Dissertation
submitted to the
Combined Faculties of the Natural Sciences and Mathematics
of the Ruperto-Carola-University of Heidelberg, Germany
for the degree of
Doctor of Natural Sciences

put forward by

Sebastian Klewin (M.Sc. Physics)
born in Friedrichshafen, Germany

Oral examination: April 17$^{\text{th}}$, 2019

Development of the FPGA-based Raw Data Preprocessor for the TPC Readout Upgrade in ALICE

Referees: Prof. Dr. Johanna Stachel
Prof. Dr. Peter Fischer

Entwicklung des FPGA-basierten Rohdaten-Vorprozessors für die Aufrüstung der TPC Ausleseeinheiten bei ALICE

ALICE ist eines der vier großen Experimente am Large Hadron Collider (LHC). Es ist das Schwerionenexperiment und untersucht daher in erster Linie das Quark–Gluon-Plasma. Um sich auf die Bedingungen von Blei-Blei-Kollisionen mit einer Rate von 50 kHz am LHC nach der Umrüstphase 2 (2018–2021) vorzubereiten, wird ein umfangreiches Aufrüst-Programm durchgeführt. Ziel ist es, dass die Zeitprojektionskammer kontinuierlich ausgelesen werden können. Die enorme Datenrate von 3,7 TB/s, die durch den aufgerüsteten Detektor erzeugt wird, muss bereits während der Datennahme um den Faktor 60 reduziert werden. Andernfalls würde das Datenvolumen die voraussichtlich verfügbare Bandbreite und Speicherkapazität überschreiten.

In dieser Arbeit wurde ein Online-Cluster-Finder (CF) für FPGAs entwickelt und implementiert, der das gesamte Datenvolumen in Echtzeit bereits während der Detektorauslese verarbeitet. Dies ist der erste Schritt in einer ganzen Reihe von Datenreduktionsschritten, welcher für sich genommen bereits einen Kompressionsfaktor von etwa 5 erreicht, indem nur physikalisch relevante Informationen behalten werden und ein geeigneteres Datenformat verwendet wird. Zusätzlich zum CF wurde auch die gesamte Datenaufbereitungskette konzipiert und implementiert. Diese dekodiert den Eingangsdatenstrom, sortiert die einzelnen Kanäle so um, dass eine Suche von Ladungsclustern überhaupt erst möglich wird und korrigiert die Detektoreffekte in den Eingangssignalen. Alle implementierten Module wurden ausführlich simuliert, um ihre korrekte Funktionalität nachzuweisen. Damit wurde die gesamte Datenverarbeitungskette innerhalb des FPGAs vorbereitet und validiert.

Development of the FPGA-based Raw Data Preprocessor for the TPC Readout Upgrade in ALICE

ALICE is one of the four major experiments at the Large Hadron Collider (LHC). It is the dedicated heavy-ion experiment and therefore primarily examines the Quark–Gluon Plasma. In order to prepare for the running conditions of 50 kHz lead-lead interactions at the LHC after the Long Shutdown 2 (2018–2021), an extensive upgrade program is carried out. The goal of the upgrade is a continuous readout of the TPC without the need of a trigger. It is essential to reduce the enormous data rate of 3.7 TB/s, generated by the upgraded detector, already during the data taking by a factor of about 60. Otherwise the data volume would exceed the expected available bandwidth and storage capabilities.

In this thesis, an online Cluster Finder (CF) was developed and implemented for FPGAs which processes the whole data volume in real-time during the read out. This is the first step in the data reduction sequence which achieves already a factor of about 5 by keeping only physically relevant information and making use of a better suited data format. In addition to the CF, also the whole data preparation chain was designed and implemented to decode the input data stream, to resort the individual channels to allow for cluster finding and to correct the detector effects in the input signals. All modules which were implemented were extensively simulated to verify their proper functionality. With this, the complete processing chain within the FPGAs was prepared and validated.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

A few microseconds after the formation of the early universe, it is assumed that the medium was in a very high temperature and density state [1]. Under such conditions, matter is expected to exist in a state called Quark–Gluon Plasma (QGP), where the quarks and gluons are deconfined [2]. The only known way to generate this state of matter in a laboratory in order to study it is to collide heavy nuclei at ultra-relativistic energies. One of the few places where this is possible is the Large Hadron Collider (LHC) at CERN (Conseil Européen pour la Recherche Nucléaire, European Organisation for Nuclear Research) [3].

During the first years of LHC operation, the ALICE (A Large Ion Collider Experiment) collaboration was able to confirm the basic picture of the creation of a strongly interacting matter at values of temperatures and densities never seen before in a heavy-ion collision [3]. In order to continue the studies of the QGP by examining its properties such as viscosity, transport coefficients or the temperature evolution, ALICE will focus on rare probes after the Long Shutdown (LS) 2. These rare probes are e.g. heavy-flavour particles and their coupling to the medium. By measuring the azimuthal anisotropy for charmed mesons and baryons as well as beauty particles one can draw conclusions whether the heavy quarks thermalised in the system and thus participated in the collective flow. Another probe is the measurement of jets and their correlation with other probes. The properties of the QGP can be accessed by investigating the energy loss of hard scattered partons in the strongly interacting medium. Measuring the production of different quarkonium states down to very low transverse momentum will help to understand the underlying mechanisms of the formation of those states. The bulk properties and the space–time evolution of the hot and dense medium can be accessed by measurements of low-mass dileptons. It is possible to detect the electromagnetic radiation which is produced during all stages of the heavy-ion collision either as a real photon or a dilepton pair. Since they do not interact strongly with the medium, they carry information about the entire evolution of the system [3]. The LS 2, which gives the experiments the opportunity to upgrade their detectors, started in December 2018 and will end in early 2021 [4].

The excellent tracking performance even in a high-multiplicity environment, as well as the Particle Identification (PID) capabilities over a large momentum range due to the individual detector systems of ALICE are unique at the LHC. However, for many of the physics topics a high event statistics is required for precision measurements. Since many of

the anticipated measurements require complex probes at very low transverse momentum on which a traditional triggering approach is not applicable to enhance the statistics, the ALICE experiment will undergo a high-rate detector upgrade. After this upgrade, the experiment will be able to examine practically all heavy-ion collisions delivered by the LHC at a rate of 50 kHz. This is a very different approach compared to the current readout strategy and the one of the other LHC experiments, where possibly interesting events are selected by means of a trigger [3]. With the detector upgrade, ALICE will improve its low-momentum vertexing and tracking capabilities to be able to measure also complex probes at low transverse momentum. The data taking rate will be significantly increased respectively developed into a continuous one without the need of a dedicated trigger. Otherwise, the performance especially the identification of charged particles will be preserved [3].

The upgrade program for the ALICE Time-Projection Chamber (TPC) consist of a new detector readout, including new Readout Chambers (ROCs) based on Gas Electron Multiplier (GEM) technology and a completely new Front-End Electronics (FEE), allowing for a continuous data taking. The continuous readout results in a huge data volume of 3.7 TB/s which must be reduced already during the read out by a factor of about 60. Otherwise it would exceed the expected available bandwidth and storage capabilities of the experiment [5]. The reduction will be done by reconstructing the detector signals online and writing only already processed data to the permanent storage instead of raw detector signals. An essential part of the online reconstruction is the cluster finding, where charge clusters have to be found in the data stream of the TPC before a tracking can be applied. The development of this Cluster Finder (CF) which will run on Field Programmable Gate Arrays (FPGAs) as part of the readout chain was the main topic of this thesis, together with all necessary data preparation steps that must be applied beforehand.

The thesis is divided into seven chapters. The remaining chapter 1 introduces the LHC and provides a short description of the ALICE experiment, together with a few upgrade examples of the detectors. Chapter 2 is dedicated to the description of the TPC and a more detailed explanation of the individual upgrade tasks for this detector to achieve the continuous readout, together with an estimated timeline. The following chapter 3 explains the readout chain of the upgraded TPC with its individual components and presents the reasons for a change in the readout strategy with respect to the one described in the Technical Design Report (TDR) for the upgrade [5]. This is followed by a description of the general design concept of the Firmware (FW) for the Common Readout Unit (CRU) in chapter 4. This card will be used to receive the data from the FEE and is responsible for the first preprocessing steps in the readout chain. Chapter 5 contains the description of the individual modules which are needed in the FW to successfully reconstruct 2-dimensional charge clusters in the FPGA. This includes all the data preparation steps like decoding, sorting and a Baseline Correction (BLC), as well as the peak finding, calculation of the cluster properties and concludes with the overall resource consumption of the logic in the chip. In chapter 6, the validation of the previously described modules is presented in simulation and, whenever possible, in addition in a real readout system which was used during a test beam campaign in 2017. The CF, meaning the peak finding and the calculation of the cluster properties, was also implemented in software which is described in the last part of this chapter. The final chapter 7 summarises the achievements of this thesis and gives a short outlook to possible future developments.

