves in the HS.
- **intersections.** Find intersections in the HS.
- **track parameters.** Transform the intersection points back to the figure space to obtain the track parameters.

After both projections have been processed, the track candidates from both projections are usually combined to form a full 3D track. In the case of DATCON, however, the 3D track parameters are not required, but only in the intersections of the tracks with the PXD layers to create ROIs. Therefore, these additional steps are performed in each projection:

- **extrapolation.** Extrapolate the 2D track onto the PXD layers to find the intersection points.
- **ROI creation.** Create 1D ROIs around the intersection points.

DATCON then combines the 1D ROIs (intervals) from both projections to form 2D ROIs (rectangles) for the PXD sensors.

To avoid ambiguity, the notation in this thesis distinguishes between three sets of variables: the HS coordinates θ_{HS} and ρ_{HS}, the track parameters ϕ_{track}, θ_{track} and r, and the projections φ and θ.

Conformal mapping If charged particles propagate through a magnetic field, as in the Belle II detector, their trajectory is not a straight line, but a helix. In the φ projection ($r\phi$ plane), the helix maps to a circle. By applying a conformal mapping, the circle can be transformed to a straight line with

$$x_{\text{conf}} = \frac{x}{x^2 + y^2} \equiv x_i \qquad y_{\text{conf}} = \frac{y}{x^2 + y^2} \equiv y_i \quad .$$

After track finding, the radius r and the azimuth ϕ_{track} of the fitted track are obtained from the HS as

$$r = \frac{1}{2\rho_{\text{HS}}} \qquad \phi = \theta_{\text{HS}} - \frac{\pi}{2} \quad .$$

***z* approximation** For the θ projection, the particle trajectory is approximated by a straight line, so no conformal mapping is required for the handling of the N-strips. The z and r coordinates can directly be used for the Hough transformation:

$$z \equiv x_i \qquad r \equiv y_i \quad .$$

The HS then encodes the polar track angle θ_{track}:

$$\theta_{\text{track}} = \theta_{\text{HS}} - \frac{\pi}{2}$$

The ρ_{HS}-axis of the HS corresponds to the line's normal distance to the origin, which is not used for DATCON.

Example. *Figure 2.3a* shows the 3D rendering of one simulated ϒ(4S) event in the PXD (blue) and SVD (gray). As this is a simulation event, the true trajectories of the particles are known – those are shown in — blue and numbered ① to ⑨. Three trajectories do not leave a signature in the PXD: the first one, in the very forward direction, has a low zenith angle θ which is outside the detector acceptance. The other two originate from secondary particles created in the beam pipe.

After having read out the N-strips, DATCON has no information about the ϕ coordinate of the hits. The event thus appears projected onto the zr-plane (θ projections), as shown in *Figure 2.3b*.

From the P-strips, DATCON obtains the xy-plane projection (φ) of the event, shown in *Figure 2.4*.

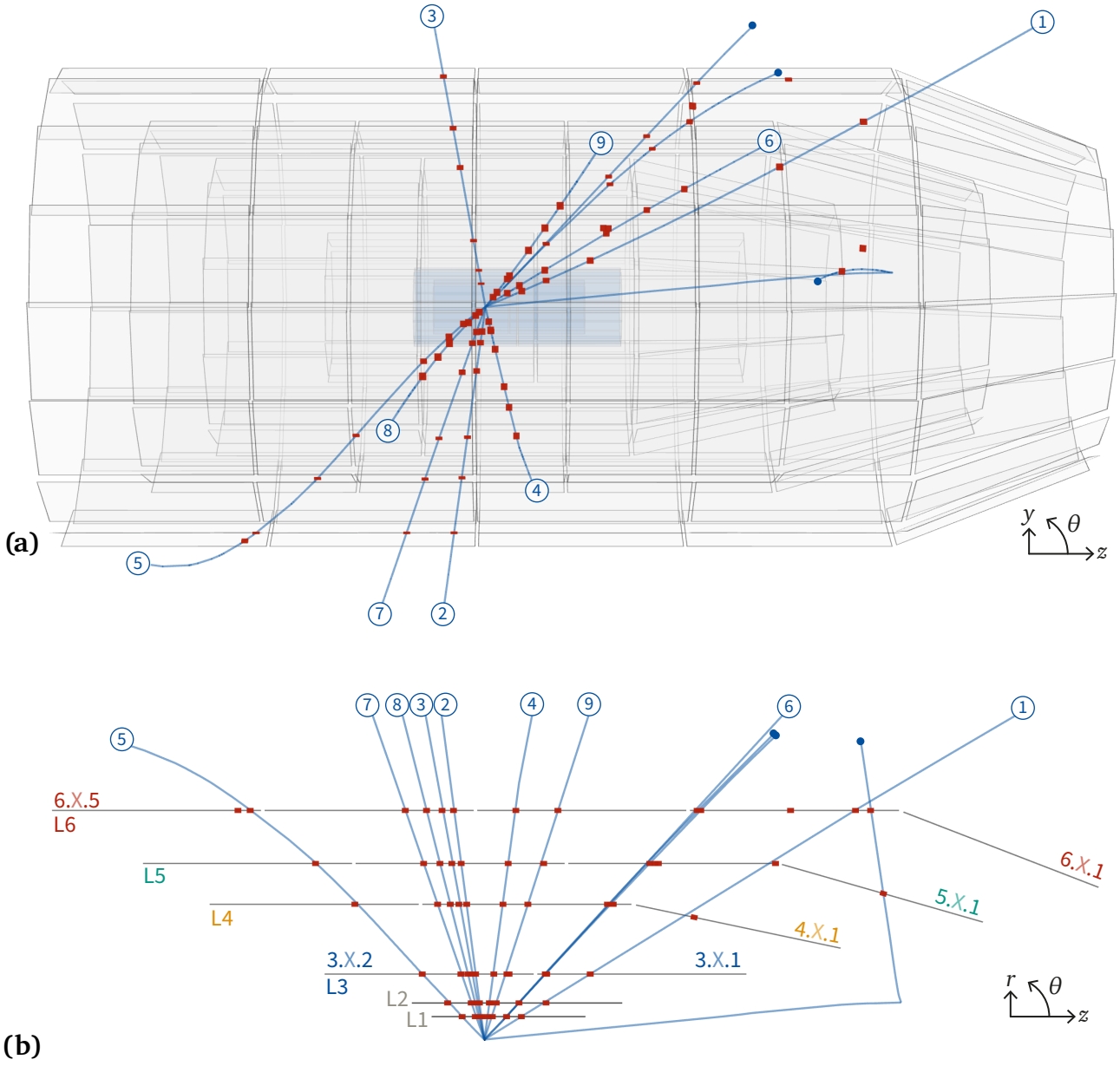


Figure 2.3 – Example event in DATCON's θ projection. *In* **(a)***, the full PXD and SVD detectors are shown in the zy-plane. 9 particle tracks are considered, numbered with their simulation-internal track number. Three tracks are not numbered, as they did not leave a signature in the PXD and thus no further evaluation is possible. The true hits in the PXD and SVD are marked in* ■ red*.* **(b)** *shows the θ projection (zr-plane) of the same event, as seen by DATCON after decoding the SVD N-strip information. The tracks* (7) (8) (3) *and* (2) *have similar polar angles θ. The resulting HS is shown in Figure 4.1.*

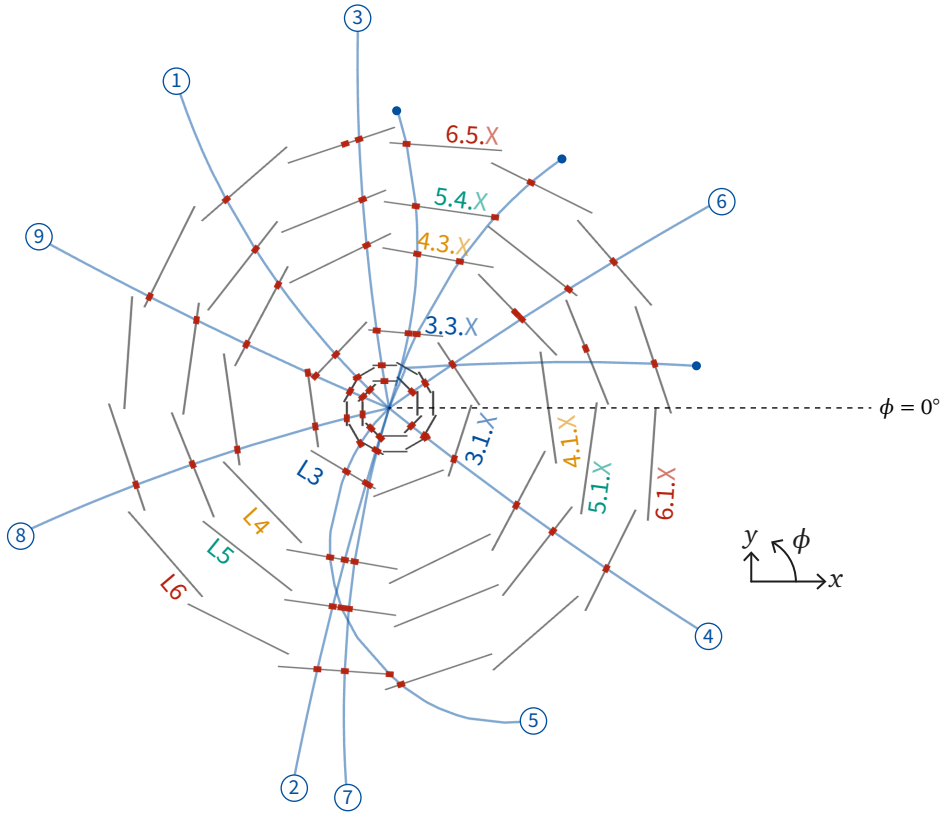


Figure 2.4 – DATCON's φ projection. *The event from Figure 2.3 is shown in DATCON's φ projection (x y-plane). The P-strip data received from the SVD sensors do not contain information about the z coordinate. The curvature of the tracks, related to the momentum of the particle, needs to be considered as well for a successful track reconstruction on the φ side. Track (5) has a very low momentum of $p_T = 40\,MeV$, resulting in a high curvature. Additional ■ hits are visible on sensor 4.2.X, along track (6), which do not belong to the track. The corresponding HS is shown in Figure 4.2.*

Chapter 3

DATCON firmware

The DATCON system runs on 15 interconnected FPGAs, which are mounted on AMC carrier boards. 13 cards, equipped with Xilinx Virtex-5 FPGAs, are running the **Concentrator** firmware. Each Concentrator receives SVD data via optical connections from four FTB readout boards, and decodes and preprocesses their signals. The **Tracking** firmware runs on two Virtex-6 FPGAs. They receive the P- and N-side signals from the Concentrators, perform the track reconstruction using Hough transformations (as outlined in *Chapter 2: Track reconstruction*) and construct the PXD ROIs, which will be sent to ONSEN.

In the following, a chronological description of the data flow through DATCON is given. Starting with the data input, *Section 3.1: FTB-DATCON link* describes the communication between the SVD FTBs and the Concentrators. The two main firmware modules of DATCON are described in *Section 3.2: Concentrator* and *Section 3.3: Tracking firmware*.

A special focus is put on the segmentation in *Section 3.4: Segment veto*, which is this thesis' major contribution to the DATCON system.

Section 3.5: DATCON-ONSEN link describes the communication between DATCON and ONSEN, and *Section 3.6: DATCON-mini* describes the DATCON firmware setup used for simulations and development.

3.1 FTB-DATCON link

The communication between the 52 FTBs and the 13 DATCON Concentrators happens over optical links, using a 16-bit 8b/10b Aurora protocol.

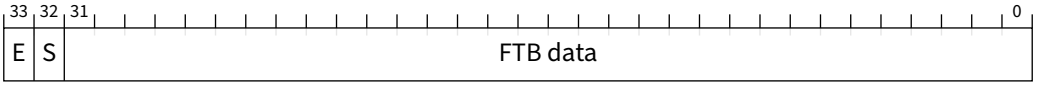
The FTB Firmware runs on Xilinx Spartan-6 (XC6SLX100T) FPGAs. The internal data format of the FTBs is organized in 32-bit words, according to the FTB specifications shown in *Figure 1.8*. The DATCON Concentrator firmware runs on Xilinx Virtex-5 (XC5VLX50T) FPGAs, which feature 12 GTP RocketIO Transceivers [57]. The Aurora lane width of those GTP Transceivers is limited to 2 bytes [50], so the FTB data cannot be sent as-is to the Concentrator boards. A 16-bit instead of a 32-bit communication needs to be used.

In the previous implementation of the FTB-DATCON communication [1], the part of the FTB firmware which is responsible for the data transfer to the Concentrator boards, used a complex finite-state machine (FSM) to parse the FTB frames, wrap them with

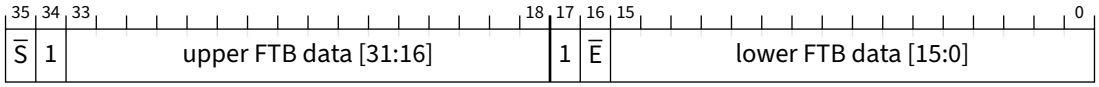
an additional checksum, and split them to a 16-bit word length to be transferred over the Aurora link.

Observations of the Slow Control revealed that the Concentrator modules often received corrupted data. For the current implementation, all DATCON related logic on the FTB side was removed, and replaced by a combinatorial implementation:

In the FTB firmware, the DATCON transmission module obtains the FTB data from a 34-bit FIFO, which contains the 32-bit FTB words and 1-bit start- and end-of-event flags:



Those 34 bits are padded to 36 bits, so that each 16-bit half-word has its own start- and end-of-event flags. For the upper half-word, the global start-of-event flag is re-used, as it will be sent first, and its end-of-event flag is set to *false*. For the lower half-word, the end-of-event flag is used, the start-of-event always being *false*:



A 36-to-18-bit asymmetric FIFO, implemented on the block RAM (BRAM) blocks on the FPGA, splits the 36-bit words into 18-bit words, which can be directly handed over to the 16-bit Aurora module. The start- and end-of-event flags are used as start-of-frame **sof** and end-of-frame **eof** signals for Aurora. A schematic overview of the signals is shown in *Figure 3.1*.

On reassembly, the receiver on the DATCON Concentrator side uses a 16-to-32-bit FIFO to reconstruct the original 32-bit FTB data. By verifying Aurora's start-of-frame **sof** signal, it can be assured that the FIFO only starts accepting data when the first 16-bit segment maps to the most-significant half-word. This prevents swapped-half misalignment of the 32-bit FTB data stream.

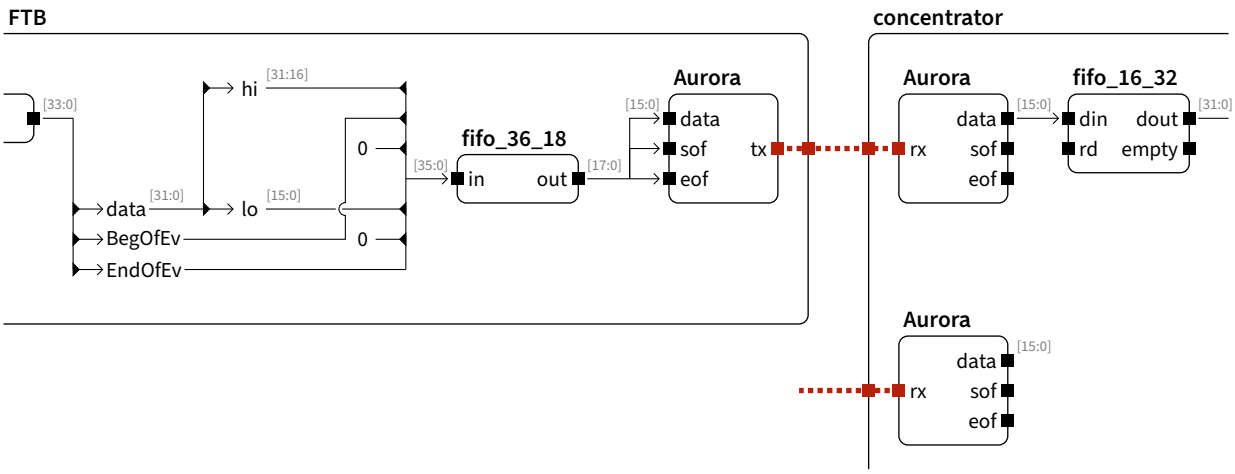


Figure 3.1 – Block diagram of the FTB-Concentrator communication. *The optical transfer happens over a 16-bit Aurora protocol, so the 32-bit FTB frames need to be split on the FTB side and reconstructed on the Concentrator side.*

3.2 Concentrator

After having reconstructed the 32-bit FTB data stream, the Concentrator's main task is to extract the P- and N-side strip information and to apply a filtering to the hits.

The schematic overview of the Concentrator module, with a focus on the event data management, is shown in *Figure 3.2*, and explained in the following.

3.2.1 Event management

The SVD reads out both the P- and the N-side strip information using APV25 readout chips on a single Origami module. Additionally, up to 48 APV25 chips are handled by one FADC/FTB pair. This means that over each optical fibre from one FTB module to DATCON, the data of multiple SVD sensors, and both sides is contained. Moreover, as each DATCON Concentrator board handles optical connections from four FTBs, a significant amount of work is needed by the Concentrators to separate the P and the N side information from inside the streams, and to ensure the temporal ordering of the data, so that each event is complete before it is passed on to the Tracking modules. Most of the work from [2], concerning the Concentrator strip and event management, has been left unchanged.

3.2.2 Preprocessor

The preprocessor module receives the 32-bit FTB data that have been reconstructed by the fifo_16_32. It is responsible for parsing the FTB data stream and preparing it for the Tracking modules. It consists of two main submodules:

- **svd_decoder.** The svd_decoder module consists of a large FSM which decodes the FTB data stream, as defined by the FTB data format in *Figure 1.8*. After obtaining the temporal samples of a hit, an optional filtering can be applied to reduce the noise. The most straightforward method is the *peak noise filter*, which applies a fixed threshold to the peak sample of the hit. A more sophisticated approach, the *noise map* filter, applies a per-sensor noise threshold based on known noise levels in the SVD.
- **svd_clusterizer.** The svd_clusterizer translates the FADC number (0–255), the APV channel number (0–47) and the per-channel strip number (0–127) to the layer/ladder/sensor ID, the side (P or N) and the per-sensor strip ID (0–767). This mapping is defined by the SVD wiring, and documented by the SVD group. Additionally, the svd_clusterizer merges adjacent strips for each sensor to one single hit, as it is expected for one particle to trigger clusters of SVD strips.