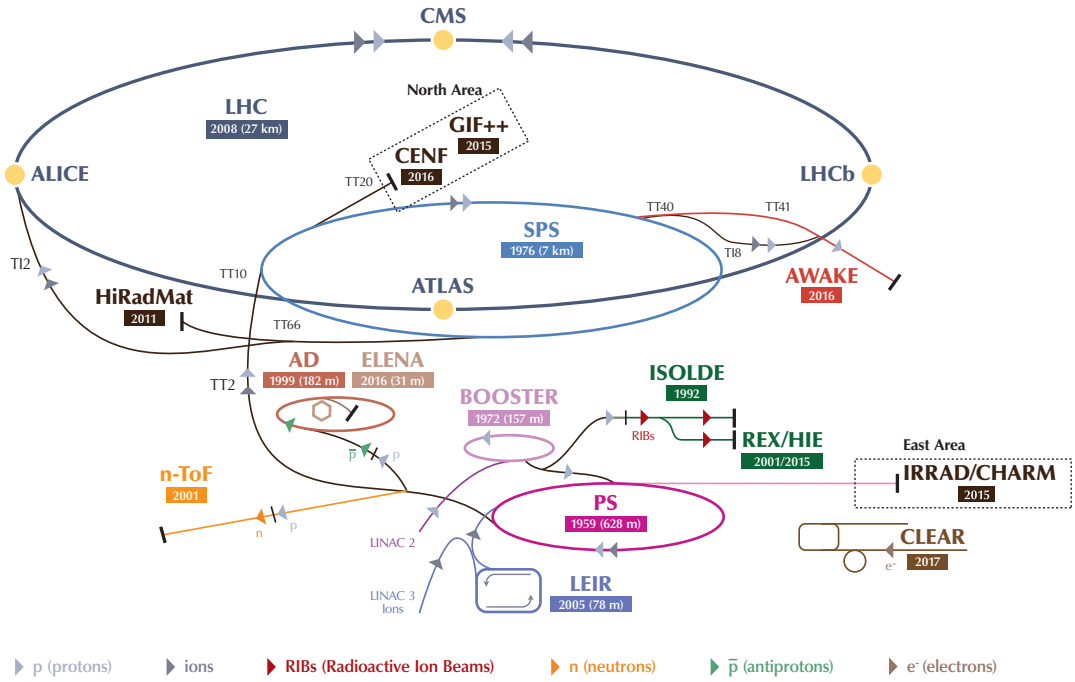


Figure 1.1: The CERN accelerator complex. The LHC is the last step in a cascade of accelerators, starting with the LINAC 2 for protons and LINAC 3 for ions. Protons are further accelerated by the PSB while the ions go to the LEIR. Both continue then with the PS and the SPS before finally reaching the LHC, taken from [7].

1.1 The Large Hadron Collider

ALICE is one of the four major experiments of the LHC. It is located at CERN near Geneva, Switzerland. Founded in 1954, CERN has been pursuing its mission to enable research in fundamental physics and to push the frontiers of science and technology. The LHC is the worlds largest collider and reaches the highest energies ever achieved of 13 TeV in proton-proton (pp) and 5.02 TeV/nucleon in lead-lead (PbPb) collisions. It is a superconducting hadron accelerator and collider and was installed in the already existing 26.7 km tunnel of the previous Large Electron Positron (LEP) machine. The LEP tunnel lies between 45 m and 170 m below the surface and its plane has an inclination of 1.4 %, sloping towards the lake Geneva. The *ring* consist actually of eight straight sections and eight arcs in which 1 232 superconducting dipole magnets are build, providing a nominal magnetic field of up to 8.33 T to keep the beams on track [6]. The LHC is the last piece in a series of successive accelerators, each increasing the energy of the particles further. This accelerator chain is shown in figure 1.1.

Protons start from a simple hydrogen gas bottle after which an electric field is used to strip the electrons from the hydrogen atoms to get the naked protons. Those are then accelerated by a linear accelerator (LINAC 2) to 50 MeV after which they are injected into the Proton Synchrotron Booster (PSB) to increase the energy to 1.4 GeV. The PSB is followed by the Proton Synchrotron (PS) to accelerate the protons to 25 GeV. The last step before the LHC is the Super Proton Synchrotron (SPS) to increase the energy further to 450 GeV.

In the LHC the protons are then accelerated to the final energy of up to 6.5 TeV per particle beam. The ions on the other hand take a slightly different path. The lead-ions are generated from a vaporised solid lead block and enter LINAC 3 for a first acceleration, followed by the Low Energy Ion Ring (LEIR) which pushes them from 4.2 MeV to 72 MeV. From here on, they are injected into the PS and follow the same path as the protons to end in the LHC [8].

The particle beams are brought to collision at four interaction points in the LHC, around which the four major experiments are build to study the reaction products of the collisions. These four experiments are:

ATLAS (A Toroidal LHC ApparatuS): A general purpose detector located at interaction point 1. The detector was designed for the search of the, by now discovered, Higgs boson [9, 10]. Also the search for hints of new physics beyond the standard model, such as decays of supersymmetric particles, and extra dimensions are part of the physics program [11].

ALICE (A Large Ion Collider Experiment): The dedicated heavy-ion experiment at the LHC. The detector is installed at interaction point 2 and was designed for the detection and the investigation of the QGP [12]. A more detailed description is given in the following section.

CMS (Compact Muon Solenoid): A general purpose detector as well, installed at interaction point 5. The objectives are similar to those of ATLAS but with different detector techniques for a complementary measurement [13].

LHCb (LHC beauty): The dedicated experiment for heavy flavour physics, located at interaction point 8. The experiment is build to look at CP violation and rare decays of bottom and charmed hadrons to find indirect evidence of new physics. Since these hadrons are predominantly produced at small opening angles with respect to the beam pipe in pp collisions. The construction of the experiment differs strongly from those of the other major experiments: LHCb is build as a single-arm spectrometer [14].

Three more, but smaller experiments can be found at the LHC. LHCf (LHC forward) is located on both sides of interaction point 1 with a distance of approximately 140 m to the collision point. It measures neutral particles, emitted in the very forward directions [15]. TOTEM is located along the beam pipe at distances of ± 147 m and ± 220 m from interaction point 5. It measures the total cross-section in pp collisions and studies the elastic and diffractive scattering independent of the luminosity [16]. The last one is the MOEDAL (Monopole and Exotics Detector At the LHC) with the task to look for magnetic monopoles in a direct search. The experiment is build at interaction point 8 [17].

1.2 ALICE at the LHC

As the dedicated heavy-ion experiment at the LHC, ALICE is designed to withstand the high particle density environment, expected at such collisions. The high granularity allows to measure charged particle multiplicities of up to $\mathrm{d}N_{\mathrm{ch}}/\mathrm{d}y \approx 8000$ from a minimum transverse momentum of $p_{\mathrm{T}} \approx 0.15\,\mathrm{GeV}/c$ and, due to the individual sub-detector concepts a PID is possible for up to $20\,\mathrm{GeV}/c$ [18].

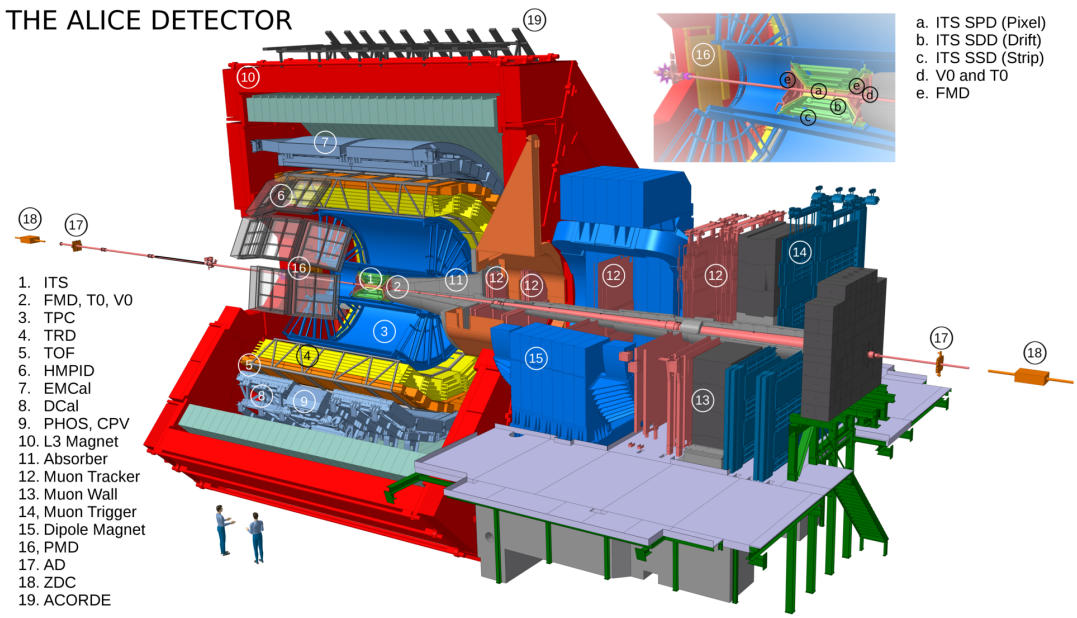


Figure 1.2: A schematic layout of the ALICE detector, showing the individual detectors. On the left side is surrounded by the red L3 solenoid the central barrel and on the right side is the muon spectrometer, taken from [19].