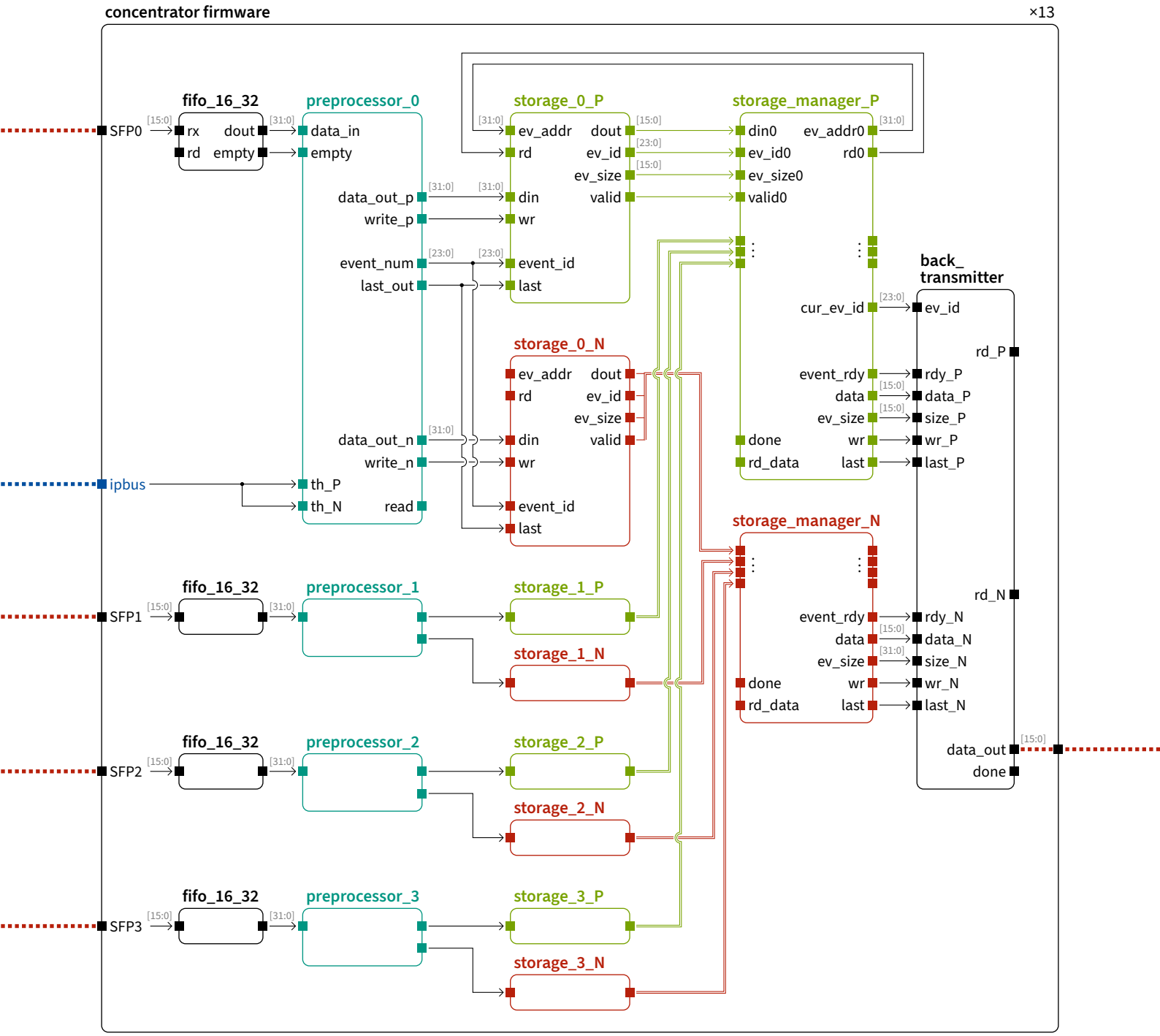


Figure 3.2 – Block diagram of the Concentrator firmware. *The* concentrator *receives its input from the SVD FTBs via Aurora signals through the four optical SFP modules* SFP0-SFP3*. Each has an assigned* preprocessor*, which parses the FTB data stream, does the SVD coordinate lookup and applies filters to the hits. Based on the side of the hit (*P *or* N*), the data will be buffered in one of two storages per* preprocessor*. The* storage_managers *ensure that the hit information from all storages is passed event-wise to the* back_transmitter*, which sends both the P- and N-strip information to the Tracking module via electrical backplane connection.*

3.3 Tracking firmware

DATCON uses two Tracking FPGA cards, one in each shelf. The firmware of the two FPGAs is largely identical and follows the following phases:

- **back_receiver.** The FPGA receives the preprocessed hits from the 6 (N) or 7 (P) Concentrators via electrical backplane connection. The collected P- and N-hits are stored and organized similar to the Concentrator's event handling.
- **crate_transmitter/crate_receiver.** The two FPGAs exchange the P- and N-side hits, so that the FPGA in the upper shelf only has to handle N-strips, and the FPGA in the lower shelf only the P-strips.
- **P_side / N_side.** The main track reconstruction logic, which contains the Hough space (HS) building (hs_phi/theta), the cluster finder (clusterizer_phi/theta), and 1D ROI creation (roi_phi/theta) submodules.

Then, the N-side FPGA sends its 1D θ ROIs to the P-side FPGA. The P-side FPGA then continues with:

- **roi_combine.** The 1D ROIs from both sides are grouped by sensor and combined to 2D ROIs.
- **roi_send.** The ROIs are packed into frames and transmitted to ONSEN.

The following sections detail the P/N_side and roi_combine modules, which are of particular importance as they contain the necessary updates for the segment veto discussed in *Section 3.4: Segment veto*.

Figure 3.3 shows a schematic overview of the process in the P_side module, from the SVD hits to the PXD ROIs. *Figure 3.4* the corresponding block diagram.

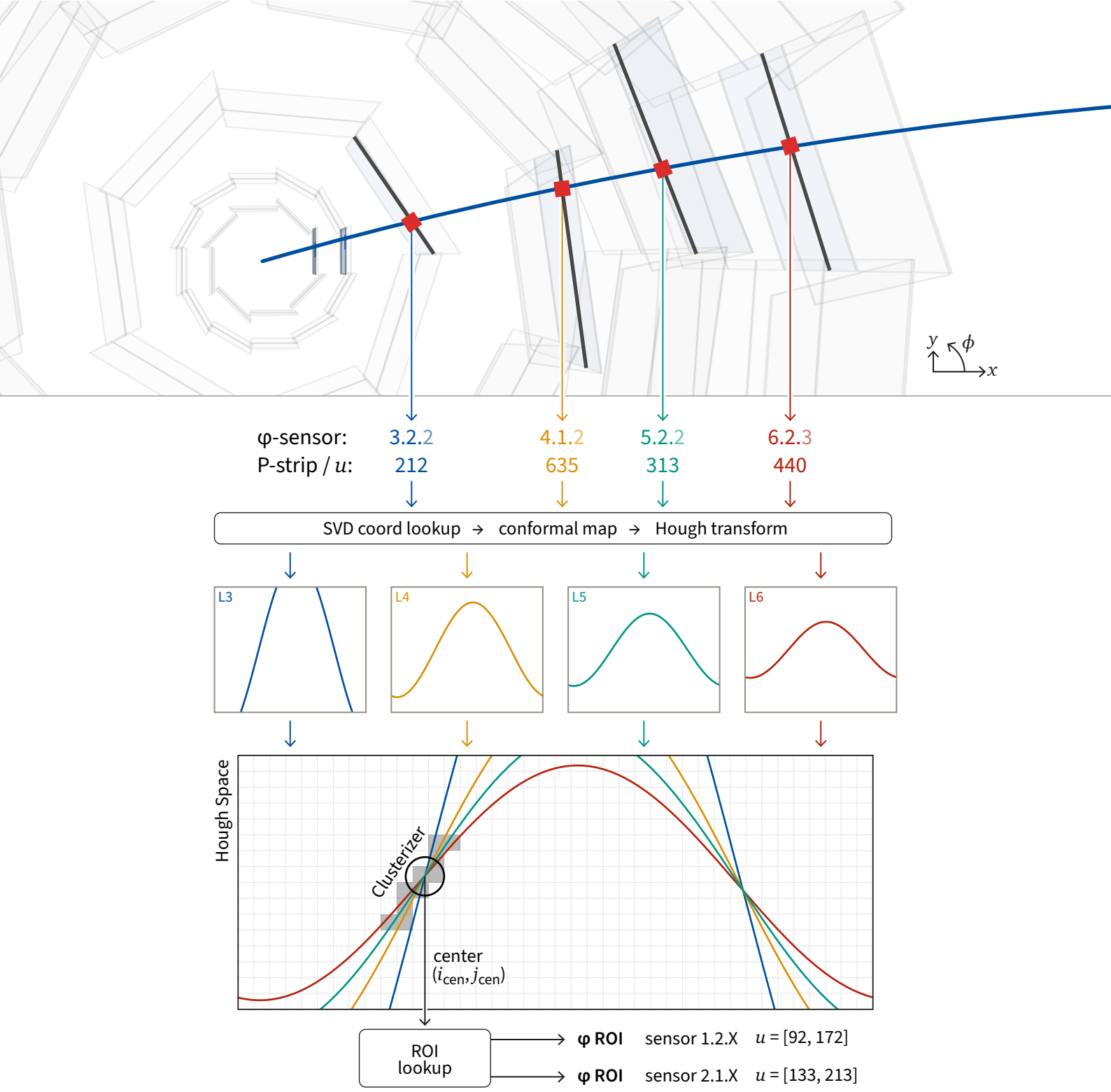


Figure 3.3 – From SVD hits to ROIs with coordinate transformations and lookups. *Shown is the processing of a single track in the φ projection. For each of the four SVD hits (in layers 3, 4, 5 and 6), the SVD sensor without the forward-backward-information is available, together with the index of the active strip. The 2D coordinates of the strip in the φ projection are obtained, and the conformal mapping is applied. The Hough transformation maps those coordinates to sinusoids in the Hough space. The Clusterizer searches intersections of the curves. The cluster centres correspond to a set of ROIs on the PXD sensors.*

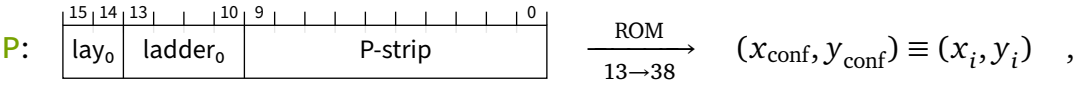
3.3.1 Hough space construction

The HS is constructed in the **hs_phi** and **hs_theta** modules. *Figure 3.4b* shows the block diagram of the **hs_phi** module, the N-side module works similarly.

First, the SVD strip information, which are the sensor ID and strip number, is translated to 2D coordinates in the respective projection: $(x_{\text{conf}}, y_{\text{conf}})$ for the P-side, and (z, r) for the N-side.

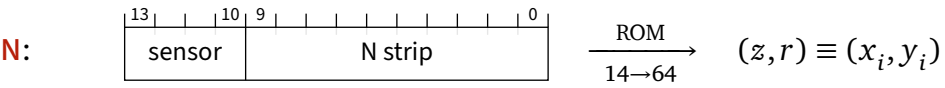
In DATCON, the HS has a hardcoded, fixed size – typically 128×64 cells. Therefore, the geometric calculations for the 3D-2D SVD coordinate transformation, including the conformal mapping for the φ-side, are pre-calculated. A python module fetches the geometry information through **basf2** and applies the projections. The results are scaled, rounded to integers, and stored in a ROM, for lookups by the **coord_phi** and **coord_theta** modules during runtime.

For φ, a 16-bit to 38-bit lookup is used:



where **lay$_0$** and **ladder$_0$** are 0-based indices for the φ-sensors, and **P-strip** can take the values 0–767. The x_{conf} and y_{conf} are in units of mm^{-1}, scaled by a factor of 10^7 and stored as signed 19-bit integers.

For the θ projection, a 14-bit to 64-bit lookup is used:



The N **sensor** is a compact representation of the 14 possible θ-sensors. **N strip** takes the values 0–511 for L3 and 0–767 for L4–L6. The z and r values are in mm, scaled by 10^3 and stored as signed 32-bit integers.

In addition to the SVD coordinates, the θ_{HS} values at the edges of the HS columns are known in advance. All possible $\cos\theta_{\text{HS}}$ and $\sin\theta_{\text{HS}}$ values are precomputed and stored as *edge* values for lookup. The FPGA then performs only multiply-accumulate operations on the looked-up values, which can efficiently be implemented using Digital Signal Processing (DSP) slices:

$$\rho_{\text{HS}} = \underbrace{x_i}_{\text{coord ROM}} \cdot \underbrace{\cos\theta_{\text{HS}}}_{\text{edge ROM}} + \underbrace{y_i}_{\text{coord ROM}} \cdot \underbrace{\sin\theta_{\text{HS}}}_{\text{edge ROM}}$$

From the ρ_{HS} values at the edges of the HS columns for each hit, the intersections of the sinusoids with each cell can be determined. This is done in the column_control and column_check modules. To increase throughput, the HS is parallelized into L *levels*. In the base configuration with 128 HS columns and $L = 8$, the first group of 8 columns are processed in parallel by 8 column_check workers. For each hit (x_i, y_i), each worker computes the ρ_{HS} values for the left and right edge of its current column and compares them with a hard-coded list of ρ_{HS} values of the fixed HS rows.

Any cell traversed by a sinusoid is flagged as *crossed*. Additionally, the layer of origin of the SVD hit is recorded in a per-cell 4-bit register, the *layer filter*, where each bit corresponds to one of the four SVD layers. After all hits for a column are processed, the cells with only 1 or 2 active bits are *discarded*, as they correspond to track candidates which have seen hits from fewer than 3 out of 4 SVD layers. The remaining cells are flagged as *active*.

After the column_check workers finish, they are assigned the next group of columns (e.g. columns 9–16 in the second cycle), and continue until all columns are processed. In the base configuration with 128 columns and $L = 8$, this requires $128/8 = 16$ cycles. The processed columns, represented as a 64-bit vector with one bit per row, are passed on to the clusterizer after each cycle.

The number of parallelization levels L affects the FPGA resource utilization, while the total number of cycles determines the processing time. Both parameters are configurable and must be tuned, as explored in *Section 4.8: Optimizations for hardware*.

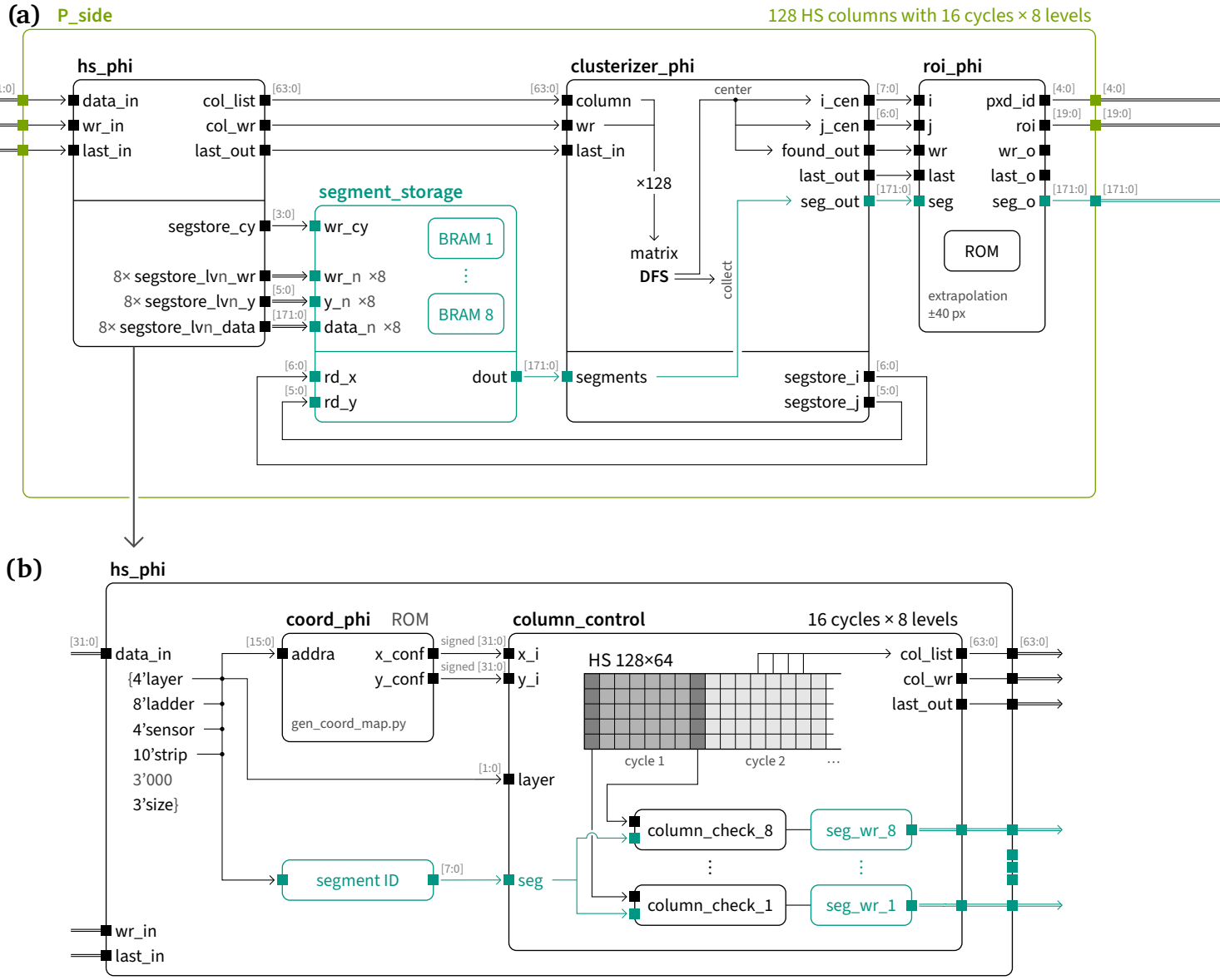


Figure 3.4 – Block diagram of the P-side main modules. **(a)** *The* P_side *main module receives the hit data after the strip exchange with the N-side FPGA. It passes the hits to the* hs_phi *module, which applies the conformal mapping and Hough transformation from the LUT and creates the binary HS matrix. The resulting columns are passed as the* clusterizer_phi *module, which uses a depth-first search to detect adjacent cells, and obtain their centres as* i *and* j *coordinates. Those coordinates are mapped to 1D ROIs by* roi_phi. **(b)** *The* hs_phi *module calculates an* address *and a* segment_id *from the SVD hit data. The* coord_phi *ROM is used to transform the 3D SVD strip information to HS parameters* (x_i, y_i). *The* column_control *processes the HS by* $L = 8$ *parallel levels. 8* column_check *worker modules process one column at the time, and calculate the intersections of the sinusoids with the column edges. They apply the layer filter to determine the active cells, and merge the* segment_ids *to* segment_mask*s which handled by the* segment_writer *and* segment_storage *modules.*

3.3.2 Clusterizer

The Clusterizer finds all groups of adjacent active cells in the HS matrix, called *clusters*. The centre of each cluster approximates the intersection point of the underlying HS sinusoids, and corresponds to a particle track.

The algorithm identifies clusters by scanning the HS for an unprocessed active cell. From that cell, a depth-first search (DFS) finds all other connected cells belonging to the same cluster. The efficiency of this search is improved by restricting the search neighbourhood based on the expected cluster orientation.

For the P-side, only the rising sections of the sinusoids are considered, where their slope is positive. This results in clusters that are elongated from the bottom-left to the top-right in the HS matrix.

For the N-side, the restricted θ_{HS} range of 101.5° to 188.5° means that only the falling sections of the sinusoids are present in the HS. These sections have a negative slope, causing clusters to be elongated from the top-left to the bottom-right.

If a neighbouring cell is active, the Clusterizer moves to that cell and checks again for neighbours. If no further neighbours are available, the Clusterizer steps back to the previously visited cell and continues to check the remaining directions, until all adjacent active cells have been visited. While walking through the active cells, the Clusterizer sums the i and j coordinates of each cell and maintains a count of the visited cells. This count defines the cluster size, N_{cluster}.

The coordinates of the cluster's centre of gravity, $(i_{\text{cen}}, j_{\text{cen}})$, are obtained on the FPGA using integer division:

$$i_{\text{cen}} = \left\lfloor \frac{\sum i}{N_{\text{cluster}}} \right\rfloor \qquad j_{\text{cen}} = \left\lfloor \frac{\sum j}{N_{\text{cluster}}} \right\rfloor$$

This process repeats until all active cells in the HS have been assigned to a cluster.

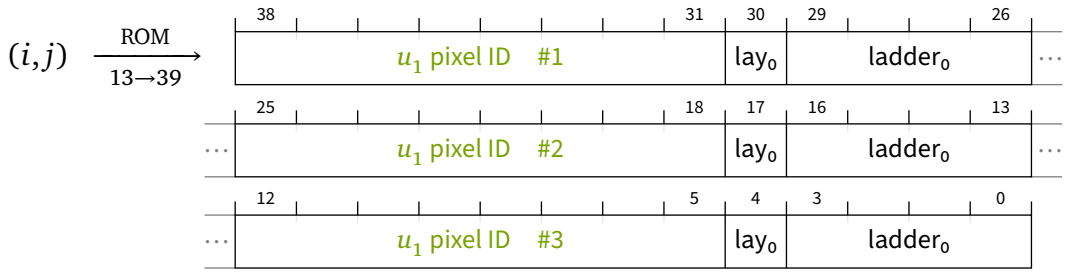
3.3.3 Extrapolation lookup and ROI creation

After the Clusterizer found a cluster centre, it's coordinates i_{cen} and j_{cen} are passed to the roi_phi module. At the core of the module is a precomputed ROM, stored as BRAM on the FPGA. It corresponds to the counterpart of the coord_phi module, and applies the reverse Hough transformation and extrapolation of the tracks to the PXD layers. This replaces the on-FPGA computation in the previous version of DATCON.

Each HS cell (i, j) corresponds to exactly one set of track parameters: r, ϕ for the P-side, and θ for the N-side. Those track parameters define a circle (P) or a straight line (N), which is then extrapolated onto the PXD layers. The pixel closest to the intersection is obtained.

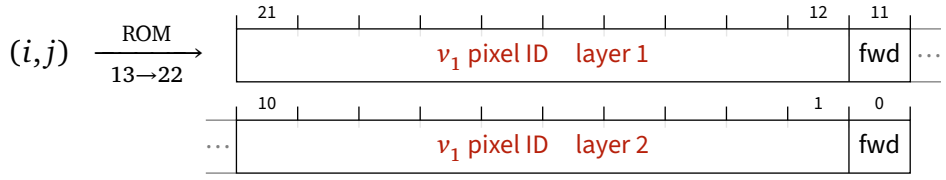
For the φ projection, we expect between one and three intersections of each track with the PXD: one in L1, one in L2, and optionally a second intersection on either layer in the overlapping regions.

In the case of a 128×64 cell HS, the φ lookup is implemented as 13-to-39-bit ROM:



The obtained u_1 pixel ID has a value between 1 and 250, a special value of 0 indicates that no ROI was created. **lay₀** (0 or 1) and **ladder₀** (0–7 for L1, 0–11 for L2) are 0-based indices representing the 20 different φ-sensors.

For the θ projection, there are four sensors. A track can have up to two intersections with the PXD sensors, one in the first layer and one in the second layer. For the extrapolation, the backwards sensors are extended in forward direction to close the gap between the sensors. The data stored in the lookup table are 22 bits per HS cell:



The v_1 pixel ID takes values between 1 and 768. The two pixel IDs from the lookup are assigned to the two layers. The **fwd** flag indicates whether each intersection's pixel is on the forward or backward sensor of the corresponding layer.

After the extrapolation lookup, the extrapolated position on the PXD sensors are converted to intervals by adding ±40 pixels to the extrapolated pixel ID. In the φ projection, a 1D ROI is then defined by a φ-sensor identifier with 20 distinct values, and the pixel interval $[u_{\min}, u_{\max}]$ along the short side of the PXD module. In the θ projection, a 1D ROI is defined by a θ-sensor identifier with four distinct values, and the interval $[v_{\min}, v_{\max}]$.

For the θ side, if the extrapolated pixel ID is too close to the edge of the sensor, an additional ROI is created on the adjacent sensor. This *ROI bleed* allows to correct for small extrapolation errors which could otherwise lead to missing ROIs, as the wrong sensor was found. While the ROI bleed increases number of ROIs and lowers the data reduction factor (DRF), it is a necessary trade-off for the increased efficiency of the track reconstruction. For the φ side, the bleed is implemented in the ROM directly.

3.4 Segment veto

After the track reconstruction was performed separately, the P-side module then *combines* the 1D φ- and θ-ROIs to form the two-dimensional ROIs describing a region on a PXD sensor, which can be sent to the ONSEN system.

The combination of φ and θ ROIs is performed for each sensor individually, by 40 roi_combine modules. For a given PXD sensor, such as sensor 2.6.2, each φ ROI on the same layer and ladder (2.6.X) is combined with each θ ROI on the same layer and sensor (2.X.2). This approach results in a total number of ROIs equal to the product of the number of φ ROIs and θ ROIs matching that sensor. Notably, the same θ ROI will be present on all ladders on the same layer, if they also happen to have any φ ROI. This results in *bands* of ROIs in the φ direction across the PXD. The same holds for the φ ROIs as well, which produce bands along the z direction, as can be seen in *Figure 3.5a*. This increases the overall coverage of the PXD and makes it impossible to achieve the desired DRF.

This banding is the major limitation of DATCON's previous implementation. A new method was developed and implemented to reduce the amount of combinations and prevent the creation of bands in the final output of DATCON. This approach is referred to as *segment veto* and is described in the following. *Figure 3.5b* gives a preview on the effect of the segment veto.

Consider an example with two particle tracks: one travelling *forwards and upwards*, and the other *backwards and downwards*. After applying the projections, DATCON's θ side would reconstruct two θ-tracks (one *forward* and one *backward*) and the φ-side two φ-tracks (*upwards* and *downwards*). Combining these would yield four ROIs per layer, although only the *forward-upward* and *backward-downward* combinations correspond to real tracks. The information linking the directionality in three dimensions is lost when SVD hit information is separated into the θ and φ projections and processed independently.

The core idea of the segment veto is to exclude specific combinations of θ and φ ROIs that are physically implausible, retaining only those consistent with actual tracks. To achieve this, it is necessary to trace each 1D ROI back to its originating track in 3D space, and match the 1D ROIs based on this information. The 3D information is encoded in *segments*, which are distinct regions of the SVD.

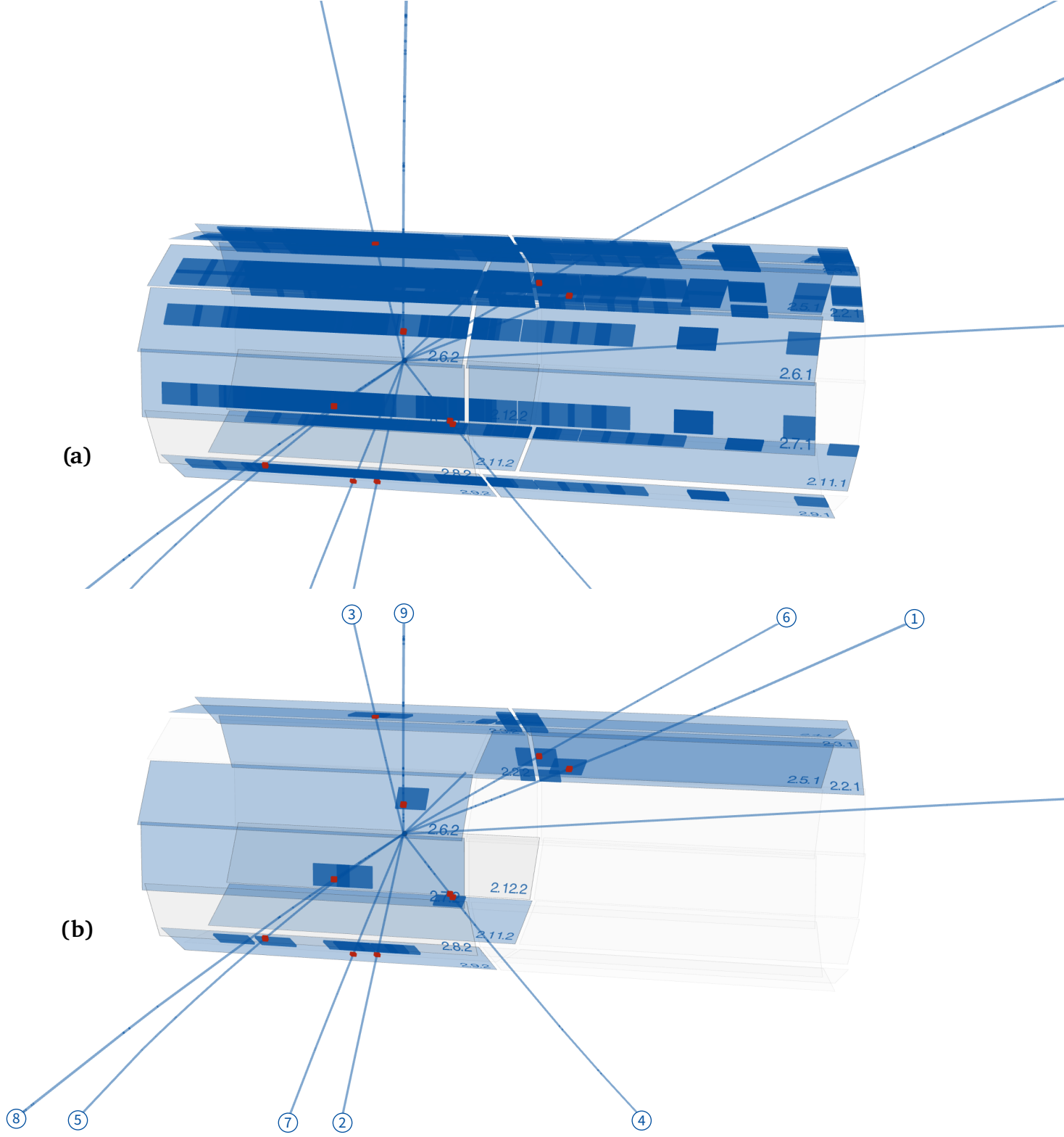


Figure 3.5 – Fake 2D ROIs from 1D ROI combination, impact of segment veto. *Displayed is the second layer L2 of the PXD, with simulated* — tracks, ■ true PXD hits *and the* ■ 2D ROIs *created by DATCON.* **(a)** *shows the previous version of DATCON. The combination of all φ with all θ-ROIs leads to bands of fake ROIs, as can be seen both along the z direction, and also along ϕ (on the right).* **(b)** *is the same event, but processed with a segmentation of 172 and a veto of 3 segments. Most fake ROIs are removed. The HS are shown in Figures 4.1a and 4.2.*

3.4.1 Segment ID

The 3D coordinates of a hit are not available at the time DATCON runs, as the P and N strips of the SVD are read out separately. However, each triggered SVD strip is associated with a specific SVD sensor, meaning that each SVD hit will trigger a P-strip and an N-strip in the same sensor. Thus, a first choice of a 3D segmentation is to use the SVD sensors themselves. For each hit, a *segment ID* is calculated. In this case, the segment ID is the index of the SVD sensor, 0–171. A particle track is then represented by a list of segment IDs, or *segment mask*, listing all the sensors it passed through. If the segment masks match between a φ and a θ ROI, they originate from the same 3D track, and their combination is retained.

In the firmware, the segment ID is an 8-bit integer calculated in the hs_phi/theta modules (see *Figure 3.4b*). Together with the HS parameters x_i and y_i (for calculating ρ_{HS}), and the layer number (for the layer filter), the segment ID is passed to the column_check modules. The column_check modules keep track of a segment mask for each cell of the HS column they currently process. The segment mask is a 172-bit register, where each bit corresponds to one segment. For each crossing of a curve with a cell, the bit corresponding to the segment ID of that curve is set. After all hits have been processed for the column, the column_check modules hand over the list of segment masks together with the list of active cells to the Clusterizer, through the segment_writer and segment_storage modules.

While the Clusterizer processes the active cells in a cluster, it also retrieves the corresponding per-cell segment mask from the segment_storage. The segment masks of all cells in a cluster are collected and combined with a logical OR operation. The final segment mask of the cluster contains all segments of the contributing hits. It is then passed to the roi_phi/theta modules, through the rest of the DATCON firmware, and finally to the roi_combine module, where the φ and θ segments are compared.

veto threshold Returning to the two-track example, the segments associated with the *forward* θ-track will largely overlap with those of the *upwards* φ-track, as both originate from the same 3D trajectory and thus the same SVD sensors. They are not expected to match exactly, as different other hits may have contributed to the cluster in each projection. In contrast, the *downwards* φ-track will have a distinct segment mask. When combining φ and θ ROIs, a criterion can now be set on the degree of overlap between their segment masks, called the *veto threshold*.

For a segmentation of 172, where each segment represents one SVD sensor, a veto of up to 4 segments is possible. In case of a perfect track reconstruction in the HS, and no other tracks in the detector, the track will have passed exactly four SVD sensors, so their four segment IDs will be present in both θ and φ ROIs.

3.4.2 Segment storage

The Clusterizer records the incoming columns in a 2D matrix (memory) of 128×64 bits. Each cell requires one bit to indicate if it is active or not. Contrarily, each cell requires 172 bits to store the segment mask, which is too large to be implemented directly in the logic cells of the FPGA. Instead, a set of dual-port BRAM is used, managed by the segment_storage module. Because the columns in the HS are checked using multiple levels in parallel by the column_check modules, multiple independent BRAMs are necessary. A block diagram of the segment_storage module is shown in *Figure 3.6*.

Each column_check module has an associated segment_writer, which provides the interface to one of the segment_storage BRAMs. After the column_check module finished processing all hits for a column, it passes the list of active cells (64-bit) and the list of segment masks (64×172-bit) to its segment_writer module. The segment_writer module iterates over the list of active cells, and writes the corresponding segment masks to the segment_storage, addressed by the cell index and the level. The BRAMs are large enough to hold segment masks for the entire HS. To save clock cycles, the segment_writer scans the list of active cells in chunks of 16 cells in advance. The column_control will send the next set of columns after all segment_writer modules have finished writing their segment masks.

Besides the 8 parallel write interfaces (for the segment_writer modules), the segment_storage module also provides a single read interface for the clusterizer. The Clusterizer processes the HS cell-by-cell, thus only a single reading port is required. The segment_storage uses the Clusterizer's current cell index (i, j) to select the correct BRAM and returns the corresponding 172-bit segment mask.

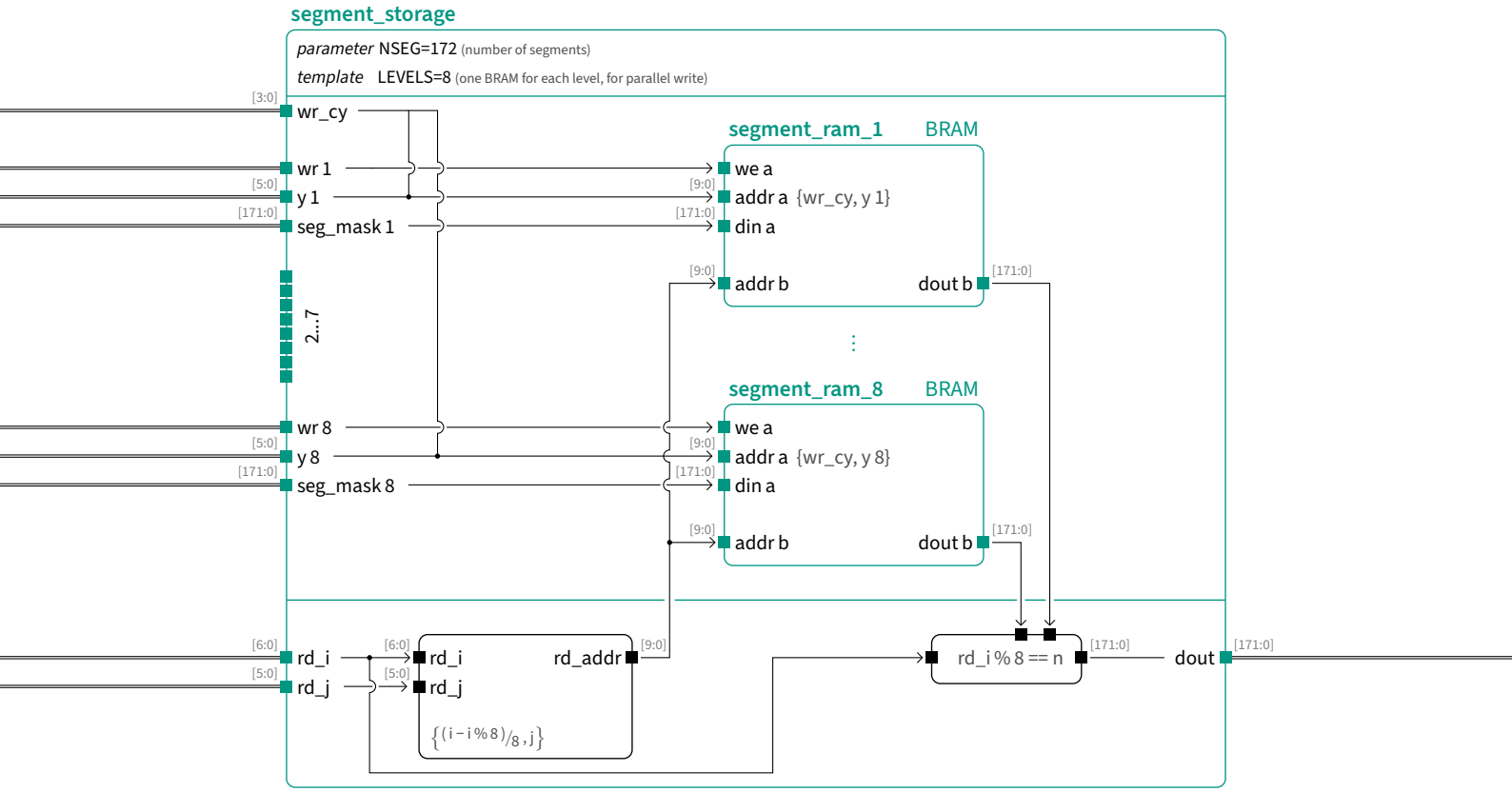


Figure 3.6 – Block diagram of the segment storage module. *The* segment_storage *module provides a set of dual port BRAMs for storing the segment masks associated with a HS cell. It is written by the* hs_phi/theta *module, when the* column_check *worker modules process the columns, and read by the* clusterizer *while iterating over the HS cells. The number of BRAMs match the number of* column_check*, which will write the segment masks for their active cells in parallel (upper part). When the* segment_storage *is read out by the* clusterizer*, the access happens one HS cell at the time (lower part).*

3.4.3 Number of segments

While the choice of 172 segments is the closest match to a full 3D representation with the limited SVD data, it is also the most resource-intensive, as 172 bits need to be stored and transferred for each 1D ROI before the veto can be applied to their combination. Other choices with fewer segments are also considered, to reduce the load on the FPGA:

- **87 segments.** Two adjacent SVD sensors could be grouped to one segment, either along z or along ϕ. When merging two ladders together, 87 segments are required.
- **81 segments.** Under the assumption that each track should reach the outermost SVD layer (SVD6), the number of segments could further be reduced. The SVD6 layer has 16 ladders with 5 sensors each, for a total of 80 sensors. Only the hits on the sensors of SVD6 are used as veto, the hits on the other layers won't be considered. As each hit requires a hit ID to be passed along the HS, an additional 81st bit is reserved for hits originating from layers 3–5. This allows differentiation from SVD6 hits. The 81st bit is disregarded when applying the segment veto. Only a segment veto threshold of 1 is possible.
- **15 segments.** As a minimal configuration, the innermost SVD3 layer can be used similarly. With 7 ladders and 2 sensors each, 15 bits are required.

The performance of DATCON with different segment numbers is evaluated in *Section 4.5: Segmentations*, while the hardware implementation constraints imposed by the different segmentations on the FPGA is discussed in *Section 4.8: Optimizations for hardware*.

3.4.4 ROI merging and ROI combination

The introduction of segments and the segment veto now also necessitates a change in the way overlapping ROIs are merged. Previously, overlapping 1D ROIs were merged on their respective side right after the extrapolation lookup, inside the roi_phi/theta modules. The ROI merging was introduced to assure the number of ROIs sent to ONSEN does not exceed the limit of 128, but it does not have any impact on the data reduction factor, as the covered sensor area remains the same.

However, with the segment veto, it is now possible that two 1D ROIs on the same sensor originate from different segments, merging them before the segment veto is applied would lead to a loss of information. Therefore, the 1D-merging of overlapping ROIs was replaced with a 2D-merging, which is now performed after the combination of φ and θ ROIs, in the roi_combine module. A schematic overview of the ROI merging with veto is shown in *Figure 3.7*.

The state machine of the updated roi_combine module with 2D merging proceeds through the following phases:

– **collect.** The roi_combine module receives the φ- and θ ROIs on two write ports and stores them in separate BRAMs.

– **combine.** For every (φ, θ) ROI pair, retrieve their segment masks. The Hamming weight of the bitwise AND of the two masks is calculated and compared with the set veto threshold

$$\text{popcnt}(\text{segments\_phi} \wedge \text{segments\_theta}) \geq \text{veto} \quad .$$

If enough overlapping segments are found, the resulting 2D rectangle is written to a 2D-ROI BRAM.

– **wait.** After all 1D ROIs are consumed, the FSM waits for read from the roi_send module, as it processes the 40 roi_combine modules in order.

– **merge.** The roi_combine performs an in-place merging on the 2D-ROI BRAM using two monotonically increasing address pointers a and b. Starting from $a \leftarrow 0$ and $b \leftarrow 1$:

  – Retrieve the 2D ROIs at addresses a and b.

  – If the rectangles do not overlap, increment b.

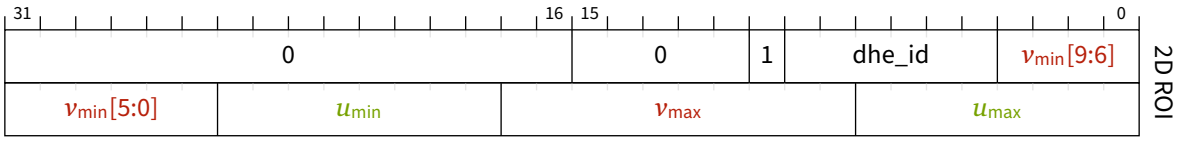
  – If they overlap, merge them to a larger ROI, write the result to address b and clear address a. Increment a and set $b \leftarrow a + 1$.

  – If b reaches the end of the written ROIs, increment a and set $b \leftarrow a + 1$.

  – If a reaches the end of the written ROIs, the merging is complete.

The in-place merging has a minimal BRAM footprint, as necessary for implementation on the FPGAs (see *Section 4.8: Optimizations for hardware*).