The ALICE detector is divided into the central barrel, fully contained within a solenoid magnet, and the muon arm on the C-side. This layout is shown in figure 1.2. The A-side is the detector part where the anti-clockwise circulation particle beam moves to, whereas the C-side is the opposite half, where the clockwise circulating beam moves to. The solenoid provides a magnetic field of 0.5 T and was inherited from the L3 experiment at LEP. The muon spectrometer starts also inside the central barrel with the hadron absorber, followed by the first two muon tracking stations. A third tracking station is enclosed in a dipole magnet, next to the solenoid. This is followed by two more tracking stations and iron absorber wall. Behind this absorber are the two muon trigger chambers. The beam pipe which has to go through the whole spectrometer is surrounded by low angle absorbers. The detectors of the central barrel are dedicated for tracking and PID. From inside out, there is first the Inner Tracking System (ITS) surrounding the interaction point. The ITS is enclosed by the TPC and the Transition Radiation Detector (TRD), followed by the Time-Of-Flight (TOF) detector, all having a full azimuthal coverage. Further outside with a reduced coverage there are three calorimeters, the ElectroMagnetic CALorimeter (EMCAL), the Di-jet CALorimeter (DCAL) and the PHOton Spectrometer (PHOS), and a Ring-imaging Cherenkov detector (HMPID, High Momentum Particle IDentification). There are more detectors for the event characterisation, T0, V0 and the Zero Degree Craorimeter (ZDC). The Photon Multiplicity Detector (PMD) and Forward Multiplicity Detector (FMD) are used for particle multiplicity measurements. The ALICE COsmic Ray DEtector (ACORDE), which is located on top of the L3 magnet. It allows to trigger on muons from cosmic ray showers. In the following, the four main detectors of the central barrel are described in more detail.

1.2.1 Inner Tracking System

The ITS is the innermost detector of ALICE. It combines three different technologies of silicon detectors in six cylindrical layers covering a radius from 4 cm to 44 cm. From the inside out there are first two layers of Silicon Pixel Detector (SPD), then two layers of Silicon Drift Detector (SDD) and then two layers of Silicon Strip Detector (SSD). Its main task is to provide the primary vertex reconstruction with a resolution of better than 100 μm. In addition, the SDD and SSD can be used for PID via the $\mathrm{d}E/\mathrm{d}x$ signal of the traversing particles due to the analog readout of the four outer layers [12].

1.2.2 Time-Projection Chamber

The TPC is the main tracking and PID device of ALICE. It consists of a cylinder whose axis is aligned with the beam axis and ranges from an inner radius of about 85 cm to an outer radius of about 250 cm. The cylinder has a length in z-direction of 500 cm, giving an active volume of $\sim$87 m^3. The TPC covers the full azimuth and a pseudorapidity range of $|\eta| \lesssim 0.9$ for full tracks. The pseudorapidity is defined as

$$\eta = -\ln\left[\tan\left(\frac{\theta}{2}\right)\right], \tag{1.1}$$

with the polar angle θ between the positive z and y-axis. The z-axis points from the collision point towards the A-side of the detector, while the y-axis points upwards, to the surface. For completeness, the x-axis points horizontally towards the centre of the colliders.

The volume of the TPC is split by a central electrode at 100 kV into two drift regions. This drift field of 400 V/cm leads to a drift time of 94 μs for the electrons for the maximum drift length. The ROCs of the TPC are installed on both end plates of the cylinder. During Run 1 (2009–2013) and Run 2 (2015–2018) of the LHC their design was based on Multi-Wire Proportional Chambers (MWPCs) with a cathode pad readout. The replacement of those chambers by new ones based on GEM technology is the essential part of the upgrade program and described in chapter 2. The end plates are subdivided into 18 trapezoidal sectors, each covering 20° in azimuth. Since the track density depends on the radius, the requirements for the ROCs differ as a function of the radius. In order to meet these different requirements, each sector is divided into two separate chambers, an Inner Readout Chamber (IROC) which extends from 84.1 cm to 132.1 cm and an Outer Readout Chamber (OROC) from 134.6 cm to 246.6 cm [20]. The basic working principle of a TPC is described in section 2.1.

1.2.3 Transition Radiation Detector

The TRD is the next detector of the central barrel in radial direction. It covers the full azimuth as well and a pseudorapidity range of $|\eta| < 0.84$. The subdivision into 18 sectors, the so called Super-Modules (SMs), was done to follow the segmentation of the TPC. Each SM consists of five stacks in z-direction, each stack of six layers of TRD chambers. This sums up to a total number of 522 individual chambers since the central stack of three SMs were kept empty to reduce the material budget in front of the PHOS detector. The layers are arranged at a radial distance between 2.90 m and 3.68 m from the beam axis. Each

chamber consists of a foam/fibre radiator, followed by a 3 cm drift region and a Xe-CO_2 filled MWPC. The electronics to read out the individual pads is directly mounted on top of the chambers. The radiator is used to generate the Transition Radiation (TR) photons, which are produced when a particle crosses the boundary between two media with different dielectric constants. Since the production probability of the TR at a single boundary is very low — in the order of the fine structure constant $\alpha = 1/137$ — many boundaries are needed for a significant photon yield. To absorb the TR photons efficiently, which extend into the X-ray domain due to the high γ factor* of the highly relativistic particles, the chamber is filled with a high-Z gas (Xe).

By having a sufficiently high sampling rate, a temporal information is extracted, representing the depth in the drift volume at which the signal is created. With that, the signal of the TR photons, which are preferably absorbed at the entrance of the chamber near to the radiator, can be distinguished from the signals resulting from the specific energy loss of the charged particle. This is uniformly distributed along the crossing particle trajectory.

To separate electrons from other charged particles, in particular pions, the presence of the TR signal respectively the absence of the signal for other particles due to a lower γ factor can be used. An additional criterion for the separation is also the higher $\mathrm{d}E/\mathrm{d}x$ signal for electrons due to the relativistic rise of the specific energy loss. The overall momentum resolution of the tracking in the ALICE central barrel is improved by including the six extra space points of the TRD at larger radii. Thanks to the fast readout and the online reconstruction capabilities, the TRD has also been used to trigger on electrons with a high transverse momentum and on jets [21].

1.2.4 Time-Of-Flight detector

Also the TOF detector adapts the segmentation into 18 sectors of the TPC and covers the full azimuth. It is located adjacent to the TRD at radii of 370 cm to 399 cm from the beam axis. As the other central detectors of ALICE, it covers a pseudorapidity range of $|\eta| \lesssim 0.9$. The utilised detector technique is that of a Multi-gap Resistive-Plate Chamber (MRPC). The individual chambers consist of resistive glass plates with a high and uniform electric field, due to an applied voltage of 13 kV. Any initial ionisation, generated by a crossing charged particle, starts immediately an avalanche leading to the observed signal on the pick-up electrodes. Since there is no electron drift involved, the time jitter is caused only by the fluctuations in the growth process of the avalanche [12]. With this MRPC technology, a time resolutions of better than 50 ps is achieved [22]. By calculating the time-of-flight between the TOF signal and the collision time — either measured by the T0 detector, the LHC central timing or by TOF itself if enough tracks are present — a PID can be done since the flight time is characteristic for each particle species at a given momentum.

*The Lorentz factor is defined as $\gamma = \sqrt{1 + \left(\frac{p}{m_0 \cdot c}\right)^2} = \frac{E_{\mathrm{tot}}}{m_0}$, where c is the speed of light, p the momentum, m_0 the mass and E_{tot} the total energy of the particle.

1.3 Detector and Readout Upgrades for the LHC Run 3

To be able to read out the all PbPb events at an interaction rate of 50 kHz, as it is foreseen for LHC Run 3, and in order to improve the vertexing and tracking capabilities of ALICE, several detector upgrades are foreseen. Those upgrades will be done during the LS 2, ranging from December 2018 to spring 2021. The upgrade program starts with a new beam pipe with a smaller diameter, continues with a completely new, high-resolution and low-material ITS and a major upgrade of the TPC by replacing all MWPC based ROCs with new ones employing the GEM technology. The readout of TRD, TOF, PHOS and the muon spectrometer are upgraded and optimised for the very high rate. The forward trigger detectors are upgraded as well and the software framework for the online systems, the offline reconstruction and the physics analysis is completely rebuild [3]. It is also foreseen to add the new Muon Forward Tracker (MFT) to the acceptance of the muon spectrometer. This is a silicon pixel detector placed in front of the absorber which increases the pointing accuracy of the extrapolated muon tracks to reliably measure their offset with respect to the primary vertex of the collision [23].

One of the key ingredients in improving the general impact parameter resolution of the experiment is the new beam pipe with a reduced radius from previously 29 mm to 18.2 mm. This allows to move the innermost layer of the new ITS closer to the interaction point to a radius of 22.4 mm [24]. This together with a strongly reduced material budget by utilising the Monolithic Active Pixel Sensors (MAPS) technology (radiation length of only 1.1 % X_0 per layer) and an all-pixel ITS with now seven layers at radial distances of 22.4, 30.1 and 37.8 mm for the inner barrel and 194.4, 243.9, 342.3 and 391.8 mm for the outer barrel will improve the track position resolution at the primary vertex by a factor three or better [25].

The exchange of the ROCs for the TPC allows for a continuous readout. With this it will be possible to record all events of the 50 kHz PbPb collision rate but requires new electronics as well. The upgrade will be described in more detail in chapter 2.

The TRD FEE can only be operated in a triggered, single event readout mode. Implementing a continuous readout would mean to exchange these electronics, requiring a complete disassembly and reconstruction of all the 18 SMs to replace the front-end boards mounted on each chamber. Since this is not realistically feasible, a different approach will be implemented. A reduction in event readout time by reducing the data volume will increase the possible trigger rate of the TRD. By transmitting only the so called *tracklets* (preprocessed track segments in a single detector chamber) instead of the raw data, it will be possible to record more than 70 % of the events in the 50 kHz PbPb collision scenario. Besides that, no issues concerning the detector stability or space charge effects due to the increased multiplicity are expected [26].

The TOF upgrade program foresees only changes in the readout tree, too. Measurements have shown that the intrinsics rate capabilities of the MRPCs is not an issue and that they can easily sustain the expected particle fluxes of 100 Hz/cm^2. The current TOF readout is already able to work with a trigger rate of tens of kHz. However, the upgrade aims to further increase this limit for both, PbPb and pp interactions [3].

Chapter 2

TPC with a Continuous Readout

This chapter covers first the basic working principle of TPCs. Afterwards the ALICE TPC is introduced as it was used during Run 1 (2009–2013) and Run 2 (2015–2018) of the LHC. The readout upgrade of the TPC is presented together with the timeline at the end of this chapter, to put the developments of this thesis into context.

2.1 Working principle of Time-Projection Chambers

A charged particle with high enough momentum traversing the active volume of a TPC will ionise the atoms and molecules on its path. Due to an applied electric field, the drift field $\vec{E}_d$, the electrons will move towards the Readout Chambers (ROCs) and the ions in the opposite direction, towards the central cathode. After arrival in the ROCs the electrons will be amplified and the generated signal can be read out. The x and y-coordinates of the arriving electrons is then determined by the location of arrival e.g. via a segmented pad plane. The z-coordinate is calculated from the arrival time t_a. By knowing the time t_0 when the charged particle crossed the TPC, for example by an external trigger detector, and by knowing the drift velocity v_d of the electrons in the used medium, the coordinate in z-direction can be calculated with $z = (t_a - t_0) \cdot v_d$. If a sufficient number of space points are obtained, the 3-dimensional trajectory of the incident particle can be reconstructed.

A TPC can also have Particle Identification (PID) capabilities via the signal amplitude. If the read out signal is proportional to the originally deposited charge, the particle can be identified via the specific energy loss $\mathrm{d}E/\mathrm{d}x$, which is described by the Bethe-formula [27], in combination with the momentum information. Therefore, it is beneficial to place such a TPC in a magnetic field $\vec{B}$. First, the trajectories of the charged particles will be curved due to the Lorentz force. By measuring the curvature of the trajectory the particles momentum can be obtained directly with the TPC. Second, a magnetic field in parallel to the drift field $\vec{B} \parallel \vec{E}_d$ reduces the diffusion in transverse direction of the electron cloud along the drift path and has therefore a focusing effect [28]. Orienting the two fields parallel to each other has also the advantage that the Lorentz force does not act in the drift direction, only perpendicular.

A disadvantage of TPCs for high rate experiments is the long drift time of the electrons in combination with a possible additional dead time imposed by an ion capturing process.

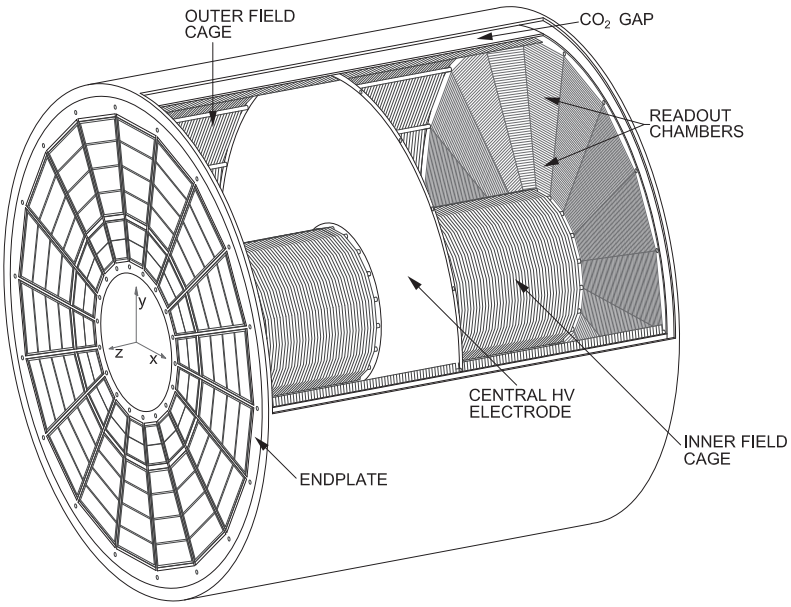


Figure 2.1: Schematic view of the ALICE TPC, taken from [5].

The electrons which were generated close to the cathode must drift through the complete volume to the end caps. Depending on the length and the used material, this can easily take tens of μs. The ions which are created in the electron amplification process may have to be captured to avoid an accumulation of charges in the drift volume. This can e.g. be done with a gated readout introducing a not negligible detector dead time of tens to hundreds of μs due to the slow ion mobility. Crucial for the operation of a TPC is the control of the environmental conditions. It must be ensured that the drift velocity stays constant to be able to reconstruct the z-position with a high resolution. If the pressure or the temperature changes, this must be taken into account. Also the electric field must be uniform over the full volume of the TPC.

2.2 The ALICE Time-Projection Chamber

A layout of the ALICE TPC is shown in figure 2.1. As already mentioned in subsection 1.2.2 the general structure is a hollow cylinder with an outer radius of about 250 cm and an inner one of about 85 cm. The overall length in z-direction is 500 cm, evenly split into two drift regions by the central electrode, each with a length of about 250 cm. The two endplates are divided into 18 sectors, each equipped with an Inner Readout Chamber (IROC) and an Outer Readout Chamber (OROC), indicated by the trapezoidal shapes well visible on the right hand side of the figure.

2.2.1 The Field Cage

To provide a uniform electrostatic field in the drift volume, a field cage is used. In addition the cage provides a stable mechanical structure for the individual detector elements, like

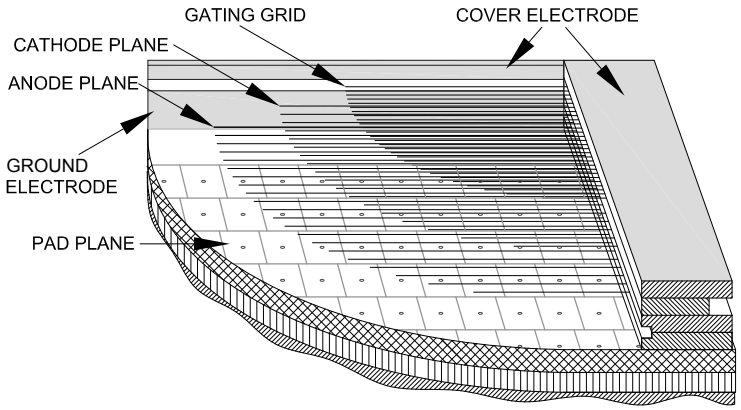


Figure 2.2: Cross-section of a MWPC based ROC, taken from [20].

the ROCs, forms a gas-tight envelope and ensures an electrical insulation from the other detectors of the experiment. The insulation is also achieved by the CO_2 filled gas gap in-between the field cage and the containment vessel. The field cage consists of two parts, the vessel and the actual field cage, consisting of 165 strips (in both drift volumes) surrounding the drift volume on the inside and outside. The vessel has a set of coarsely segmented guard rings which help to avoid the build-up of charges on the surface. It is made out of 13 mm wide strips of aluminium tape on both sides of the vessel, spaced by 92 mm. The two corresponding rings are electrically connected through small drilled holes with an aluminium foil feed-through which are sealed with epoxy. The potential of the individual rings is defined by a resistor chain, which is connected on one side to the central electrode and on the other side to ground, to match the field gradient of the inner, finer segmented field cage. Also the potential on the field cage strips is defined by a separate resistor chain. With this design it is possible to have the central electrode on a potential of 100 kV to generate a drift field of 400 V/cm. This leads to a drift velocity of 2.65 cm/μs for the electrons at nominal temperature and pressure and with that to a maximum drift time of 94 μs [20].

2.2.2 The MWPC Readout Chambers

At the end of the drift regions there are the ROCs mounted to amplify the incoming electrons and read out the signals. Different technologies have been considered and tested as amplification structures, such as Ring Cathode Chambers (RCCs) or Gas Electron Multipliers (GEMs) [29]. However, at the time only the Multi-Wire Proportional Chambers (MWPCs) met the requirements of the ALICE TPC and were in an R&D state to be readily adopted in such a large detector project. Therefore MWPCs were employed as the ROCs [20].

The MWPCs of the TPC have a commonly used scheme of wire planes. In the cross-section shown in figure 2.2 one can see (from the bottom up) the segmented cathode pad plane for the readout, then the anode wire plane for the amplification and next the cathode wire plane. On the very top there is another wire plane, the gating grid. This is used to shield the amplification regions from the drift volume. In the absence of a valid trigger the gate can be closed to prevent further electrons to enter the amplification region and

also to prevent ions created in the avalanche process to drift back into the drift volume. If they would not be stopped, they would accumulate in the TPC volume and cause severe distortions of the drift field leading to deflections of the drifting electrons and with that to degradations of the space-point resolution [20].

2.3 Readout Upgrade of the ALICE TPC

The just mentioned necessity of the gating grid imposes an intrinsic upper limit on the readout rate of the TPC of ∼3.5 kHz. The gate must be opened for about 100 μs to collect all electrons even from the maximum drift distance. Afterwards the gating grid must remain closed for about 180 μs, due to the low ion mobility of $\mu_{\text{ion}} = \mathcal{O}(10^{-3}) \cdot \mu_{\text{electron}}$, to stop all ions generated during the amplification process. It must be noted that the data rate of the present readout system is an even more limiting factor, in central PbPb collisions is a readout rate of only ∼300 Hz possible [5]. Such a system can not be used in LHC Run 3 where ALICE plans to inspect all events of the foreseen interaction rate of 50 kHz in PbPb collisions. So the whole readout system, the ROCs and the subsequent readout chain, must be upgraded in order to overcome this limitation.