– **send.** The BRAM is scanned again, and the non-zero ROIs are written to the output, as two 32-bit words:



where the dhe_id is the unique identifier of the DHE associated with the PXD sensor, defined as

$$\text{dhe\_id} = 32 \times (\text{layer} - 1) + 2 \times \text{ladder} + (\text{sensor} - 1) \quad .$$

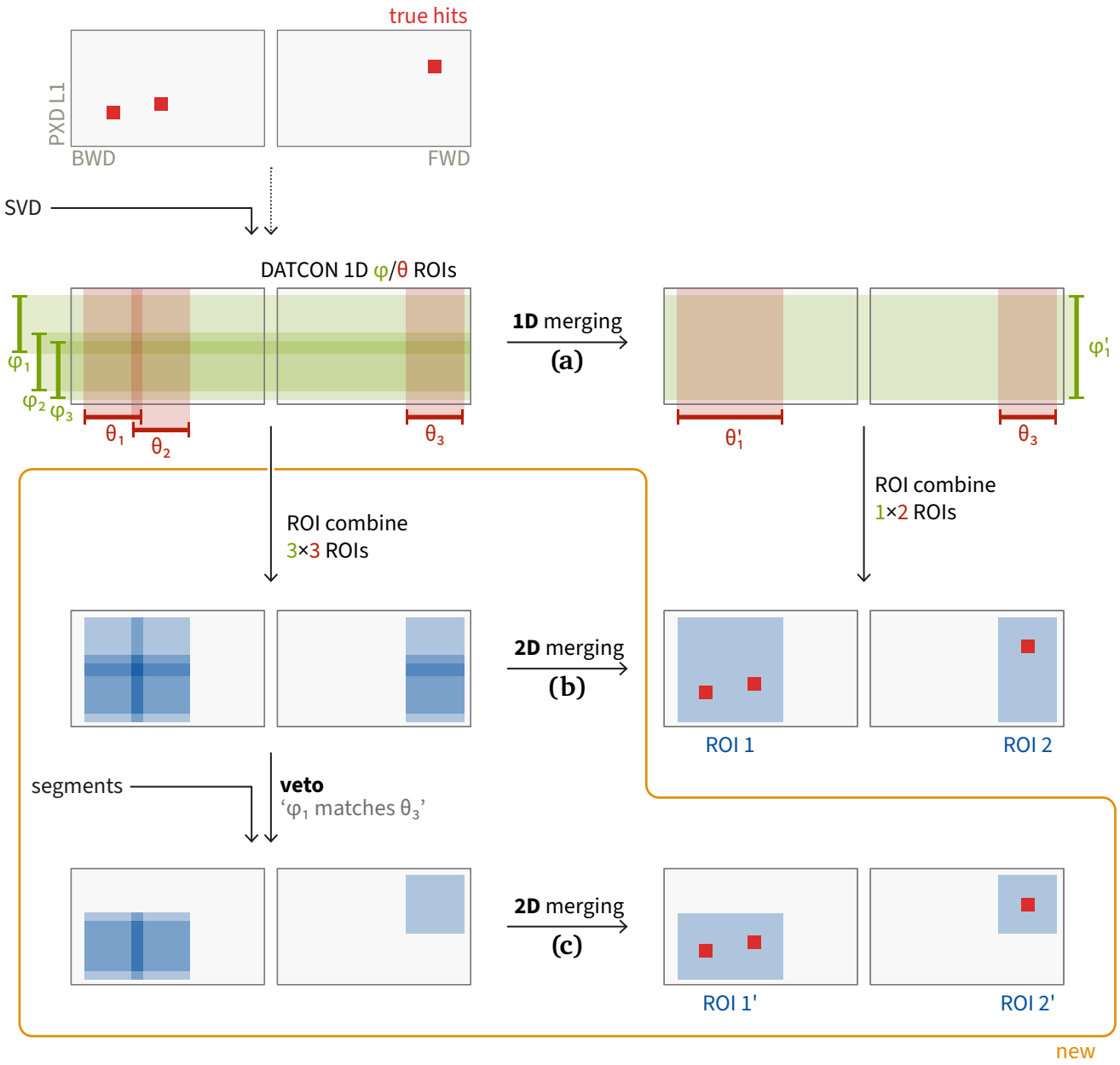
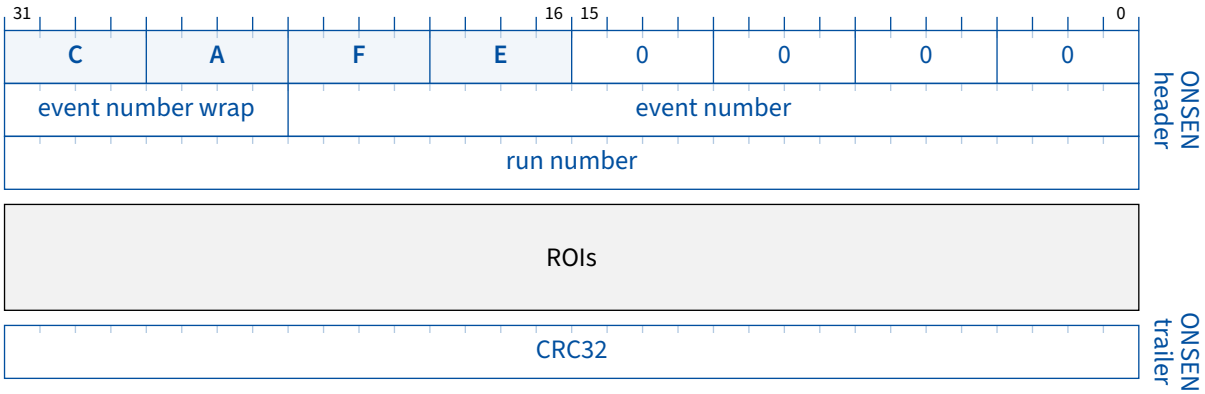


Figure 3.7 – Comparison of 1D and 2D ROI merging. *Shown are the forward and backward sensor of a PXD ladder. The ROI combine step creates the 2D ROIs from the 1D φ- and θ-ROIs. The merging of overlapping ROIs can either happen with* **(a)** *the 1D ROIs, or* **(b)** *the 2D ROIs, leading to identical results. For the segment veto, 2D ROIs are required. After filtering out 2D ROIs with non-compatible φ and θ combinations, the 2D merging* **(c)** *leads to a smaller covered detector surface.*

3.5 DATCON-ONSEN link

The ROIs are sent to ONSEN ordered by increasing sensor ID / **dhe_id**. This is handled by the **roi_send** module, which iterates over the 40 **roi_combine** modules and requests their ROIs via the **read** signal. If 128 ROIs have been retrieved, the **roi_send** module stops requesting further ROIs, and completes the event.

The per-event data sent to ONSEN are prepended with a header containing the event number and appended with a CRC32 checksum:



3.6 DATCON-mini

While the physical DATCON is implemented using 15 interconnected FPGA cards, a reduced version, called DATCON-mini, is used for testing, simulation and development. Firmware-wise, it contains only the core modules of the full DATCON system, and skips the event/strip management and the logic for handling the data transfer between the FPGAs. The firmware layout is shown in *Figure 3.8*. DATCON-mini can be run on a single Virtex-6 FPGA card.

DATCON-mini has multiple ways of receiving input data:

- **IPbus.** If DATCON-mini runs on hardware, it can receive SVD data from a connected computer via the IPbus protocol.
- **txt file.** If DATCON-mini runs in a Questa simulation, stored SVD data can be read from a text file using the **$readmemh** Verilog function.
- **cocotb.** If the simulation is run through **cocotb**, data can be fed from various sources directly to the **strip_data** registers.

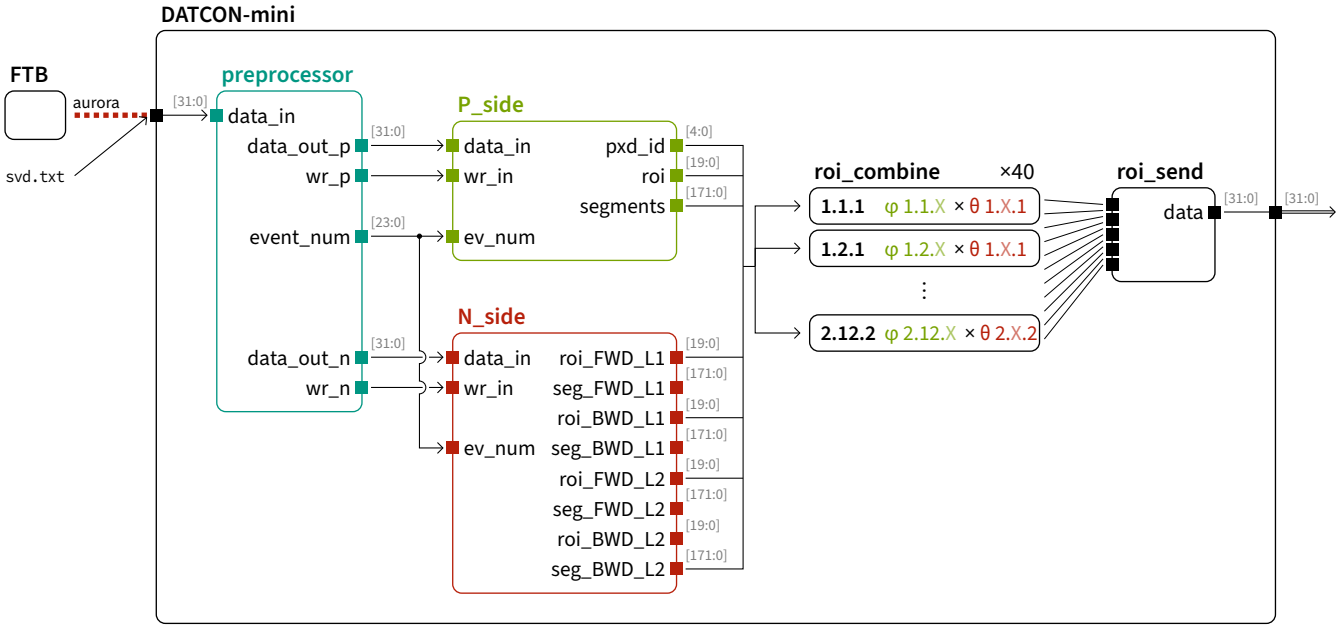


Figure 3.8 – Block diagram of DATCON-mini. *Instead of the 52 FTBs present at Belle II, a single input source is used for DATCON-mini. One* preprocessor *is sufficient to handle the decoding of the data and passing them to the two Tracking modules. The* P-side *and* N-side *modules obtain the hit data, perform the track reconstruction and output 1D φ and θ ROIs, along with their segment masks. The* roi_combine *modules merge the 1D ROIs into 2D ROIs for each of the 40 PXD sensors, optionally applying a segment veto and merging overlapping 2D ROIs. The* roi_send *reads the ROIs sequentially and formats them for ONSEN. In simulation, the output can be intercepted and stored to disk for further analysis.*

Chapter 4

Performance

In the following, various aspects of the performance of DATCON are discussed, comparing the previous and improved versions of the firmware using simulated $\Upsilon(4S)$ events. After an introduction to the test setup, and an overview of the figures of merit (*Sections 4.1 and 4.2*), the chapter evaluates the following aspects:

- **Hough space.** The track finding happens in the two Hough spaces (HSs). An analysis of the Clusterizer output provides a first insight into the track finding abilities of DATCON, as explored in *Section 4.3: Hough space resolution*.
- **1D ROI creation.** *Section 4.4: Verification of ROI size with residuals* discusses the choice of ROI padding around the extrapolated PXD position.
- **segmentation and vetoing.** The impact of the number of segments is explored in *Section 4.5: Segmentations*, and of the veto threshold in *Section 4.6: Performance of veto threshold settings*.
- **HLT.** The ROIs generated by DATCON are compared against those from the HLT to benchamrk performance, as shown in *Section 4.7: Comparison with HLT*.
- **hardware limitations.** Finally, *Section 4.8: Optimizations for hardware* discusses the real-world limitations imposed by the FPGA hardware and their impact on the achievable performance.

4.1 Setup

For the presented performance studies, simulated events have been used, as they allow access to the truth information:

- **true PXD hits.** The intersections of the simulated particle tracks with the PXD sensors are required to be able to define if DATCON correctly identified a hit or not.
- **true tracks.** The performance of DATCON can be characterized in function of the track parameters of the true track like θ or p_{T}. Also, the true PXD hits of a track will be excluded from the analysis if it did not originate from the primary process. The true particle trajectories are also used to generate event displays.

The simulation is done within Belle II's basf2 framework. For the simulation of the $\Upsilon(4S)$ and B decay chains, the EvtGen [58] event generator was used. Those particles are then handled by Geant4 [59] which simulates the particle interactions with the detector material and their propagation. While the particles propagate through the detector, the basf2 framework simulates the readout of the subdetectors and allows the storage of the particle tracks and the detector response. As input for DATCON, the simulated detector readout of the SVD is required. All data are stored on disk as HDF5 files, so that the DATCON algorithm can be re-run on the same simulation with different parameters. All plots in this thesis have been produced based on a single simulation of 10 000 $\Upsilon(4S)$ events.

4.2 Figures of merit

The goal of DATCON is to reconstruct all hits on the PXD, while also keeping the PXD readout rate low. In the following, different figures of merit are introduced to quantify the working point of DATCON.

4.2.1 Hit finding efficiency (HFE)

The hit finding efficiency (HFE) describes the number of true hits which are contained inside the ROIs found by DATCON:

$$\mathrm{HFE} = \frac{n_{\text{true hits inside ROI}}}{n_{\text{all true hits}}} \quad .$$

If all true hits would be contained inside DATCON ROIs, a HFE of 100 % would be achieved.

A HFE can only be given for simulated data, when the true hits of the tracks inside an event are known. In the following, the HFE is given as a function of the p_T and the polar angle θ_{track} of the corresponding true track.

In practice, the HFE is calculated by

- iterating over true PXD hits, obtained from the simulation
- iterating over all DATCON ROIs
- if any DATCON ROI contains the true hit, count the hit as *found*.

In the previous publications on DATCON [2, 60], the HFE was also called *ROI Finding efficiency* or *ROI efficiency*.

4.2.2 Track finding efficiency (TFE)

Instead of looking at the hits, one could also define an efficiency based on the tracks, a track finding efficiency (TFE). However, in the context of DATCON, it is much more natural to argue in terms of hits, as the ROIs produced by DATCON directly relate to the signals on the PXD sensors.

While internally DATCON also does track finding and fitting, its definition of a *track* is different from the usual track parameterization of a helix. The track finding is done in the HS, for both projections separately, so a comparison with the true tracks from the simulation can only be done on a 2D-level.

4.2.3 Fake rate (FR)

The fake rate (FR) is directly related to the hit finding efficiency and quantifies the number of ROIs which do not contain any true hits. It is given by

$$\mathrm{FR}_{\mathrm{ROI}} = \frac{n_{\text{fake ROIs}}}{n_{\text{ROIs}}}$$

or

$$\mathrm{FR}_{\mathrm{hit}} = \frac{n_{\text{ROIs without a true hit}}}{n_{\text{true hits}}}$$

Fake ROIs could be either caused by the clustering:

- fake hits in the SVD, caused by noise or background, leading to a very crowded HS,
- large clusters, which fail to resolve the individual intersections,
- clusters being split, because the layer filter missed a cell,

or from wrong combination of θ and φ-ROIs (*ghost ROIs*).

The former can be addressed by tuning the HS parameters like its resolution, and prefiltering of the SVD hits. The latter can be addressed by the segment veto.

4.2.4 Number of ROIs per event (nROIs)

The maximum number of ROIs accepted by ONSEN is 128, which imposes a hard limit on the number that DATCON can send. The final module of the DATCON firmware, `roi_send`, iterates over all PXD sensors in order of their ID and transmits their ROIs sequentially. If the 128 ROI limit is reached, any remaining ROIs from sensors with higher IDs are simply discarded, leading to a potential bias by systematically losing information from certain geometric regions of the PXD.

As each ROI is encoded as two 32-bit words using the lower and upper limit in the θ and φ direction, the number of pixels inside a ROI doesn't directly matter for ONSEN, but then affects the data reduction factor (DRF).

While the number of ROIs per event varies with the population of the HS, a high fraction of events with fewer than 128 ROIs is necessary. The fraction of non-exceedance $\hat{p}$ is given by

$$\hat{p} = \frac{\text{count}(\text{nROIs} \leq 128)}{n_{\text{events}}} \qquad \sigma_{\hat{p}} = \sqrt{\frac{\hat{p}(1-\hat{p})}{n_{\text{events}}}} \quad .$$

The binomial standard error $\sigma_{\hat{p}}$ is $< 0.5\,\%$ for a dataset of 10 000 events. For the following plots, the limit of 128 ROIs is not enforced, the shown HFE and DRF are calculated based on all DATCON ROIs.

4.2.5 Data reduction factor (DRF)

The HFE, FR and nROI are not sufficient to characterize the performance of the ROI creation alone. Imagine a situation where one large ROI is created for each PXD sensor, spanning the entire sensor area. The HFE will be 100 %, as no hit can be outside the ROIs. The FR is zero, as there are no additional ROIs without a true hit. The number of ROIs will be 40, well below the limit of 128.

The DRF is a metric which penalizes such large ROIs. It is defined by the surface area of the PXD sensors which are covered by ROIs, expressed by the number of pixels:

$$\text{DRF} = \frac{n_{\text{PXD pixels}}}{n_{\text{pixels inside ROIs}}} \quad ,$$

where the total number of pixels in the PXD is $40 \times 250 \times 768 = 7\,680\,000$.

For further processing (event building, bandwidth, storage), a DRF of 10 is targeted, so only 10 % of the PXD sensor area should be covered by ROIs. While such a reduction is currently not yet enforced by ONSEN, it will become necessary once the luminosity of the SuperKEKB accelerator approaches its design value, and the PXD is expected to deliver 20 GB/s after zero-suppression for an occupancy of 3 % [61].

In the plots in this chapter, the mean DRF is given for the events in the datasets, as well as a percentage of those events which have a DRF above the target of 10.

4.3 Hough space resolution

The HS is directly related to the ability of DATCON to detect tracks. The resolution and range of the HS are crucial to achieve a good track finding performance. As explained in *Section 3.3.1: Hough space construction*, the HS is discrete matrix of cells along its width (θ_{HS}) and its height (ρ_{HS}). Because of the additional conformal mapping applied to the φ side, the physical quantities represented by the axes of the φ and θ HS are different.

θ Hough space. The event from *Figures 2.3 and 2.4* is used to illustrate the HS clustering. *Figures 4.1 and 4.2* show the resulting θ and φ HSs. The ■ active cells, which passed the layer filter (being crossed by sinusoids from at least 3 different SVD layers), act as input to the Clusterizer. The centres of the found clusters are marked with a ⊗ red circle, while the true tracks from the simulation are marked with a ⊗ blue circle. The axes are labelled by their cell indices i and j, which map to the coordinates θ_{HS} and ρ_{HS}.

Figure 4.1a shows the θ HS with a resolution of ● 128×64 cells, which is the baseline setting. The vertical resolution was set to $\rho_{\text{HS}} = \pm 100\,\text{mm}$, which corresponds to a distance from the interaction point (IP) which much larger than meaningful, but keeps the clusters in a diagonal shape, which is optimal for detection with the Clusterizer. In total, 328 active cells passed the layer filter, and were grouped into 40 clusters.

The following observations can be made:

- Many intersections are found in the upper and lower parts of the HS, which cannot represent a true track, and wouldn't match with DATCON's extrapolation from the IP.
- A large cluster around the tracks ② ③ ⑧ and ⑦ is found. As seen in *Figure 2.3*, the tracks have similar θ_{track} values, which makes it difficult to separate them in the HS. The intersection for ⑦ created a distinct cluster, but the other three tracks are merged into one large cluster. The Clusterizer still found cluster centres which are close to the true tracks, which are artefacts from the Clusterizer's directional search.

To address these issues, two changes were made to the HS parameters: The number of cells was increased along the width, to 256. The ρ_{HS} was decreased to $\pm 25\,\text{mm}$, while reducing the number of cells along the height to 32. This leads to a better resolution along the width, which helps to separate intersections which are close in θ_{track}. Further, the lower ρ_{HS} range focuses more on tracks with a vertex close to the IP, while suppressing the fake intersections at high $|\rho_{\text{HS}}|$. The number of cells is the same, keeping the resource footprint of the FPGA mostly the same.

Figure 4.1b shows the resulting HS for ▲ 256×32. While the θ_{HS}-ρ_{HS} coverage is a fourth of the previous HS, the number of active cells (155) and the number of clusters (18) are around half of the previous values. The intersections of the four tracks have been found as distinct clusters, which are very close to the true tracks.

φ Hough space. The φ HS is shown in *Figure 4.2*, with the default resolution of ● 128×64 cells, and a θ_{HS} range of $[-\pi, \pi]$. The θ_{HS} range spans the entire 2π, so that only the rising branches of the sinusoids are considered. The HS contains 161 active cells, and 16 clusters.