2.3.1 Upgrade Goals

The goal of the TPC upgrade program is to employ a continuous readout system to overcome the intrinsic rate limitation. At the same time the upgraded TPC must preserve its current performance in momentum and $\mathrm{d}E/\mathrm{d}x$ resolution. This means, the distortions must stay at a tolerable level, so that they can be corrected to a few hundred μm, which is the intrinsic spacial track resolution of the ALICE TPC [5]. The local energy resolution must not exceed $\frac{\sigma_E}{E} < 12\,\%$ for an energy disposition of 5.9 keV, evaluated with a radioactive ^{55}Fe source [5]. To keep the distortions small enough, the Ion Back-Flow (IBF), which is defended as

$$\text{IBF} = \frac{1+\varepsilon}{G_{\text{eff}}}, \tag{2.1}$$

must be less than 1 %, where G_{eff} is the effective gas gain of 2 000 and ε is the number of ions drifting back into the drift region per incoming electron [5].

2.3.2 GEM based Readout Chambers

The goals can be achieved by using GEMs for the gas amplification. The new ROCs will be build with a quadruple GEM stack (four GEM foils on top of each other) for the amplification and ion blocking, and with an anode pad readout. The working principle and the intrinsic ion blocking capabilities of a GEM foil can be explained easiest on a picture like it is shown in figure 2.3. The GEM consists of a thin insulating polymer foil which is metal coated on both surfaces. A regular pattern of small holes is etched through the metal layers and the polymer in a conical shape so that the wider radius is on the entry sides. By applying a potential difference between the metal surfaces, an electric field develops in the holes. Already with a moderate potential difference of 200 V high electric fields of 40 kV/cm can be generated in the gaps [31].

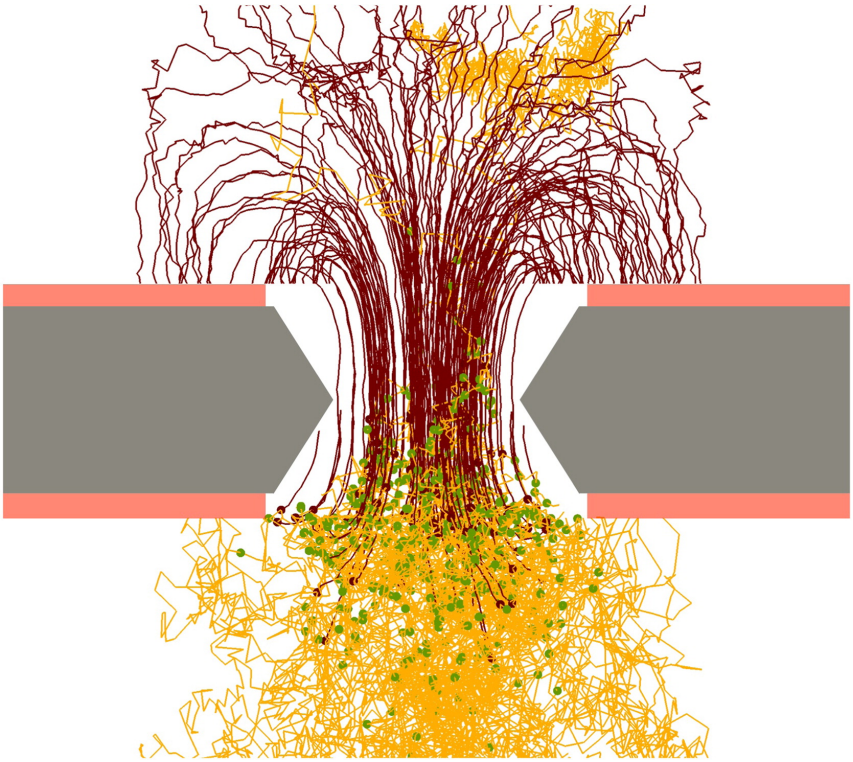


Figure 2.3: Simulation of the gas amplification of two electrons in a GEM hole with Garfield/Magboltz. The electron (yellow) and ion (red) paths are projected to the cross-section plane. Each green marked spot indicates an ionisation process, taken from [30].

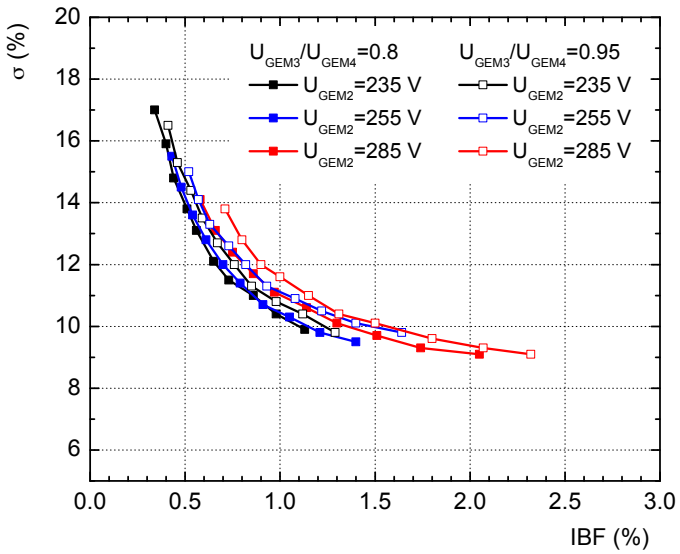


Figure 2.4: Energy resolution and IBF for different GEM voltage settings, taken from [5].

Figure 2.3 shows a simulation of an amplification process of two incoming electrons in a GEM hole. All the paths are projected to the cross-section plane. Two electrons (the yellow lines) enter the hole and start an avalanche due to the high electric field. Each green dot marks the position, were an ionisation took place. Finally, a plurality of electrons leave the hole on the lower side after the amplification region, of which some, following the field lines, end on the bottom side of the foil. The ions (red lines) which are also created at the ionisation spots, emerge the hole on the upper side, following the field lines as well. They end mostly on the top side of the foil because the field above the GEM is much lower than the field inside the hole. The extraction of the electrons can be supported by using a higher transfer field below the GEM [30]. By stacking multiple layers of GEM foils high total gains can be achieved with only moderate gains at each individual foil.

The necessary local energy resolution could be reached with a stack of four GEMs, while utilising the intrinsic ion capturing mechanism to achieve an IBF of less than 1 %, as it is shown in figure 2.4. The achieved resolution with the corresponding IBF is plotted for different voltage settings. The voltage difference on GEM 2 (GEM 1 is the uppermost foil, facing the drift volume while GEM 4 is last one before the pad plane) is directly given, the voltage on GEM 3 and GEM 4 were adjusted for a gain of 2 000 while keeping the ratio constant. The voltage on GEM 1 increases from left to right from 225 V to 315 V. As can be seen, there is some parameter space available which reaches the required properties [5].

2.3.3 Upgrade of the Front-End Electronics

The new chambers result in three major changes in the requirements for the FEE and the readout systems. First, the polarity of the GEM detector signal is opposite to the one of the MWPC. Second, the continuous readout scheme makes it necessary to develop new electronics for a concurrent data sampling and data transmission. Third, the increased interaction rate together with the continuous readout leads to a strongly increased data

rate. This makes a completely new readout chain necessary, which is described in detail in chapter 3.

2.3.4 Towards an Online Track Reconstruction

The average data size of one minimum bias PbPb event of the upgraded TPC is expected to be ∼20 MB after an already applied Zero Suppression (ZS), which reduces the volume already by a factor of about 3 [5]. This would lead (with an interaction rate of 50 kHz) to a data rate of 1 TB/s into the online data system. Integrating the data over the full Run 3 of the LHC, an amount of 3 EB (10^{18}B) would be collected. Since those numbers exceed both, the predicted available bandwidth as well as the storage space, additional compression methods on top of the ZS must be applied directly during the readout to reduce the data rate to less than 1 MB per interaction [5].

As a first processing step is a Cluster Finder (CF) foreseen (the development of the CF is the main topic of this thesis) which is supposed to run directly in the online farm on FPGAs. After the cluster finding a Huffman compression can be done on selected parameters. During the PbPb data taking of LHC Run 1 in 2011 such a compression scheme was already used successfully, leading to a compression factor of about 4 [5]. With further optimisations on the format of the cluster data in the software framework and the compression algorithm, a compression factor of five to seven is expected [3].

In order to further reduce the data volume, online track reconstruction is used, whereupon the clusters are assigned to the found particle tracks. Thus clusters that do not belong to physically relevant tracks (e.g. clusters from noise or delta electrons) can be removed. Based on experience from the Run 1 data taking, a compression factor of two is expected. In addition, the parameter distributions can be optimised for entropy encoding and eventually some cluster parameters can be replaced by track-based properties. With that, an additional compression factor of two to three is expected [5].

This gives a total expected compression in the order of 20 which reduces the average event size to less than 1 MB. With that is the data rate to the permanent storage reduced to ∼50 GB/s and only a total amount of around 150 PB for the complete Run 3 data is needed [5]. The online CF is the very first step in this chain.