Due to the larger θ_{HS} range, the intersections are more spread out along the width of the HS, which helps with their separation. Because of the conformal mapping, the ρ_{HS} axis represents the inverse of the track curvature, with straight tracks at the centre at $j = 32$. Low-p_{T} tracks, which are of particular interest to physics analyses, are found in the upper and lower bands of the HS. A reduction of the ρ_{HS} range, as done for the θ HS, is thus not meaningful.

Performance. The performance of DATCON, for the two presented configurations for the θ HS, is shown in *Figure 4.3*. The HFE for the ● 128×64 is higher in the forward direction, due to the better track separation. Additionally, the DRF increases while the nROIs decreases, as the number of clusters is reduced.

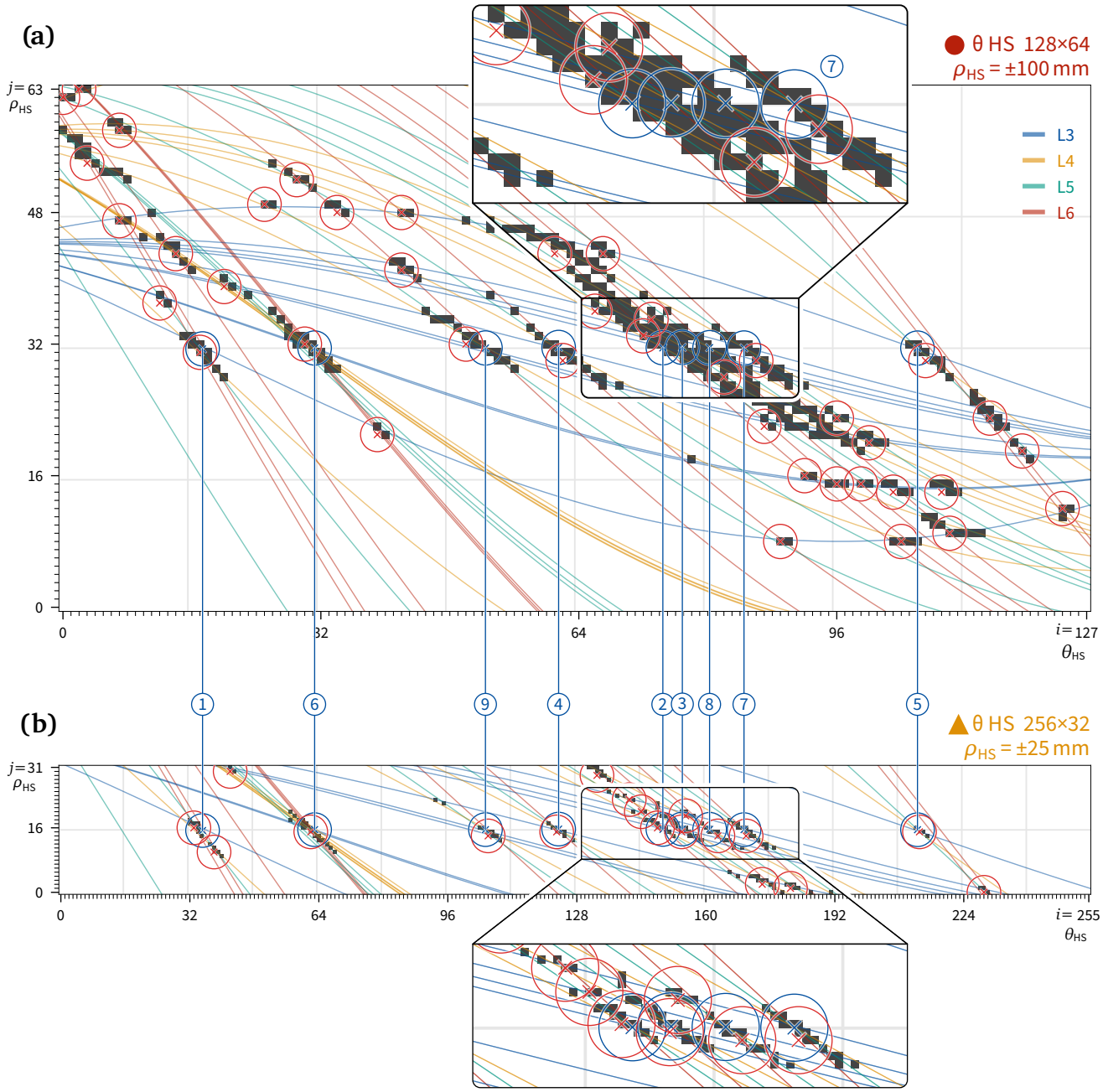


Figure 4.1 – Hough space of θ-projection with different parameterizations. *The event from Figure 2.3 is processed with a θ HS of* **(a)** *128×64 cells with* $\rho_{HS} = \pm 100\,\text{mm}$ *and* **(b)** *256×32 cells with a narrower* $\rho_{HS} = \pm 25\,\text{mm}$ *range. Shown are the* ■ active cells*, the resulting* ⊗ cluster centres *and the* ⊗ true tracks *from the simulation.* θ_{HS} *spans* $[0.2 + \pi/2, 2.76 + \pi/2]$ *rad, slightly above the detector coverage. Figure 2.3 shows that* (2) (3) (8) *and* (7) *share similar* θ_{track} *values, which complicates track reconstruction: In* **(a)** *only track* (7) *is found. The Clusterizer identifies one large active region for the other intersections. In* **(b)** *the four tracks are resolved into distinct clusters which match the true tracks closely. The narrower* ρ_{HS} *range suppresses the fake clusters at the top and bottom of the HS.*

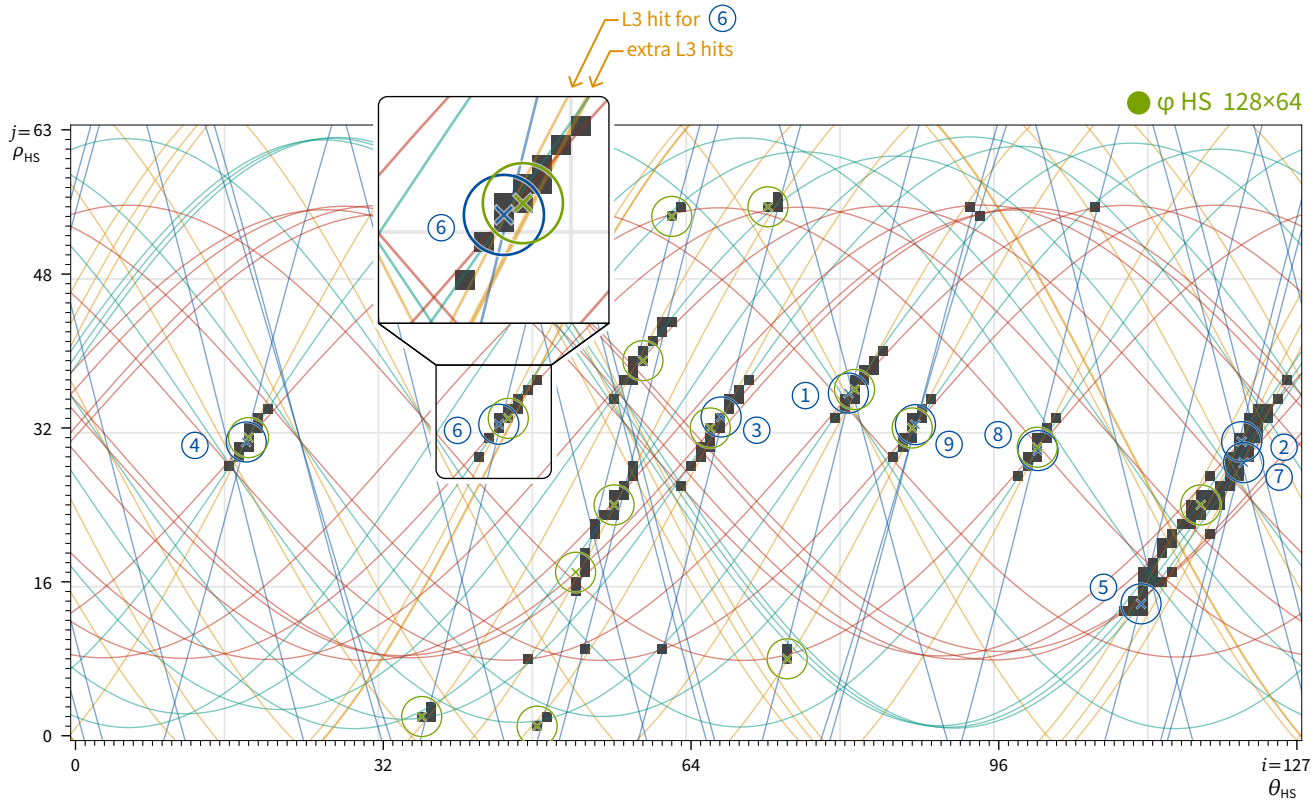


Figure 4.2 – Hough space of φ-projection. *The φ HS is shown with the default parameterization of 128×64 cells, with ⊗ cluster centres and ⊗ true tracks. The θ_{HS} range spans $[-\pi, \pi]$, so only the rising branches are considered. Because of the conformal mapping, the ρ_{HS}-axis shows the inverse of the track curvature, with straight tracks at the centre at $j = 32$. The low-p_T tracks, with a high curvature, can be found at the upper and lower bands. The lowest p_T track in the event, track (5) at the bottom right, has $p_T = 40\,MeV$. For track (6) two effects are visible: first, a cell was missed at the bottom left, due to unfavourable alignment of the cell edges with the curves. Second, the cluster was extended to the top right because of additional — hits in layer 2, which are unrelated to the true track but pass the layer filter. Still, the track could be classified as found, if the ROI size is large enough. For tracks (5) (7) and (2) at the bottom right, the three intersections are merged to one single cluster.*

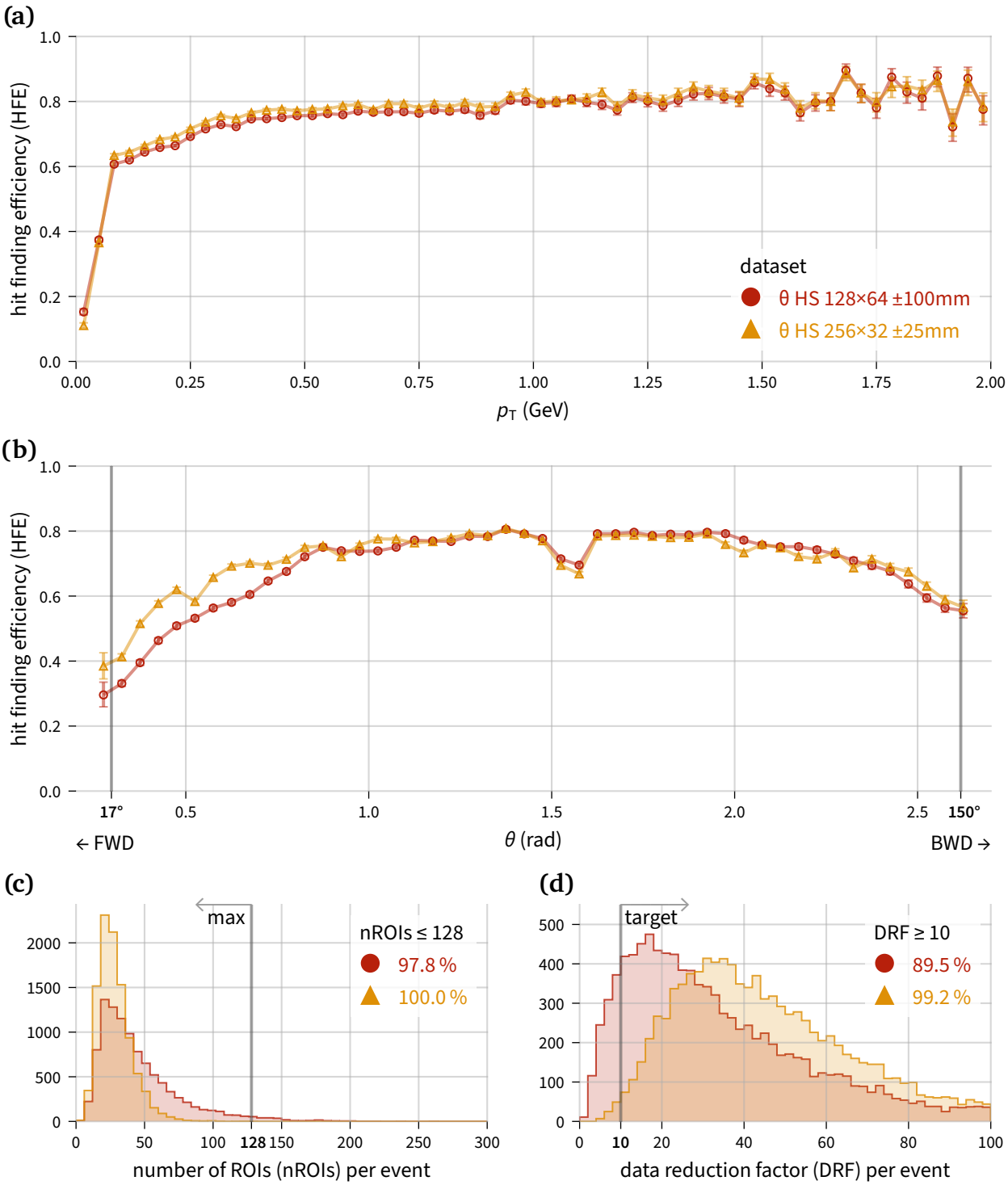


Figure 4.3 – Performance comparison of different θ HS resolutions. *The same dataset is processed with a HS of ● 128×64 cells with $\rho_{HS} = \pm 100\,mm$ and ▲ 256×32 cells with $\rho_{HS} = \pm 25\,mm$. The increased granularity along i / θ_{HS} improves the HFE by discriminating the clusters better. The reduced j / ρ_{HS} range focuses on tracks originating from close to the IP, reducing the fake ROIs far outside the meaningful range. The HSs of a single event are shown in Figures 4.1a and 4.1b.*

4.4 Verification of ROI size with residuals

The clust_to_roi module transforms HS coordinates (i, j) to 1D ROIs. After the pixel IDs corresponding to the track defined by the HS parameters was retrieved from the block RAM (BRAM), the pixels are converted to intervals by applying a fixed padding around the extrapolated hit position. The padding must be chosen to cover uncertainties in the track finding, HS discretization and extrapolation.

The residuals in this section are computed immediately after the Clusterizer finished processing the HS and before the clust_to_roi produces the 1D ROIs. The u coordinates (along the short side of the sensor) and v coordinates (along the long side of the sensor) are analysed separately to assess the P and N-side independently.

As multiple true hits and multiple DATCON extrapolations can occur per sensor, the matching is performed as follows. For each true hit, the DATCON extrapolations on the same sensor are considered. If multiple extrapolations exist, the one with the smallest distance to the true hit is selected. The residuals are then defined as

$$\begin{aligned} \text{P:} \quad & \Delta u = u_{\text{true}} - u_{\text{DATCON}} \qquad u \in [0, 249\,\text{px}] \\ \text{N:} \quad & \Delta v = v_{\text{true}} - v_{\text{DATCON}} \qquad v \in [0, 767\,\text{px}] \quad . \end{aligned}$$

The residuals are calculated in pixel units, not taking into account the varying pixel size of the PXD sensors.

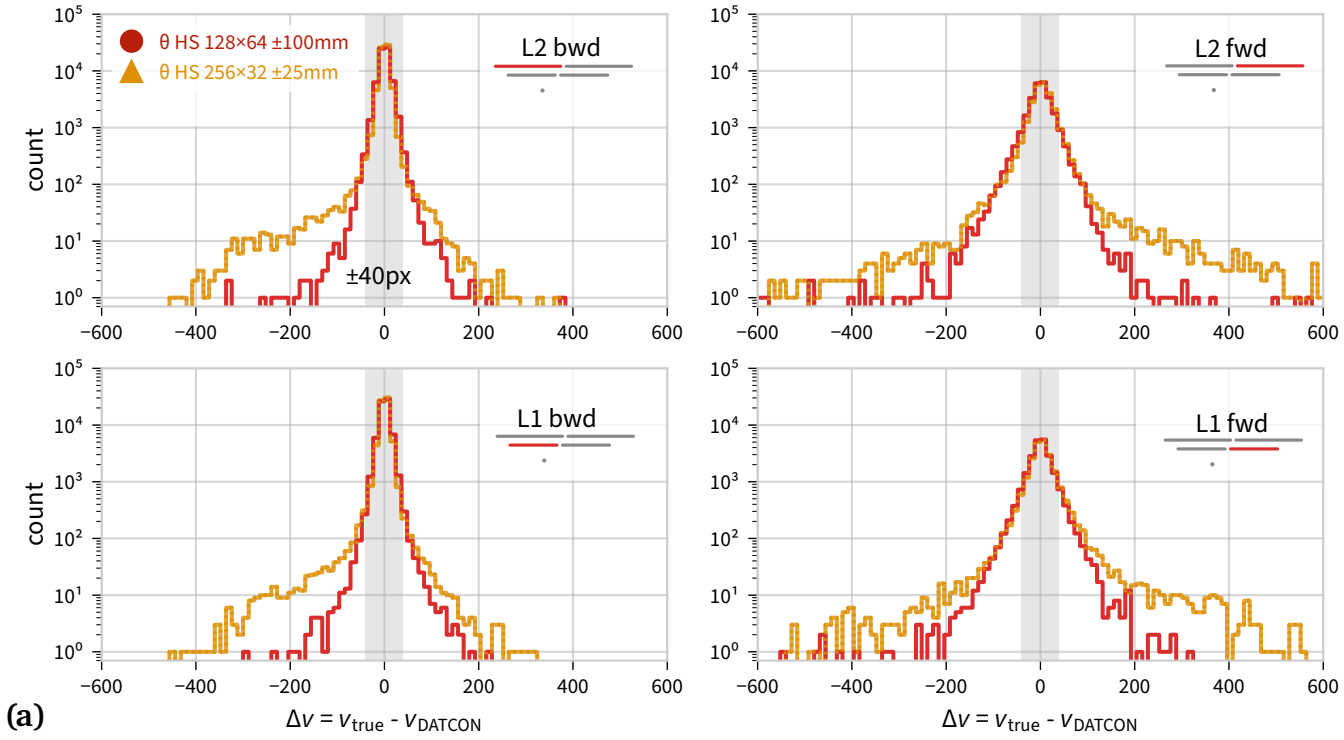
Figure 4.4a shows the residual distribution of the N-side. The following observations can be made:

- The ROI padding of ±40 px is shown as gray band around the origin. The ROI size of 80 px appears to be an adequate choice, as most true hits lie inside that band. For the ● 128×64 HS, the fraction of true hits inside the ROI is 88.4 %/99.1 % on L1 (forward/backward) and 88.0 %/98.9 % on L2 (forward/backward). Note that only the cases where DATCON found a ROI on the same sensor are considered, so these numbers are not the same as the HFE.
- The distribution in the forward direction (L1 fwd, L2 fwd) is broader than in the backward direction. This could motivate region-dependent padding, e.g. increase the ROI size for the forward ROIs.
- Additionally, the HS of ▲ 256×32 is shown. As shown in *Section 4.3: Hough space resolution* and *Figure 4.1*, this reduces the amount of (fake) HS clusters dramatically. On the other hand, this leads to a wider residual distribution, as for each true track there are fewer DATCON candidates. The fraction of true hits contained in the gray band is thus slightly lower, with 85.6 %/98.2 % on L1 and 87.4 %/98.1 % on L2. This is still in an acceptable range.

Figure 4.4b shows the residual distribution for the P-side, along the short side of the sensor with 250 pixels. No individual sensors are shown, as they behave similarly. Instead, the sensors of the same layer have been sliced into three parts: The left plots show the residuals when the true hit lies on the lower part of the sensor, with low $u_{\text{true}} = 0$–83, the middle plots the centre ($u_{\text{true}} = 84$–167) and the right plots the upper part (high $u_{\text{true}} = 168$–250).

- If the true track lies close to the edges of the sensor, the residual distribution is asymmetric. This is expected, as DATCON hits cannot be found outside the sensor, setting an absolute limit on the residuals.
- The DATCON hits which seem to move from left to right on the three plots are probably fake hits, where DATCON could not detect the individual intersections in the HS, but still created a hit on the same sensor. The ROI size should be kept small to not include those random hits.
- The peak of the distribution is slightly shifted to the right by $\sim$20 px, meaning DATCON reconstructs true hits with a slightly too low u coordinate, or too low ϕ angle. This corresponds to about 1 mm on the sensor, or $\phi = 3.0°$–$4.1°$ on L1 and $2.3°$–$2.6°$ on L2. The cause of this offset is unknown.

Choosing the ROI size much larger would not only make the data reduction factor worse, but would also include true hits which DATCON technically wasn't able to find in its Hough space, leading to *random lucky ROIs*. The ROI padding should only account for smaller uncertainties, and not be used to increase DATCON's efficiency arbitrarily. A ROI size of ± 40 px is kept in the following.



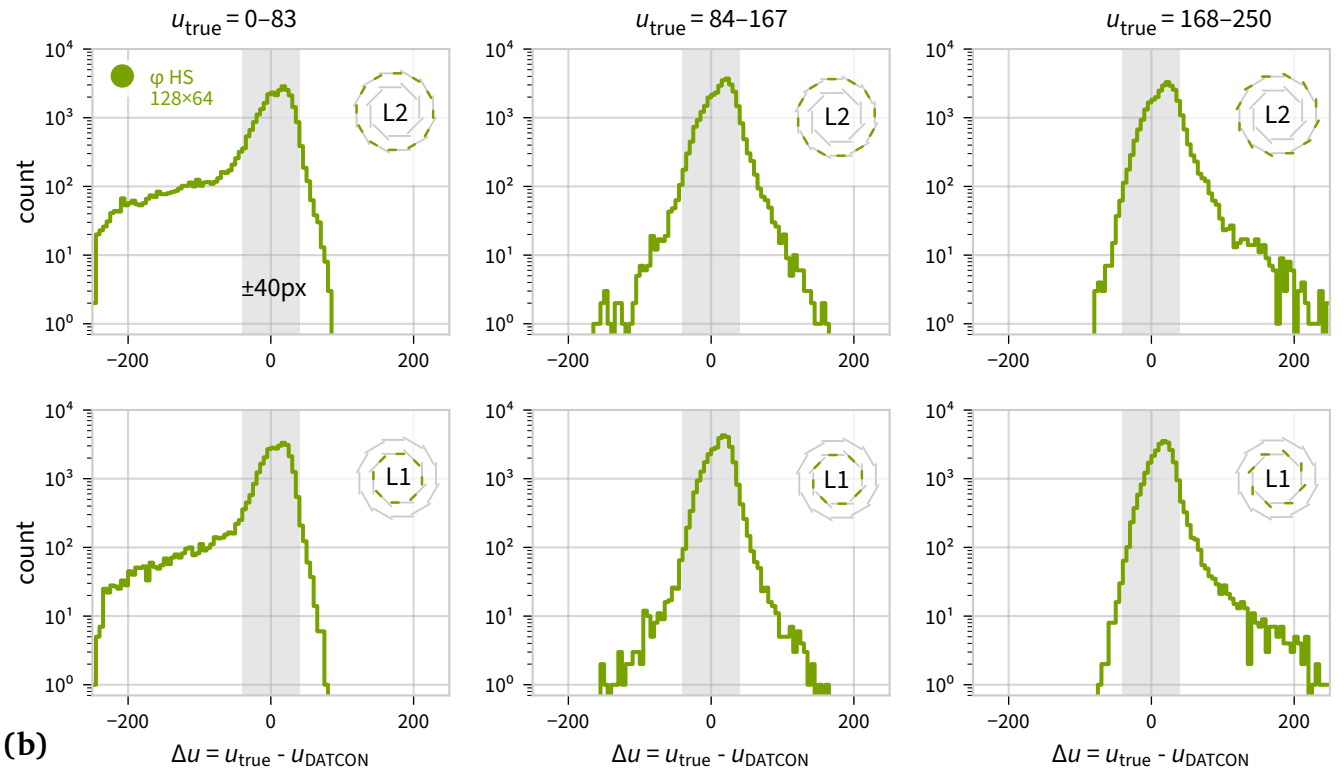


Figure 4.4 – On-sensor residuals for N and P. *The pixel positions v and u on the PXD sensor from DATCON's HS extrapolation are compared with the true hit. In gray, the nominal ROI size of ±40 px is shown around the true hit position. In* **(a)***, the four sensors in the θ projection are shown separately.* **(b)** *shows the φ projection. The φ sensors are subdivided in three sectors.*

4.5 Segmentations

Figure 4.5 compares the performance of DATCON with different segmentations. Besides the baseline of ▼ 172 segments (with a veto threshold of 2), the configurations with fewer segments ● 15 (only layer 3) and ■ 81 (only layer 6) are shown.

As ● 15 and ■ 81 are single-layer segmentations, only a veto threshold of 1 can be used. This also makes the configurations very sensitive to the geometry in that layer, as can be seen by the sharp drops in the HFE at distinct θ values. Those drops correspond to the gaps between the sensors:

layer	**sensors**	z_{gap} **[mm]**		θ_{gap} **[rad]**		**radius [mm]**
L3	2 and 1	28.79	30.91	0.935	0.900	38.99
L6	2 and 1	243.50	246.15	0.507	0.499	135.15
	3 and 2	118.50	120.62	0.851	0.842	
	4 and 3	−6.50	−4.38	1.619	1.603	
	5 and 4	−131.50	−129.38	2.343	2.334	

If a particle passes between the two sensors, no hit is recorded in that layer, so the segment ID will always be `0`. Even if the Clusterizer correctly identified a three-hit intersection, the segment veto will discard all combinations involving the resulting ROIs.

The ● 15-segment configuration has a higher HFE than ■ 81, but also a lower DRF, as more ROI combinations are kept. The ▼ 172-segment configuration has the best overall performance, and should be preferred if possible.

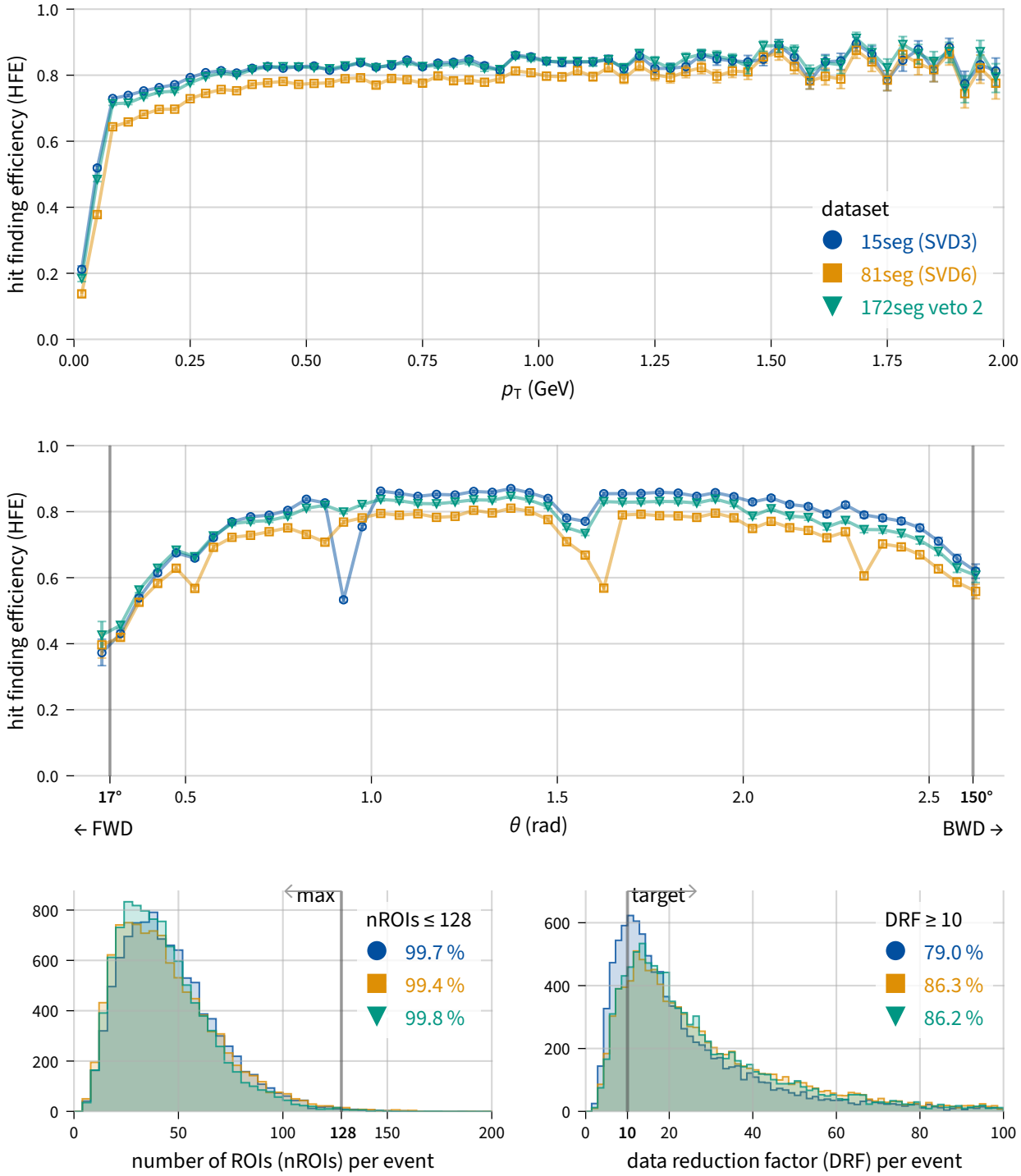


Figure 4.5 – Performance comparison of different segmentations. *For ● 15 segments and ■ 81 segments, which correspond to the SVD L3 resp. L6 only, a veto of 1 was used. For ▼ 172 segments, a veto threshold of 2 was applied. ● 15 has a higher HFE than ■ 81, but also has a lower DRF. Both ● 15 and ■ 81 have sharp drops in the HFE at θ values corresponding to the sensor gaps, which are not present in the 172 segment datasets.*

4.6 Performance of veto threshold settings

Figure 4.6 shows the performance of DATCON with 172 segments, and different settings for the segment veto.

The baseline for the comparison is the ● veto OFF dataset, which is processed without segment veto. All φ and θ 1D-ROIs are combined, and the resulting overlapping 2D ROIs are merged to larger ROIs. This results in a high HFE, but also a low DRF. Only 18.6 ± 0.4 % of the processed events have a DFE >10, the median DRF is 5.20. The number of ROIs is reduced with the 2D merging, but 18.3 % of the events will need to discard ROIs before sending them to ONSEN.

With the segment veto enabled, requiring at least *one* matching segment (■ veto 1 dataset), the number of output ROIs increases, which may appear counterintuitive: Because the veto removes some ROIs before the merging, fewer overlaps are found. Nevertheless, the DRF improves by about a factor of 3: 50.4 % of events have a DRF > 10, compared with 18.6 % without a veto.

When the number of required matching segments is increased (≥2 matching segments for ▼ veto 2, ≥3 matching segments for ◆ veto 3) the DRF increases further. The HFE drops with increasing veto because the probability that a *lucky ROI* captures a true hit by chance decreases. Put differently, without a veto, or a weak one, additional ROIs from incorrect combinations of θ and φ ROIs inflate the effective PXD detector surface and thereby collect more true hits by coincidence, which raises the HFE. This high HFE alone does not indicate a better algorithmic performance. A stronger veto provides a more faithful representation of DATCON's actual performance. However, if DRF and nROIs remain within the limits, a lower veto may be considered.

If the segment requirement is too high (e.g. ≥5 segments for a segmentation of 172), the HFE drops sharply. In these cases, even valid ROIs are removed, because a single particle track cannot produce such a signature in the SVD segments and DATCON's HS.

In a very crowded HS with many hits, clusters will usually contain contributions from many sinusoids, so the 1D ROIs created from such large clusters also contain many segments. The segment veto does not help in this case, as those 1D ROIs still pass the veto with many other ROIs. The HS thus also needs to be adapted to the expected occupancy.

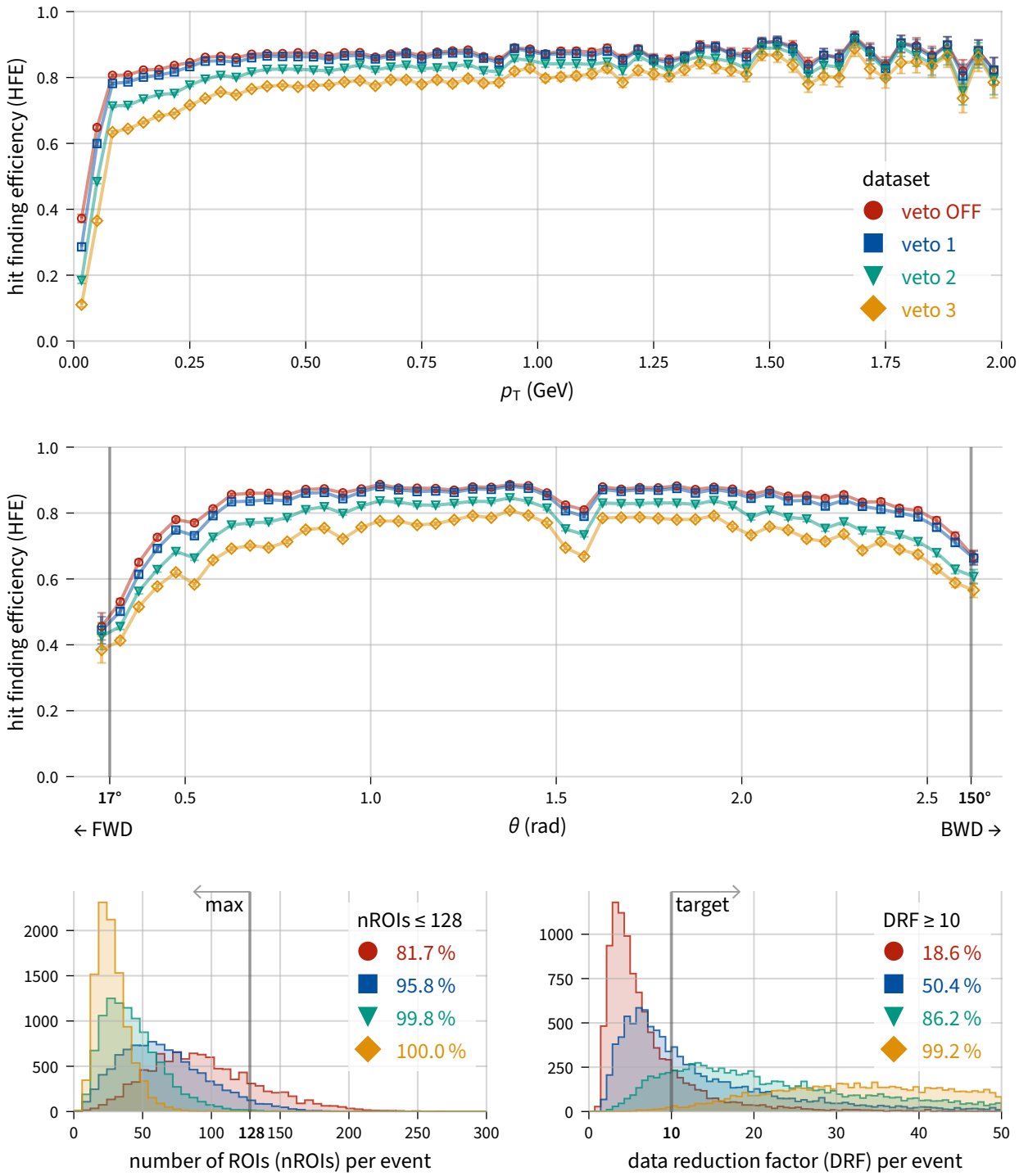


Figure 4.6 – Performance comparison of different veto settings. *The same dataset has been processed with 172 segments and different settings for the segment veto. The ● veto OFF dataset is processed without segment veto, applying 2D merging of all ROIs. The drop in HFE with increasing veto requirement (■ 1→▼ 2→◆ 3) is expected, as the chance of a random-lucky-ROI is decreased. At the same time, the number of ROIs and the DRF drastically improve.*

4.7 Comparison with HLT

Figure 4.7 shows the performance comparison of the ▶ HLT with three variants of DATCON. ◆ optimal is the best possible configuration for DATCON, with large HSs (256×128 for φ, 256×32 for θ), 172 segments with a veto of 3, and large enough buffers not to run into any limits. ■ realistic is a more conservative configuration, which can run unchanged on the hardware, as shown in *Section 4.8: Optimizations for hardware*. It uses the smaller φ and θ HSs shown the previous section (128×64 and 256×32), only uses 81 segments with a veto of 1, and uses buffers which fit the FPGA resources. ● 1D merging is the previous implementation of DATCON, with 1D merging of ROIs, no veto, and a HS of 128×64 in both projections.

- The HLT has an overall better performance than DATCON. Especially the lower and higher θ regions, in the forward and backward directions, seem to be problematic for DATCON. This is probably related to the straight-track approximation in the θ projections.
- Compared to DATCON, the HLT creates smaller ROIs, on average 50×50 px, which is beneficial for the DRF. The HLT achieves a very high DRF with a median of 131.6, while DATCON achieves a median DRF of 24.5 (◆ optimal), 22.4 (■ realistic) and 3.2 (● 1D merging).
- The ■ realistic implementation already achieves a sufficient DRF, while keeping the HFE at a reasonable level.

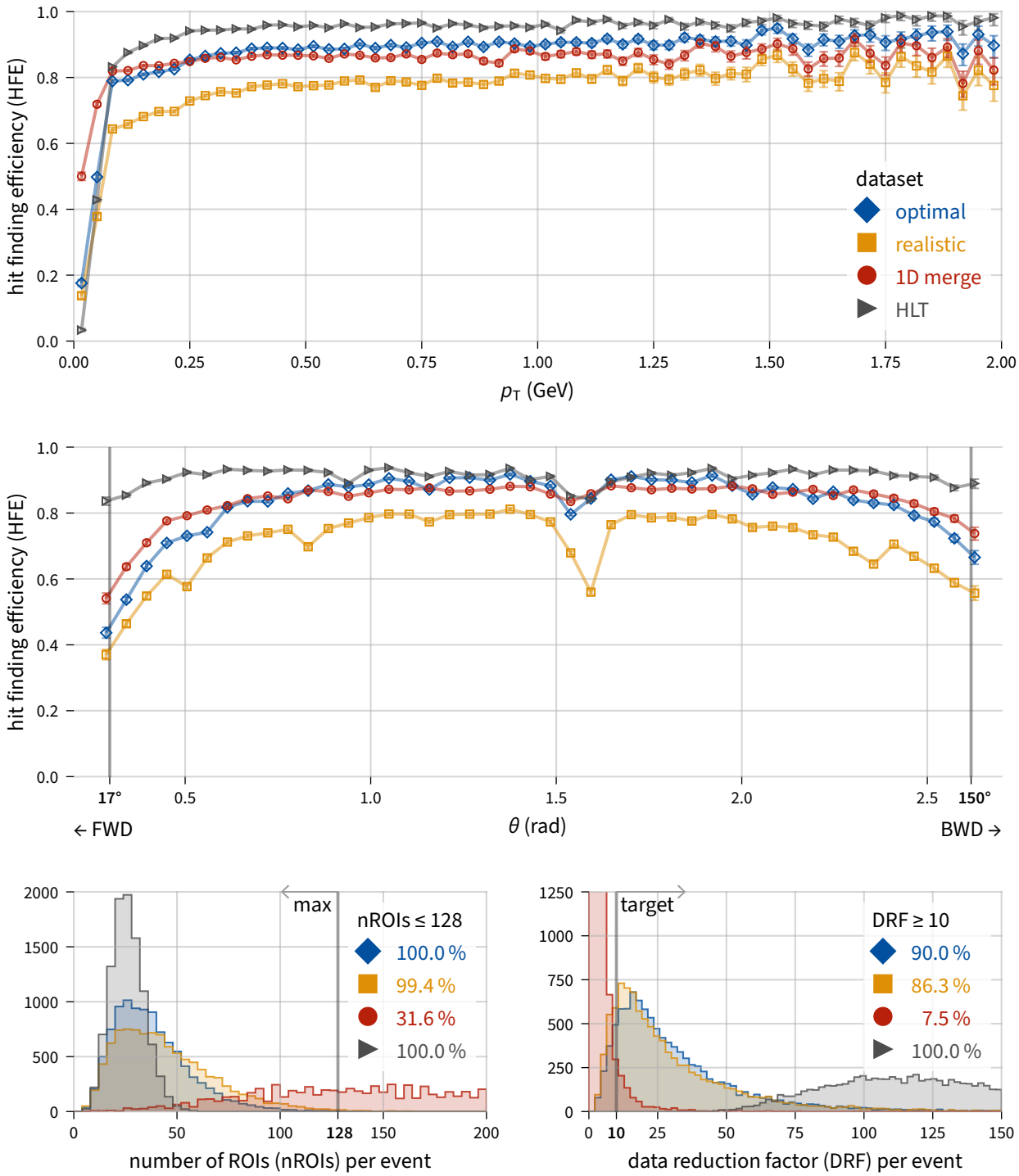


Figure 4.7 – Comparison of HLT and DATCON performance. *The performance of the ▶ HLT is compared with three different configurations of DATCON.* ◆ optimal *uses the best-possible settings for DATCON, with 172 segment, veto of 3, φ HS of 256×128, θ HS 256×32.* ■ realistic *uses 81 segments (L6 only) with a veto of 1, φ HS of 128×64, θ HS 256×32.* ● 1D merging *uses the previous implementation of DATCON, without veto, φ and θ HS of 128×64.*

4.8 Optimizations for hardware

The previous performance studies used various parameterizations for DATCON, both exploring the limits of the revised algorithm with best-possible settings, and also more restrictive parameter sets. Due to the limited resources on the FPGA, not all configurations can be realized in the firmware. The following section evaluates optimizations and limitations for DATCON's hardware.