2.4 Timeline of the TPC Upgrade

The upgrade program and all the individual tasks which are needed to successfully assemble the upgraded TPC are described in [5]. Though, there were some adaptions and delays since. The main milestone that must be kept is the Long Shutdown (LS) 2 which started in December 2018. During this shutdown the TPC must be taken out of the L3 magnet and brought to the surface in a clean room in which the replacement of the ROCs takes place.

The program started already six years earlier at the beginning of 2013 with the R&D phase for the ROCs, followed closely with the beginning of the design phase of the FEE in mid 2013. The production of the first prototypes could then take place in 2015 until one of the almost final *pre-production* IROCs was tested with the first Front-End Cards (FECs) in a test beam campaign in May 2017. This was also used to test the first Firmware (FW) components, developed in the scope of this thesis, in a real readout system. During the

rework of the TPC in the cleanroom between March 2019 and February 2020 [4] there are many individual tasks, described in detail in [32]. It is expected that the first side of the TPC will be equipped with the new ROCs and the new electronics in October 2019, so that the system can then be tested. This will be the first time that realistic signals can be read out by the new readout system. It will be the first step in the commissioning of the FW modules developed in the scope of this thesis. After the reinstallation of the TPC in the experiment which will be finished in May 2020 [4], further validations can be done to be ready for the restart of the LHC in spring 2021.

Chapter 3

Readout Strategy for the ALICE TPC

The readout scheme for the ALICE TPC in Run 3 was originally described in the Technical Design Report (TDR) for the Upgrade of the ALICE TPC [5] and in the TDR for the Upgrade of the Readout & Trigger System [26]. The basic idea is that signal pulses induced on the individual pads of the pad plane of the Readout Chambers (ROCs) are amplified, shaped and digitised continuously by the newly developed SAMPA ASIC [26]. These ASICs are placed on Front-End Cards (FECs), which are located close to the detector. The continuously sampled data is then transferred via two GigaBit Transceiver (GBT) ASICs and the versatile optical link components to the Common Readout Unit (CRU). Further processing steps, especially a clusterisation which is the main topic of this thesis, is then applied in the CRU. The CRUs are located off-detector, outside of the radiation area, in Counting Room (CR) 1, see section 3.4. Always two CRUs are hosted by one First Level Processor (FLP) server. This readout chain is schematically shown in figure 3.1. Two major design changes with respect to the original concept of the TDRs were implemented. It was observed that due to the so called Common Mode (CM) effect, no Zero Suppression (ZS) can be applied on the FEC, which is why the SAMPA must be operated in Direct ADC Serialisation (DAS) mode to read out unmodified raw data. To be able to transmit this data volume, the ADC sampling frequency had to be reduced from 10 to 5 MHz [33]. This chapter covers a short introduction to the original strategy and presents then the CM effect. Afterwards, a study is performed about the possible application of the loss-less Huffman compression to reduce the data volume without the reduction of the sampling frequency, which was also part of this thesis. Then the updated readout scheme is introduced and finally the main components of the readout chain, the SAMPA, the GBTx and the CRU, are described with the focus on the TPC readout for Run 3.

3.1 The TDR Baseline

The general parameters of the digitisation process, such as the sampling rate and the ADC precision were taken from the currently installed system which was running very successfully during Run 1 and 2 of the LHC. The individual pads of the TPC will be connected to an ASIC, called the SAMPA, which includes a Charge Sensitive Amplifier (CSA), a shaper and a 10 bit ADC, sampling with a frequency of 10 MHz. Each FEC contains five SAMPAs with

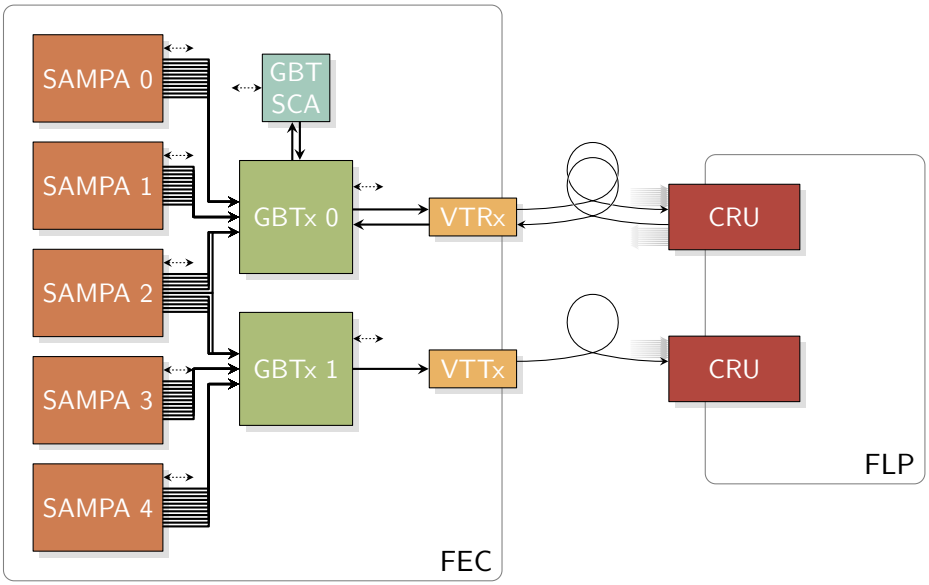


Figure 3.1: The main components of the TPC readout chain. The FEC on the left hosts five SAMPA ASICs which are connected to the two GBTx. The SC, implemented through the GBT-SCA, is indicated with dotted arrow lines. The FEC is read out by two CRUs, hosted by a FLP server. Each CRU receives the data from multiple, up to 20, FECs.

32 channels each. This gives in total 160 channels per FEC. The five SAMPAs will therefore generate with those parameters a combined continuous data rate of

$$5 \times 32 \times 10\,\text{MHz} \times 10\,\text{bit} = 16\,\text{Gbit/s}. \tag{3.1}$$

Such a FEC is schematically shown on the left side of figure 3.1. The SAMPAs on the left are connected to the two GBTx ASICs which are responsible for the communication with the outside-world. With 91 FECs installed in each of the 36 sectors of the TPC, a total amount of 3 276 FECs will be needed in the final system [34]. The 524 160 individual readout channels will generate an overall data rate of

$$3\,276 \times 16\,\text{Gbit/s} = 52.416\,\text{Tbit/s} = 6.552\,\text{TB/s} \tag{3.2}$$

for the whole TPC. These numbers are changed slightly with respect to the TDR due to adaptions of the pad plane layout. To reduce the overall data volume, it was foreseen to apply a ZS on the digitised signals directly within the SAMPA chip, which contains a Digital Signal Processor (DSP) also for this purpose. In a ZS readout, those values are dropped which are below a certain threshold. The general assumption for a ZS is that the dropped value is too small to contain any useful information and therefore can be omitted. The remaining signals are then run-length encoded, which requires an additional information about the start and the length of the not zero suppressed sequence [5]. The ZS algorithm can also be more complex and can require for example several consecutive time bins above the threshold or it stores in addition some pre- and post-samples around the bin of interest. Applying a fixed threshold on each channel makes it necessary to first restore the baseline.

In a real system, the baseline can be different for each channel which requires a (channel wise) pedestal subtraction. Further, with the Gas Electron Multiplier (GEM) TPC, a big contribution of the so called CM effect is expected which needs to be corrected first. The CM effect is described in more detail in subsection 3.1.1.

As an alternative compression method (or applied in addition), the effect of loss-less transformations like variable length codes (e.g. Huffman coding) were studied (see subsection 3.1.2). If the probability distribution of the individual ADC values coming from the detector is not uniform then it may make sense to assign short codes to values which occur more often and long codes to values which occur rarely. Huffman coding approaches the theoretical lower limit for the average word size which can be achieved by this concept (also called the entropy of the data source).

To collect the data for further processing and storage, it is transmitted via two GBTx ASICs and the Versatile Link components (a Versatile Transceiver, VTRx, to implement one bidirectional link and a Versatile Twin Transmitter, VTTx, for a second up-link) from the FECs to the CRUs which are loceted off-detector in the CR. Each CRU receives the data of up to 20 half-FECs, so up to 1600 pads are combined in one CRU. The mapping of the individual pads via the SAMPAs and the FEC towards the CRUs is explained in more detail in appendix B. An important note at this point is, that the data of a single FEC is split and sent evenly distributed to two independent CRUs in a way which is indicated in figure 3.1. The data of SAMPA 0 and 1 is sent together with half of the channels of SAMPA 2 to one CRU while the remaining channels of SAMPA 2 are sent together with the data of SAMPA 3 and 4 to another CRU. With this scheme, always complete pad rows can be recovered in a CRU which is very important for the cluster finding later on.

The GBTx can transmit either 3.2 Gbit/s with an automatic error correction enabled, or 4.48 Gbit/s without (for more details see section 3.3). This implies that, to be able to transmit the 16 Gbit/s of each FEC, a compression factor of

$$\frac{16\,\text{Gbit/s}}{2 \times 3.2\,\text{Gbit/s}} = 2.5 \qquad \text{or} \qquad \frac{16\,\text{Gbit/s}}{2 \times 4.48\,\text{Gbit/s}} = 1.8 \tag{3.3}$$

must be achieved, either with a ZS, with other compression methods or with a combination of both. Especially for a Huffman encoded data stream it is important to use such error correction methods, otherwise it is easily possible, that whole chunks of data become corrupt due to a single bit-flip in the transmission. To avoid this, only the GBT mode with error correction and therefore lower bandwidth should be used for a Huffman compressed readout scheme.