4.8.1 FPGA resource utilization

The FPGA implementation tools report resource utilization both after the synthesis stage and the mapping stage. The post-synthesis report provides an initial estimate, but the final resource count is determined by the mapping stage, which places the design's logic onto the hardware primitives of the target FPGA. The mapping will only succeed if the design's resource requirements are below the device's capacity. While synthesis can always be completed, its estimate is often higher than the final utilization, as the mapping tools apply further logic optimizations.

Three metrics are key for DATCON's firmware implementation on the FPGAs:

- **number of slice registers** and **number of slice LUTs.** As explained in *Section 1.8: FPGAs*, each slice of the Virtex-6 Tracking-FPGA is composed of 4 LUTs and 8 registers (flip-flops (FFs)), while the Virtex-5 architecture of the Concentrator provides 4 LUTs and 4 FFs per slice [57]. Depending on the design, it is possible that a slice is not fully used, so that either the registers or the LUTs are needed. The *number of slice registers* reflects the total FF occupied, while the *number of slice LUTs* reflects the resources consumed for implementing combinatorial logic.

- **number of block RAM.** The BRAMs are used for implementing memory-intensive structures, such as the precomputed lookup tables for hit-to-HS-parameter and HS-cells-to-PXD-hit translations. They also serve as temporary storage for the segments in the HS, the 1D and 2D ROIs within the roi_combine modules, and as FIFOs for buffering and data synchronization.

The Virtex-5 XC5VLX50T FPGA, used for the Concentrators, has a total of 7200 slices (with 28 800 LUTs and 28 800 registers) and 60 BRAM blocks of 36 Kb. The Virtex-6 XC6VLX240T, which is used for the two Tracking modules, has a total of 37 680 slices (with 150 720 LUTs and 301 440 registers) and 416 BRAM blocks of 36 Kb.

In addition to the global post-synthesis and post-map reports, Xilinx ISE provides a hierarchical post-map module-level utilization report, which allows identifying the most resource-intensive modules. Those reports are analysed in the following, where available. Notably, if the mapping fails as the resource utilization is too high, no module-level report is available.

For both Virtex-5 and Virtex-6, the 36 Kb BRAMs can be configured as two independent 18 Kb BRAMs, which is done automatically by the mapping tool if needed. For the Virtex-6 FPGA, the module-level utilization map report in Xilinx ISE only lists the total number of BRAM *primitives* used per module, without specifying if those are 36 Kb (RAMB36E1) or 18 Kb (RAMB18E1). The number of actual 36 Kb BRAM *blocks* on the FPGA fabric is only given for the entire design, and not per-module. On the BRAM charts for the P and N-side Tracking firmware in this section, the individual modules are given in *primitive* counts, while their sum is scaled according to the actual 36 Kb BRAM block count for the design.

4.8.2 Concentrator

The resource utilization of the Concentrator firmware on the Virtex-5 FPGAs is shown in *Figure 4.8*. The primary tuneable parameter of the Concentrator are the sizes of the 8 storage modules (two per link, for P and N), which are used as ring buffers to delay the preprocessed SVD data. The individual storage modules, as described in *Section 3.2: Concentrator*, need to be large enough to hold multiple (partial) events, until all SVD hits of the same event have been received and processed, so the event can be sent at once to the Tracking module. In *Figure 4.8a*, the resource utilization with memories for 1024 32-bit SVD hits is shown. The storages are the main contribution to the BRAM utilization. Still, about half of the BRAMs are free. The size of the memories cannot simply be increased, as they are limited to 1k by the FPGA architecture. A restructuring of the memory organization would be required.

Additionally, the preprocessors allow for filtering of the SVD hits, to reduce the background contribution and remove the pressure from the HS for busy events. The configuration in *Figure 4.8b* shows the slice LUT utilization after replacing the *peak* noise filter with the *noise map* filter, which thresholds the SVD hits based on per-strip noise levels obtained from a previous SVD run. The LUT utilization increases significantly, but the BRAM utilization remains unchanged, as the noise map is stored as distributed RAM (LUT RAM).

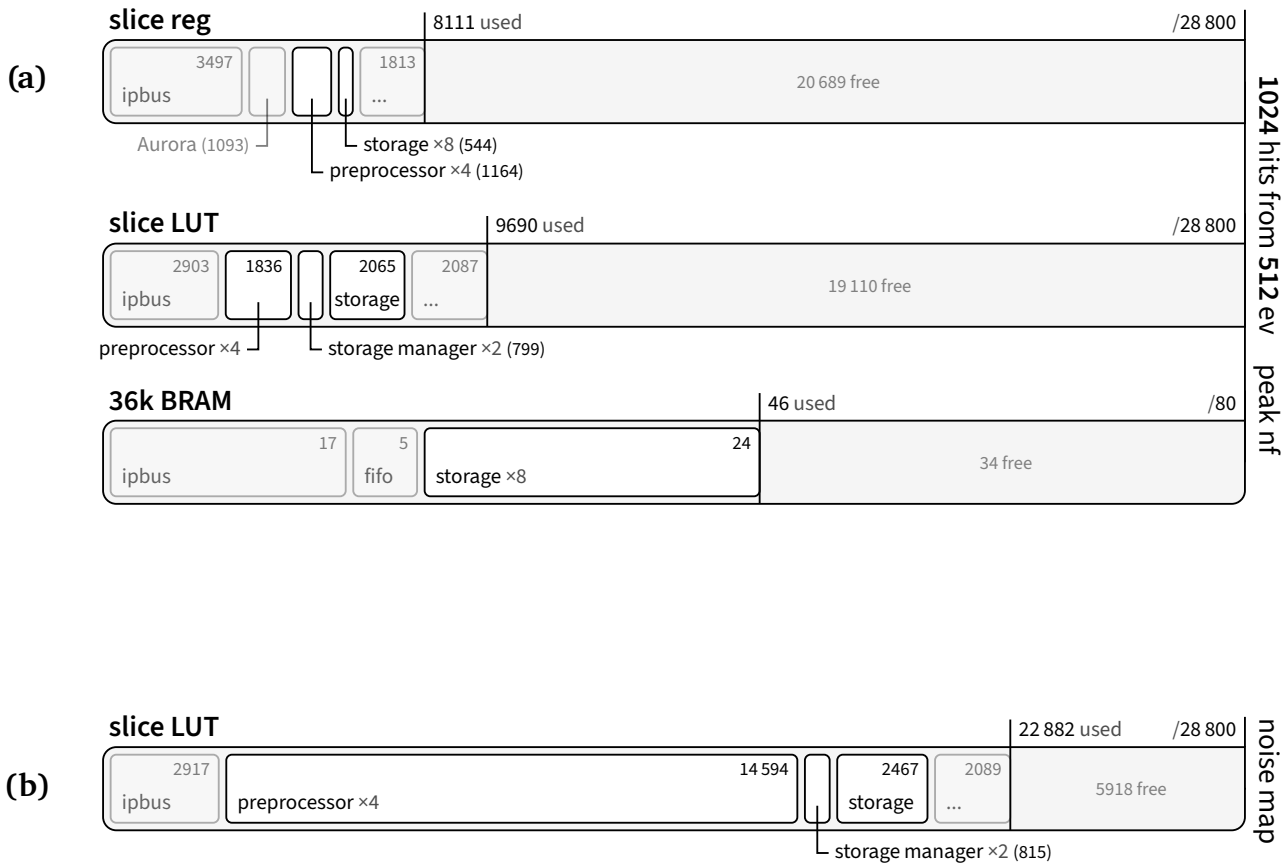


Figure 4.8 – FPGA resource utilization for the Concentrator firmware. *In* **(a)***, the base configuration of the Concentrator is shown. The depth of the* storage *modules, which buffer the asynchronous event data, can be increased, as the BRAM utilization is at about 50 %. In* **(b)***, the noise map filtering is enabled, which increases the slice LUT utilization, but leaves the BRAM utilization unchanged.*

4.8.3 Tracking

The Tracking firmware has multiple parameters which can be tuned:

- **number of segments.** The segment masks are used in the HS building and the ROI combine step, and are passed through most modules of the Tracking firmware. 15, 81 and 172 segments are considered.
- **parallelization of the HS generation.** The number of parallel levels L and sequential cycles determine a trade-off: Using more parallel column_check workers shortens the processing time for the HS construction, but increases the resource utilization. The baseline is $L = 8$.
- **resolution of the HS.** The width of the HS impacts the processing time or resource utilization, depending on L. The height of the HS impacts the segment_writer and segment_storage modules. The baseline is 128×64 cells.
- **ROI combine buffers.** On the P-side, the 2D ROI combine stores the list of each 1D φ ROI and θ ROI, and their segments. All matching combinations are additionally stored in the 2D-ROI memory. The per-sensor baseline is 2×128 for the 1D memories, and 128 for the 2D memory.

4.8.4 N-side Tracking

Figure 4.9 shows the resource utilization for the N-side Tracking for 15, 81 and 172 segments, with a HS of 128×64 cells and 8 levels.

The N-side Tracking receives the hit data from the 6 connected Concentrators in the upper shelf through the backplane connections. The data are stored in the receiver storages on BRAM. They are configured to hold 2×8192 words from 32 events for each link, before being processed. The BRAM utilization is low, so this number can be increased if needed.

The most impactful parameter is the number of segments. The column_check is the most sensitive module to the number of segments, as it tracks the 172-bit segment masks for each of the 64 cells in a column while iterating over the hits. This results in an increased register and LUT utilization for high segmentations. The segment_store grows in BRAM utilization, which stores the segment masks for all cells. The 8 segment_writer modules receive an entire column of segments at once, which then need to be stored one-by-one in the segment_store. This process is heavy on combinatorial logic (LUTs).

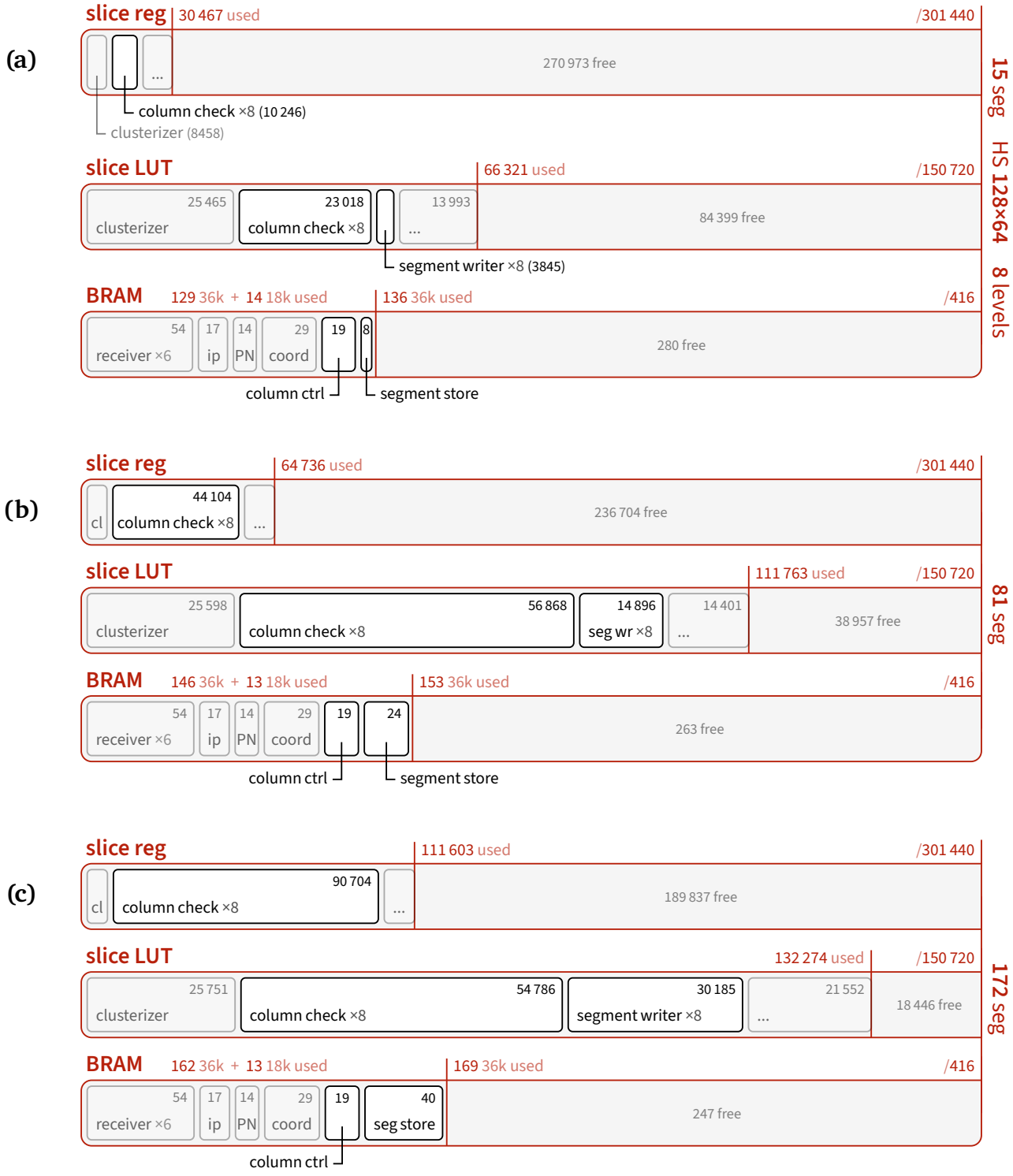


Figure 4.9 – FPGA resource utilization for N-Tracking with different segmentations. *Shown is the base configuration of a 128×64 HS with 8 parallel levels, with* **(a)** *15 segments,* **(b)** *81 and* **(c)** *172 segments. The register and LUT utilization grows with the number of segments, especially due to the* column_check *modules which handle the segment masks per cell. Storing the segments impacts the LUT utilization (*segment_writer*) and the BRAM utilization (*segment_store*).*

4.8.5 P-side Tracking

The resource utilization of the P-side Tracking module can be seen in *Figure 4.10*, for 15, 81 and 172 segments, with a HS of 128×64 cells and $L = 1$ level.

Compared with the N-side Tracking firmware, the P-side Tracking firmware has a higher resource utilization:

- The SVD coordinate lookup (coord) uses about twice as many BRAM blocks than for θ (68 instead of 29), because the number of ladders in the φ-projections require more addressable entities.
- The P-side Tracking module has one additional connected Concentrator (7 instead of 6), which increases the BRAM utilization of the receiver storages.

Additionally, the P-side Tracking module fulfils additional tasks:

- The P-side receives the 1D θ ROIs, with associated segment masks, from the N-side module. Those need to be stored before being combined (theta_roi).
- The 40 roi_combine modules require three different storages, which are implemented as BRAMs: The first two store the list of 1D φ ROIs and θ ROIs with their segment masks. Up to 128 individual ROIs are stored before being combined. If the combination passes the segment veto, the resulting 2D ROI is stored in a third memory, holding up to 128 combinations. Finally, the 2D ROIs are read from that memory pairwise, and compared with each other to merge overlapping ROIs.

Given the inherently high resource utilization, the parallelization of the HS generation has been disabled by setting $L = 1$. This change eliminates the duplicated column_check and segment_writer modules, thereby releasing resources for the roi_combine modules. Still, the mapping stage only succeeds for configurations of 15 and 81 segments. For 172 segments, the implementation results in an *overmapping* of the BRAM resources.

As both the P and the N-side need to share the same segmentation, the ■ realistic configuration with 81 segments from the previous section can in fact be implemented as-is on the FPGAs.

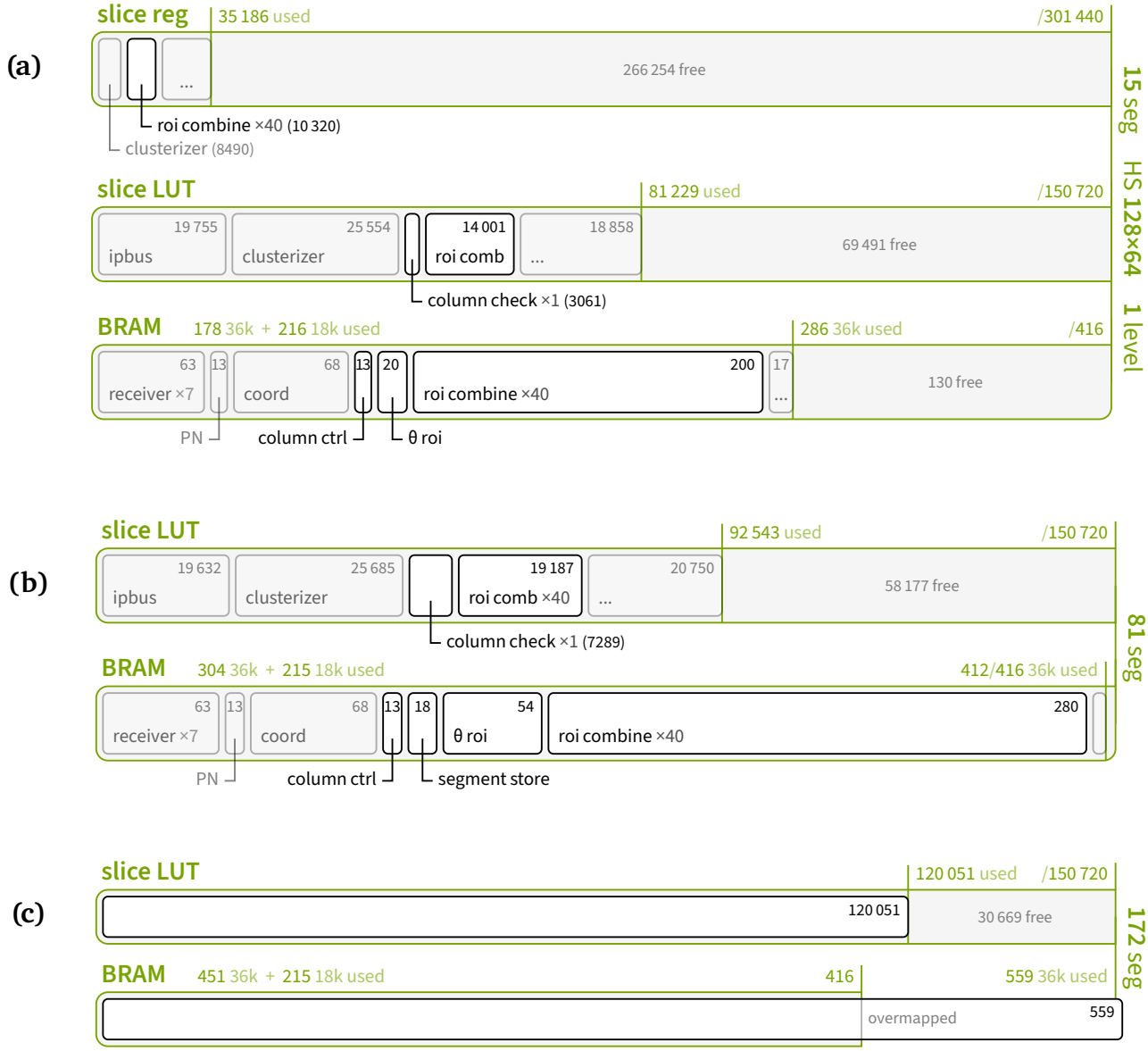


Figure 4.10 – FPGA resource utilization for P-Tracking with different segmentations. **(a)** *shows the utilization with 15 segments,* **(b)** *with 81 and* **(c)** *with 172 segments. The P-side Tracking module also needs to process the ROI combinations and the segment veto (*roi_combine*), which rely heavily on BRAMs. Additionally, the P-side module has 7 connected Concentrator modules (*receiver*), instead of 6 for the N-side, and requires larger ROMs for the coordinate lookup (*coord*). In the shown configuration, the parallelization of the HS generation has been turned off. The slight increase in* segment_store *BRAM utilization for multiple parallel levels would already exceed the available BRAMs with ≥ 81 segments. For the 172-segment configuration, no module-level utilization is available, due to overmapped BRAMs. The register utilization doesn't change significantly with the number of segments and is only shown in* **(a)**.

Chapter 5

Conclusions and outlook

In this thesis, the main weaknesses of the DATCON online tracking system were addressed, which prevent a meaningful operation during the physics runs of the Belle II experiment. The previous firmware version produced a large number of regions of interest (ROIs) per event, and an insufficient data reduction factor (DRF) prevented the effective use of its output. A 3D segmentation of the Silicon Vertex Detector (SVD) was developed, which encodes the spatial origin of hits. This allows for the implementation of a veto, which rejects physically implausible combinations, eliminating fake ROIs intrinsic to the separate processing of the P- and N-strips. Combined with an optimized Hough space (HS), DATCON now meets its design specifications for a DRF >10, necessary for the future high-luminosity operation of SuperKEKB and Belle II.

While the hit finding efficiency (HFE) maintains the high level of the previous implementation of DATCON, it stays lower than that of the High-Level Trigger (HLT), especially for the polar angles θ at the border of the detector acceptance. DATCON's assumptions about the trajectory of the particles are likely too simplistic for these regions. Further algorithmic enhancements like an improved extrapolation model or a dedicated HS parameterization could address this inefficiency.

The presented performance evaluations, conducted on simulated events, demonstrate the impact of these improvements. While essential for development, the simulations cannot fully replicate the conditions inside Belle II, both including detector noise and beam-induced backgrounds, and also the interplay with the rest of the data acquisition (DAQ) system. The next steps involve the deployment of the presented firmware configuration, and the fine-tuning of the parameters for real collision data, as is planned for the end of 2025.

Appendix A

List of figures

Appendix B

Bibliography

[1] M. Schnell:
Development of an FPGA-based Data Reduction System for the Belle II DEPFET Pixel Detector
PhD thesis: Rheinische Friedrich-Wilhelms-Universität Bonn (2015). HDL: 20.500.11811/6499

[2] B. Deschamps:
Development and Deployment of DATCON, an FPGA-based Data Reduction System for the Belle II Pixel Detector
PhD thesis: Rheinische Friedrich-Wilhelms-Universität Bonn (2022).
HDL: 20.500.11811/10353

[3] I. Neutelings:
Standard Model – TikZ
URL: tikz.net/sm_particles

[4] N. Cabibbo:
Unitary Symmetry and Leptonic Decays
Phys. Rev. Lett. 10 (1963) 531. DOI: 10.1103/PhysRevLett.10.531

[5] M. Kobayashi and T. Maskawa:
***CP* Violation in the Renormalizable Theory of Weak Interaction**
Prog. Theor. Phys. 49 (1973) 652. DOI: 10.1143/PTP.49.652

[6] Belle II Collaboration: I. Adachi *et al.*:
Measurement of the branching fraction, polarization, and time-dependent *CP* asymmetry in $B^0 \to \rho^+\rho^-$ decays and constraint on the CKM angle ϕ_2
Phys. Rev. D 111 (2025) 092001. DOI: 10.1103/PhysRevD.111.092001

[7] Belle and Belle II Collaborations: I. Adachi *et al.*:
Determination of the CKM angle ϕ_3 from a combination of Belle and Belle II results
JHEP 10 (2024) 143. DOI: 10.1007/JHEP10(2024)143

[8] Particle Data Group: S. Navas *et al.*:
Review of Particle Physics
Phys. Rev. D 110 (2024) 030001. DOI: 10.1103/PhysRevD.110.030001

[9] CKMfitter Group: J. Charles *et al.*:
CP violation and the CKM matrix: Assessing the impact of the asymmetric B factories
Eur. Phys. J. C 41 (2005) 1. DOI: 10.1140/epjc/s2005-02169-1

[10] J. H. Christenson, J. W. Cronin, V. L. Fitch and R. Turlay:
Evidence for the 2π Decay of the K_2^0 Meson
Phys. Rev. Lett. 13 (1964) 138. DOI: 10.1103/PhysRevLett.13.138

[11] Belle Collaboration: K. Abe *et al.*:
Observation of large *CP* violation in the neutral B meson system
Phys. Rev. Lett. 87 (2001) 091802. DOI: 10.1103/PhysRevLett.87.091802

[12] BaBar Collaboration: B. Aubert *et al.*:
Observation of *CP* violation in the B^0 meson system
Phys. Rev. Lett. 87 (2001) 091801. DOI: 10.1103/PhysRevLett.87.091801

[13] LHCb Collaboration: R. Aaij *et al.*:
Observation of *CP* Violation in Charm Decays
Phys. Rev. Lett. 122 (2019) 211803. DOI: 10.1103/PhysRevLett.122.211803

[14] LHCb Collaboration: R. Aaij *et al.*:
Observation of charge-parity symmetry breaking in baryon decays
2025. ARXIV: 2503.16954

[15] SuperKEKB Collaboration: K. Akai, K. Furukawa and H. Koiso:
SuperKEKB collider
Nucl. Instrum. Meth. A 907 (2018) 188. DOI: 10.1016/j.nima.2018.08.017

[16] Y. Ohnishi *et al.*:
Accelerator design at SuperKEKB
PTEP 2013 (2013) 03A011. DOI: 10.1093/ptep/pts083

[17] SuperB Collaboration: M. Bona *et al.*:
SuperB: A High-Luminosity Asymmetric e^+e^- Super Flavor Factory. Conceptual Design Report
2007. ARXIV: 0709.0451

[18] **SuperKEKB/Belle II Complete 2024 Operations**
2025. URL: www2.kek.jp/ipns/en/news/7015

[19] Belle II Upgrades Working Group: H. Aihara *et al.*:
The Belle II Detector Upgrades Framework Conceptual Design Report
2024. ARXIV: 2406.19421

[20] N. Ohuchi *et al.*:
Final-focus Superconducting Magnets for SuperKEKB
9th International Particle Accelerator Conference (2018).
DOI: 10.18429/JACoW-IPAC2018-TUZGBE2

[21] Belle II Collaboration:
Luminosity website
URL: www.belle2.org/research/luminosity

[22] Belle II Collaboration: F. Bernlochner *et al.*:
The Belle II Experiment at SuperKEKB – Input to the European Particle Physics Strategy
2025. ARXIV: 2503.24155

[23] Wikimedia Commons:
KEKB
2020. URL: commons.wikimedia.org/?oldid=455849591.
CC0 1.0 (creativecommons.org/publicdomain/zero/1.0)

[24] E. Kou *et al.*:
The Belle II Physics Book
PTEP 2019 (2019) 123C01. DOI: 10.1093/ptep/ptz106.
Open Access, CC BY 4.0 (creativecommons.org/licenses/by/4.0)

[25] Belle Collaboration: A. Abashian *et al.*:
The Belle detector
Nucl. Instrum. Meth. A479 (2002) 117. DOI: 10.1016/S0168-9002(01)02013-7

[26] Belle II DEPFET and PXD Collaborations: H. Ye *et al.*:
Commissioning and performance of the Belle II pixel detector
Nucl. Instrum. Meth. A 987 (2021) 164875. DOI: 10.1016/j.nima.2020.164875

[27] Belle II DEPFET and PXD Collaborations: P. Ahlburg *et al.*:
The new and complete Belle II DEPFET pixel detector: Commissioning and previous operational experience
Nucl. Instrum. Meth. A 1068 (2024) 169763. DOI: 10.1016/j.nima.2024.169763

[28] J. Kemmer and G. Lutz:
New Detector Concepts
Nucl. Instrum. Meth. A 253 (1987) 365. DOI: 10.1016/0168-9002(87)90518-3

[29] R. H. Richter *et al.*:
Design and technology of DEPFET pixel sensors for linear collider applications
Nucl. Instrum. Meth. A 511 (2003) 250. DOI: 10.1016/S0168-9002(03)01802-3

[30] D. Levit:
Search for $D^+ \to K^- K^0_S \pi^+ \pi^+ \pi^0$ at the Belle Experiment and Development of the Read-Out System for the Pixel Detector of the Belle II Experiment
PhD thesis: Technische Universität München (2019).
URN: nbn:de:bvb:91-diss-20190213-1468810-1-6

[31] D. Levit, I. Konorov, D. Greenwald and S. Paul:
FPGA based data read-out system of the Belle 2 Pixel Detector
IEEE Trans. Nucl. Sci. 62 (3 2015) 1033. DOI: 10.1109/TNS.2015.2424713

[32] Belle II Collaboration: T. Abe *et al.*:
Belle II Technical Design Report
2010. ARXIV: 1011.0352

[33] A. Moll *et al.*:
The vertex detector numbering scheme
Belle II Note 0010 (2015).
URL: indico.mpp.mpg.de/event/3237/contributions/7212/attachments/5936

[34] Belle II Germany:
Made in Germany und big in Japan: Pixel-Vertex-Detektor in Belle II-Experiment installiert
2023. URL: www.belle2.de/news-detail/made-in-germany-und-big-in-japan-pixel-vertex-detektor-in-belle-ii-experiment-installiert

[35] Belle II DEPFET and PXD Collaborations: A. Baur *et al.*:
The Belle II Pixel Vertex Detector
PoS VERTEX2023 (2024) 009. DOI: 10.22323/1.448.0009

[36] Y. Iwata *et al.*:
Optimal P-stop pattern for the N-side strip isolation of silicon microstrip detectors
IEEE Trans. Nucl. Sci. 45 (1998) 303. DOI: 10.1109/23.682398

[37] M. J. French *et al.*:
Design and results from the APV25, a deep sub-micron CMOS front-end chip for the CMS tracker
Nucl. Instrum. Meth. A 466 (2001) 359. DOI: 10.1016/S0168-9002(01)00589-7

[38] Belle II SVD Collaboration: F. Forti *et al.*:
The design, construction, operation and performance of the Belle II silicon vertex detector
JINST 17 (2022) P11042. DOI: 10.1088/1748-0221/17/11/P11042

[39] M. Nakao, T. Higuchi, R. Itoh and S. Y. Suzuki:
Data acquisition system for Belle II
JINST 5 (2010) C12004. DOI: 10.1088/1748-0221/5/12/C12004

[40] M. Nakao:
Timing distribution for the Belle II data acquistion system
JINST 7 (2012) C01028. DOI: 10.1088/1748-0221/7/01/C01028

[41] D. Sun, Z. Liu, J. Zhao and H. Xu:
Belle2Link: A Global Data Readout and Transmission for Belle II Experiment at KEK
Phys. Procedia 37 (2012) 1933. DOI: 10.1016/j.phpro.2012.01.036

[42] Q. D. Zhou *et al.*:
PCI-Express Based High-Speed Readout for the Belle II DAQ Upgrade
IEEE Trans. Nucl. Sci. 68 (2021) 1818. DOI: 10.1109/TNS.2021.3086526

[43] R. Itoh, T. Higuchi, M. Nakao, S. Y. Suzuki and S. Lee:
Data flow and high level trigger of Belle II DAQ system
IEEE Trans. Nucl. Sci. 60 (2013) 3720. DOI: 10.1109/RTC.2012.6418174

[44] R. Itoh *et al.*:
Improved HLT Framework for Belle II Experiment
EPJ Web Conf. 295 (2024) 02016. DOI: 10.1051/epjconf/202429502016

[45] M. T. Prim *et al.*:
Design and Performance of the Belle II High Level Trigger
PoS ICHEP2020 (2021) 769. DOI: 10.22323/1.390.0769

[46] Belle II Framework Software Group: T. Kuhr *et al.*:
The Belle II Core Software
Comput. Softw. Big Sci. 3 (2019) 1. DOI: 10.1007/s41781-018-0017-9

[47] Belle II Collaboration:
Belle II Analysis Software Framework (basf2)
DOI: 10.5281/zenodo.5574115

[48] Belle II Tracking Group: V. Bertacchi *et al.*:
Track finding at Belle II
Comput. Phys. Commun. 259 (2021) 107610. DOI: 10.1016/j.cpc.2020.107610

[49] Xilinx:
Virtex-6 Family Overview (DS150)
v5.5 August 20, 2015. URL: docs.amd.com/v/u/en-US/ds150

[50] Xilinx:
LogiCORE IP Aurora 8B/10B (DS637)
v5.3 January 18, 2012. URL: docs.amd.com/v/u/en-US/aurora_8b10b_ds637

[51] C. Ghabrous Larrea *et al.*:
IPbus: a flexible Ethernet-based control system for xTCA hardware
JINST 10 (2015) C02019. DOI: 10.1088/1748-0221/10/02/C02019

[52] The cocotb developers:
cocotb: Coroutine Based Cosimulation Testbench Environment
URL: www.cocotb.org

[53] R. E. Kálmán:
A New Approach to Linear Filtering and Prediction Problems
J. Fluids Eng. 82 (1960) 35. DOI: 10.1115/1.3662552

[54] R. Frühwirth:
Application of Kalman filtering to track and vertex fitting
Nucl. Instrum. Meth. A 262 (1987) 444. DOI: 10.1016/0168-9002(87)90887-4

[55] P. V. C. Hough:
Methods and means for recognizing complex patterns
US Patent US3069654A (1962). URL: patents.google.com/patent/US3069654A/en

[56] R. O. Duda and P. E. Hart:
Use of the Hough transformation to detect lines and curves in pictures
Commun. ACM 15 (1972) 11. DOI: 10.1145/361237.361242

[57] Xilinx:
Virtex-5 Family Overview (DS100)
v5.1 August 21, 2015. URL: docs.amd.com/v/u/en-US/ds100

[58] D. J. Lange:
The EvtGen particle decay simulation package
Nucl. Instrum. Meth. A462 (2001) 152. DOI: 10.1016/S0168-9002(01)00089-4

[59] GEANT4 Collaboration: S. Agostinelli *et al.*:
GEANT4: A simulation toolkit
Nucl.Instrum.Meth. A506 (2003) 250. DOI: 10.1016/S0168-9002(03)01368-8

[60] F. Bernlochner, B. Deschamps, J. Dingfelder, C. Marinas and C. Wessel:
Online Data Reduction for the Belle II Experiment using DATCON
EPJ Web Conf. 150 (2017) 00014. DOI: 10.1051/epjconf/201715000014

[61] T. Geßler *et al.*:
The ONSEN Data Reduction System for the Belle II Pixel Detector
IEEE Transactions on Nuclear Science 62 (2015) 1149. DOI: 10.1109/TNS.2015.2414713

[62] T. Higuchi *et al.*:
Development of a PCI based data acquisition platform for high intensity accelerator experiments
eConf C0303241 (2003) TUGT004. ARXIV: hep-ex/0305088

Appendix C

Acronyms

AMC Advanced Mezzanine Card. *Part of the PICMG MicroTCA specification. A PCB, often containing an FPGA, which is plugged into a MicroTCA Carrier.*

APV25 Analogue Pipeline Voltage mode, 0.25µm process. *SVD readout chips, directly mounted on the DSSDs [37].*

ARICH Aerogel Ring-Imaging Cherenkov Detector. *PID system in the forward endcap.*

ASIC application-specific integrated circuit

b2link *Unified hardware and firmware for detector readout [41].*

basf2 Belle II Analysis Software Framework. *[46, 47]*

BRAM block RAM. *Hardware primitives in the FPGA fabric, which can be used as FIFO, RAM or ROM.*

CDC Central Drift Chamber

CKM Cabibbo-Kobayashi-Maskawa. *[5]*

CLB configurable logic block

COPPER Common Pipeline Platform for Electronics Readout. *Common DAQ platform for all subdetectors (except PXD) [62].*

DAQ data acquisition. *Readout and data handling of the detector.*

DCD Drain Current Digitizer. *ASIC which digitizes the current from the DEPFET pixels, placed next to DEPFET matrix.*

DEPFET Depleted P-channel Field Effect Transistor

DFS depth-first search

DHC Data Handling Concentrator. *Part of the DHH, combines data of all 5 DHEs and sends data to ONSEN.*

DHE Data Handling Engine. *Part of the DHH, receives data from one PXD module / DHP.*

DHH Data Handling Hybrid. *Backend readout system for the PXD, located at ToB.*

DHI Data Handling Insulator. *Part of the DHH, sends control signals to the five DHEs.*

DHP Data Handling Processor. *ASIC placed on the PXD module, performing filtering on the DCD data.*

DRF data reduction factor

DSP Digital Signal Processing

DSSD double-sided strip detector. *Technology used in the SVD. Strips on both sides of the sensor allows for 2D position measurement.*

E-Hut Electronics Hut. *Three-floor server room next to the Belle II detector. The DATCON rack can be found here, ground floor on the right. Say hi from me!*

ECL Electromagnetic Calorimeter

FADC Flash Analog-to-Digital Converter. *processes analogue APV25 data with zero suppression. Passes data to FTB.*

FET field-effect transistor. *Type of transistor which uses an electric field to control the flow of current.*

FF flip-flop. *Basic storage element in electric circuits, storing a single bit.*

FPGA field-programmable gate array

FR fake rate

FSM finite-state machine. *Control structure where behaviour is described by states and transitions.*

FTB FINESSE Transmitter Board. *Attached to FADC, sends data to DATCON and COPPER/PCIe40. FINESSE stands for Front-end Instrumentation Entity for Subdetector Specific Electronics.*

FTSW Frontend Timing Switches. *FPGA based distribution system for the trigger signals at the Belle II DAQ [40].*

HDL hardware description language. *A programming language to describe hardware circuits.*

HFE hit finding efficiency

HLT High-Level Trigger. *Part of the DAQ chain [43].*

HS Hough space

IC integrated circuit

IOC input-output controller.

IP interaction point. *The point where the two beams collide, at the centre of the Belle II detector.*

IPMI Intelligent Platform Management Interface. *Standardized interface for monitoring and managing hardware.*

JTAG *IEEE Standard 1149.1. Commonly refers to the serial communication to an IC / FPGA.*

KLM K_L^0 and μ detector

LS1 Long Shutdown 1. *Shutdown period with upgrades to the accelerator and detector, especially the installation of PXD2.*

LS2 Long Shutdown 2. *Shutdown period around 2032, with upgrades to increase the luminosity.*

LUT look-up table. *Hardware primitives on the FPGA. Pre-defined table which maps inputs to outputs, usually a LUT6 with 6→1 bit.*

MCH MicroTCA Carrier Hub. *Switch and management controller of the MicroTCA shelf.*

ONSEN Online Selection Node. *Receives and buffers PXD data, applies ROIs from DATCON and HLT.*

PCB printed circuit board

PCIe40 *Upgrade of the COPPER system with higher performance [42].*

PID Particle Identification. *Comprises the TOP and ARICH detectors at Belle II.*

PM Power Module. *Power supply for the MicroTCA shelf.*

PXD Pixel Detector

ROI region of interest

Run 1 First run of the Belle II experiment. *March 2019–June 2022.*

Run 2 Second, current run of the Belle II experiment. *since 2024.*

SM Standard Model. *...of Elementary Particle Physics*

SVD Silicon Vertex Detector

TFE track finding efficiency

ToB Top-of-Belle. *The area right above the Belle II detector. Houses electronics like the PXD power supplies and SVD FTBs.*

TOP Time-Of-Propagation counter. *Part of the PID system, measures time of propagation of Cherenkov photons in the barrel region.*

VLAN virtual LAN. *A logically separated network within a physical network.*

VXD Vertex Detector. *The two innermost silicon detectors, PXD and SVD.*

Appendix D

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Dr. Florian Bernlochner, for the opportunity to be part of his group in Bonn and to work on ACTS and on DATCON. His constantly approachable, positive and immensely encouraging attitude made it a pleasure to discover new paths I would have never thought of.

I would like to thank Prof. Dr. Jochen Dingfelder, who was always optimistic for DATCON's future, and who convinced me to accept the challenge. I am grateful for the skills and experience I gained working on this project.

I would like to thank Dr. Eckhard von Törne, who supported me throughout my thesis, and who gave me structural guidance to tame the chaos in my head and tackle the writing of this thesis.

I would like to thank my friends and colleagues from the BLUB group, with whom I shared a lot of time and countless cups of coffee, discussing physics and non-physics. Your openness, good humour and support make you the best working group one could wish for. My thanks go almost as warmly to the PXD group in the SiLab, who also had great coffee, and who welcomed me into their hardware world with open arms.

Kedves Anyu és Apu, ti biztos többet gondolkoztatok erről a fizikáról, mint én. Köszönöm, hogy mindig támogattatok engem, finomságokkal és szeretettel.

Merci Tanja d'avoir toujours été là pour moi, pour ton écoute attentive de mes explications sur la physique et pour ton soutien. Tu me connais parfois mieux que moi-même.