3.1.1 The Common Mode Effect

It is well known that in ROCs, like they are used for the TPC, the readout pads are coupled capacitively to the amplification structure. This coupling is independent of whether this structures are wires, as in the current TPC, or GEMs, as in the upgraded TPC and leads to the CM effect. An avalanche has a flowing current as consequence which introduces a voltage drop on the electrode. Due to the voltage drop, a correlated signal with opposite polarity is induced on all pads. This is shown schematically in figure 3.2. The pads can be seen as a series of capacitors connected in parallel. When a signal occurs in one pad (red),

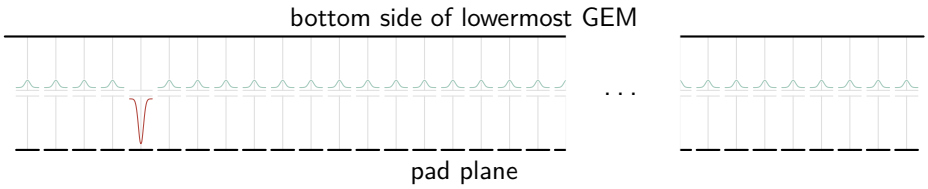


Figure 3.2: Schematic drawing of the origin of the CM effect due to capacitative coupling of the individual pads. A signal in one pad (red) will induce a correlated signal with opposite polarity on all other pads (blue) which face the same GEM electrode.

all other pads will see an inverted fraction of the signal (blue), where the amplitude of the CM signal on each pad corresponds to the original signal divided by the total number of pads facing the same GEM electrode (assuming all pads have the same capacitance). For multiple avalanches occurring at the same time those CM signals pile up. In a high-multiplicity environment this piled up signal leads to a reduction of the baseline and an effective noise contribution. By adding additional capacitance to the GEM one could damp the CM effect. However, this would counteract the efforts done in minimising the stored energy in order to improve the stability and safety of the system.

Different algorithms which could be applied directly in the SAMPA DSP have been studied and were presented in [33], with the goal to restore the baseline on a single-pad level. It occured that those filters could only partially restore the performance of the separation power between electrons and pions of the TPC. A sizeable degradation of roughly 20 % still remained. The document states also, that a better restoration of the baseline could be achieved by taking the information of a large number of pads into account. The CM signal should be the same for all pads below the same GEM electrode. By averaging over a large number of pads, excluding the signal peaks, the baseline can be restored sufficiently. The result of such an averaging approach with three different signal-exclusion mechanisms is shown in figure 3.3. On the left side the remaining bias of the baseline is displayed, the effective noise contribution of the CM effect on the right side. The blue points display the performance with a simple constant threshold of 2.5 ADC counts above which the ADC values are assumed to be signals and therefore are excluded. The performance shown by the magenta and black points is achieved by using a real peak finding algorithm where the peak is either completely taken out in case of the black points or the last time bin before the peak is still used for the calculation of the average charge. Given the black symbols, it can be seen that by averaging over more than about 100 pads (without a signal) the baseline can be restored with a negligible bias and noise contribution. However, each SAMPA has only the information of its 32 channels available. This is the reason why this correction method can not be applied in the SAMPA DSP. Since signals from up to 1600 pads are received by a single CRU, this filter can be placed there to restore the anticipated performance of the TPC, though, this requires the readout of not zero suppressed ADC values [33].

3.1.2 Application of a loss-less Data Compression

To reduce the data volume without the need of a ZS, the loss-less Huffman compression method was proposed and studied. It is a well-known technique, proposed by David

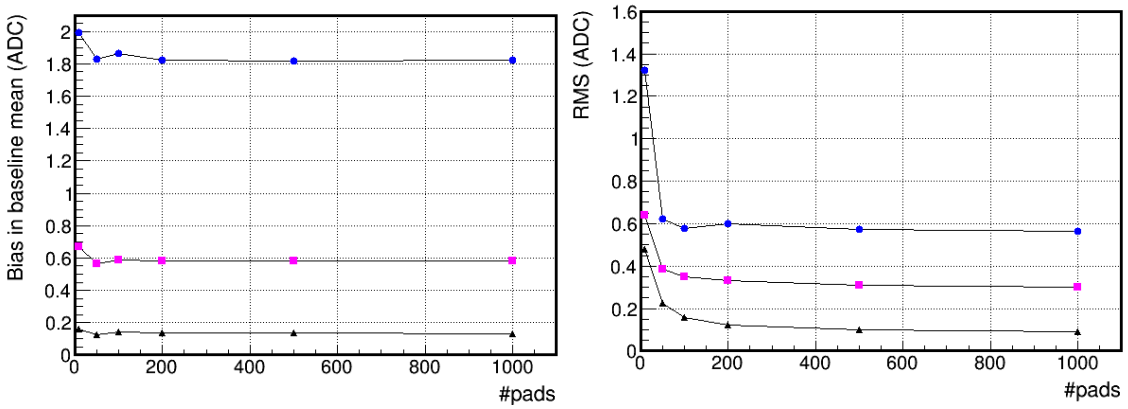


Figure 3.3: Remaining bias of the ADC value (left) and additional noise from the CM effect (right) after correcting with the averaging method. The result is shown as a function of number of pads which were taken into account. The three different colours show the three different peak exclusion algorithms (fixed threshold in blue, complete peak removal in black and in magenta where the first time bin of the peak was still included in the averaging), taken from [33].

Huffman in 1951 [35], that can easily be implemented in hardware and gives the optimal loss-less compression for a given set of probabilities. The idea behind the Huffman encoding is that the words which need to be compressed are replaced by codes of a variable length, where the length depends on the probability of their occurrence. Words that occur more frequently get shorter codes, while words that are rare get longer ones. By sending a stream of concatenated Huffman codes, the total volume of the data can be reduced. The codes are prefix codes, meaning that no code is the prefix of another one. Because of this, the stream can be decoded, by going through and detect the individual codes. The already mentioned possibility of a bit-flip during the transmission becomes therefore relevant. If one bit is changed, the detected code will be a wrong one, but even worse, the code will be detected as one with a different length. Due to that, all subsequent codes will be decoded wrongly as well. Therefore it must be ensured that no bit-flip can happen, or that it can be detected and corrected beforehand. Those codes can also be represented as a binary tree where the symbols correspond to the leafs of the tree [36]. The code can then be obtained by traversing the tree from the root node to the leaf node of the desired symbol, adding the character 0 to the binary codeword whenever the top branch is taken or the character 1 for the bottom branch. This is indicated in figure 3.4. Starting from the root node on the left, the codewords c_0 to c_4 can be obtained by following the branches and just appending the corresponding symbol to the code word. With a known probability distribution of the expected signals, those Huffman codes can be precalculated and stored in a Lookup-Table (LUT) on the Front-End Electronics (FEE).

For the study whether the Huffman encoding is applicable, the probability distributions were generated from the so called *black events* which were taken with the TPC during the PbPb run in the year 2010. Within a short period of a few minutes, around 1 000 events were recorded without a ZS applied. This raw data is then used to overlay multiple collisions inside a timeframe to emulate the occupancy levels expected for Run 3. The

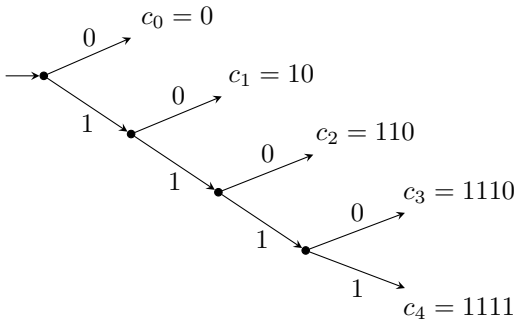
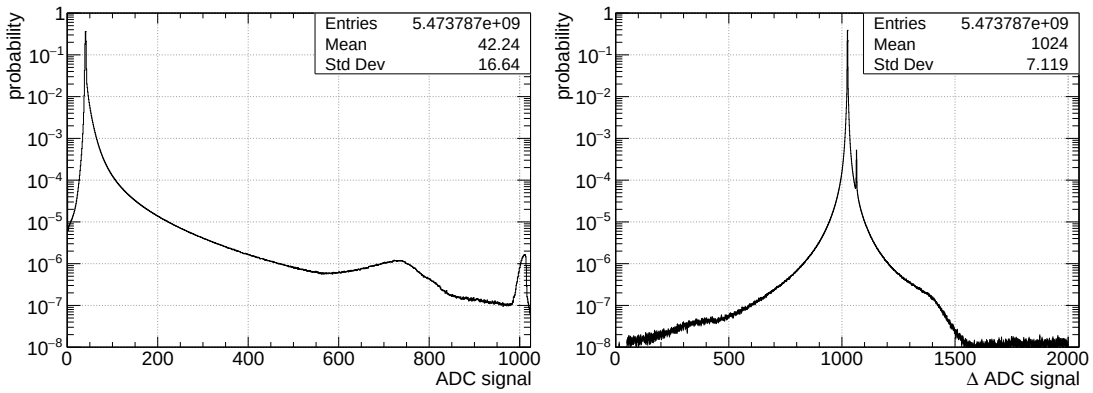


Figure 3.4: Example of a small Huffman tree with only five codes.



(a) Raw TPC signals.

(b) Differential TPC signals.

Figure 3.5: The probability spectrum of the ADC values of the ALICE TPC for PbPb collisions at $\sqrt{s_{\mathrm{NN}}} = 2.76\,\mathrm{TeV}$. On the left side is the probability of the pure detector output plotted. The right side shows the probability distribution for the differential signal, when the difference of two consecutive signals of the same pad is transmitted, shifted by 1 024 to avoid negative numbers.

resulting probability spectrum of the TPC raw signals can be seen in figure 3.5a. With the Huffman coding, an average code length of 3.21 bit was achieved, giving an average compression factor of 10 bit/3.21 bit = 3.12. It is clear that since the spectrum is rather wide, leading to a broad Huffman tree, it is possible to still improve the compression ratio. By encoding the differential signal, where always the difference of two consecutive time bins of the same channel is used, instead of the raw ADC value, a narrower distribution can be achieved. Such a distribution is shown in figure 3.5b which results in an average code length of only 3.01 bit and therefore an improved average compression factor of 3.32. For the second spectrum, a fixed offset of 1 024 was added to avoid the need of handling negative numbers in the FEE. Both compression factors would fulfil, in the ideal case, the requirement of > 2.5 to be able to transmit all the data from a FEC. This very narrow distribution becomes even more relevant by going from the standard Huffman, where the code words can have an arbitrary length, to the *truncated Huffman*, where the maximum code length is limited. Since the encoder has to be implemented in the frontend chip and

the applied Huffman table is stored there in a LUT for easy use, the maximum length is limited to 12 bit. The truncation is usually done by generating a normal standard Huffman tree and only the sufficiently short codes are used. All other words that can not be encoded, are transmitted as raw value with a special Huffman code prepended as a marker. After the detection of this special code, the decoder recognises the following bits as a raw word. This treatment increases the data volume since in such a case not only the original data needs to be transmitted, but also the special code beforehand. So the compression of the other words must be efficient enough to compensate for this overhead. And the compression is only efficient enough, if those raw data words need to be transmitted very rarely, or to put it another way, if the probability distribution is very narrow. In this case with the truncation to 12 bit only, a marginally worse average code length of 3.1 bit was observed, giving an average compression factor of 3.23.

A different way to achieve this truncation is to generate the so called length-limited Huffman codes directly. Here, an additional constraint is added to the generation procedure of the code words: the requirement that the length of all code words has to be less or equal to a given maximum length. A widely used algorithm for the construction of the code table is the *Package-Merge* algorithm from Larmore and Hirchberg [37]. As written in the paper, the algorithm actually solves the *Coin Collector's* problem, but it is also shown there that the length-limited Huffman coding problem can be reduced to an instance of the *Coin Collector's* problem. Because of that, the *Package-Merge* algorithm can also be applied to find an optimal Huffman table with this additional constraint. With this algorithm the number of words to be encoded and the maximum length of the codes can be fixed beforehand to generate the table accordingly. All words which can not be encoded are transmitted again as raw values with one of the codes prepended as a marker.

The optimal parameters for the length-limited Huffman were selected with the help of two criteria, the achieved compression factor and the required buffer size. Since the Huffman codes have a variable length, a FIFO (First In, First Out) must be placed after the encoder to compensate for the different lengths. Whenever a code occurs which is longer than the original 10 bit ADC value, then this FIFO is filled. A scan of the resulting compression factors for a variety of differential parameters is shown in figure 3.6, relative to the one obtained from the 12 bit truncated Huffman. As can be seen, the compression factor decreases for small code lengths and for an increasing number of words to be encoded. This is kind of intuitive, because if many words must be packed into a limited set of available codes, then the average number of used bits will increase and therefore the average compression factor decreases. For a comparable length of the code words (from 10 bit on), the length-limited Huffman performs slightly better than the truncated Huffman. This is expected since the Package-Merge algorithm finds the optimal code table for a given parameter set, whereas the table of the truncated Huffman is only optimal if all codes up to an arbitrary length would be taken into account. The compression factor peaks around 60 encoded words for the 10 bit case. It still increases by increasing both the number of encoded words and the code length, but not significantly. This means, by encoding only the 60 most probable words out of the differential spectrum range of 0 to 2 047, the best compression factors can already be achieved. This is again the reason why the spectrum needs to be so narrow. The optimal parameters of 60 words and 10 bit was then chosen despite the slight increase with a few more words and a longer code length, due to the required buffer size. The needed

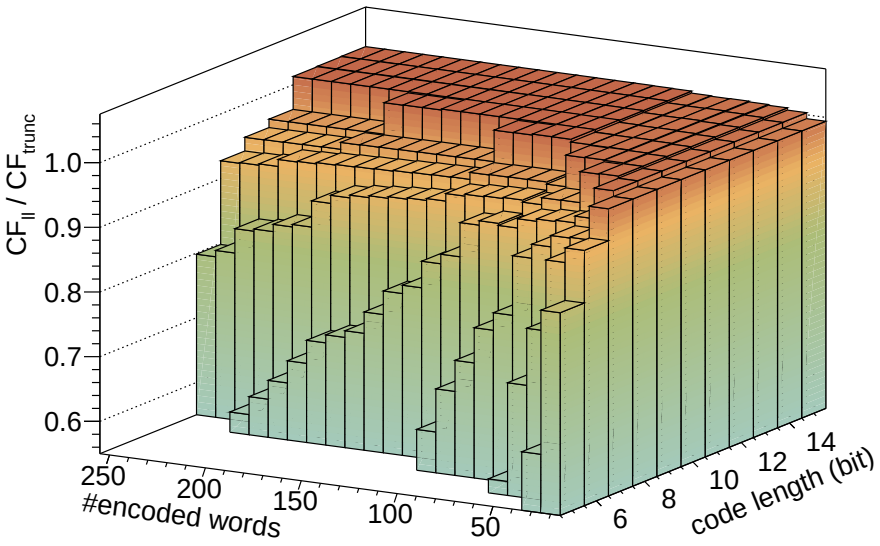


Figure 3.6: Compression factors achieved by the length-limited Huffman, relative to the to ones from the 12 bit truncated Huffman. The ratios are shown as a function of the number of encoded words and the maximum code length.

size increases dramatically, by almost a factor of two, as soon as the maximum code length is longer than the width of the input data of 10 bit. When allowing the most probable codes to be longer than this size, then it is clear that they occur rather often and fill the buffer. Therefore, the slight penalty in compression factor was accepted with the benefit of a smaller needed buffer size. With those settings, an average code length of 3.04 bit was found while compressing again the same data set, giving an average compression factor of 3.29. This is again in the same range as for the truncated Huffman.

Then the stability of the performance of those Huffman algorithms — the truncated Huffman and the 60 words and 10 bit length-limited Huffman — were studied for a variety of cases which are not unlikely to occur in the final detector. The CM effect was applied, the noise level artificially increased by a factor of two and the gain of the detector varied by up to 20 %. Every possible combination of these effects was also exercised. For all cases, the optimal Huffman table was generated and truncated to a maximum length of 12 bit and the optimal length-limited table calculated. The resulting compression factors as a function of the occupancy of the detector is shown in figure 3.7 for the length-limited Huffman. It can be seen that even for the ideal case, the achieved compression factor is for an occupancy of more than around 40 % below the required factor of 2.5. Although the maximum occupancy is expected to be not higher than 30 % [5], a bigger safety margin is needed for a reliable detector operation. It can also be seen that as soon as the noise increases, the overall compression factor decreases significantly. For a noise increase of a factor of two, the occupancy limit, until which the compression is still good enough, is reduced to below 30 % which is definitely too low. This observation is very plausible since when the fluctuation due to the higher noise gets bigger, the average differences are also higher and the peak in the probably distribution becomes broader. All the other effects

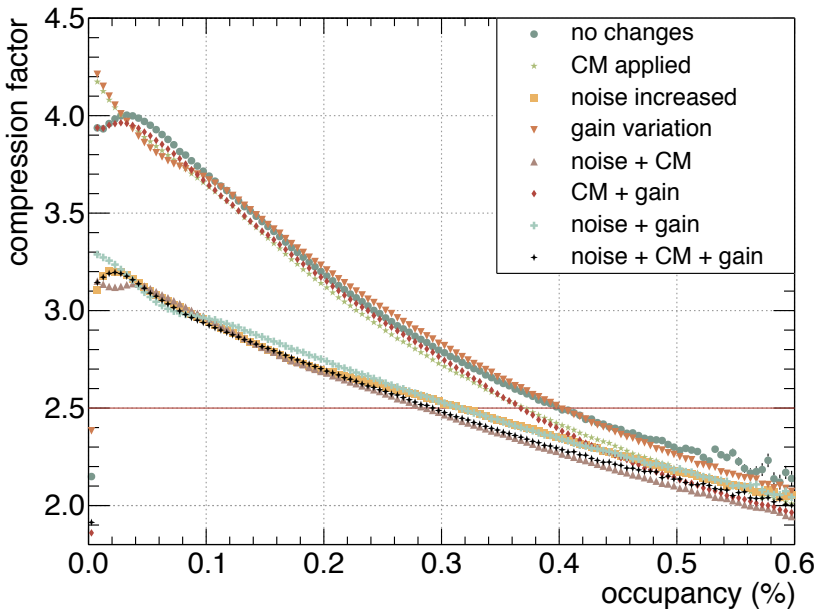


Figure 3.7: Compression factor of the length-limited Huffman encoding as a function of the occupancy for different modifications of the data set.

have only a minor impact on the achieved compression. These statements are true for both, the truncated and the length-limited Huffman. Indeed, both show a very similar compression factor however, the factor achieved by the length-limited Huffman is especially for a high occupancy environment up to 3 % better compared to the truncated Huffman. These plots were generated for the case that the probability distribution is known for the current readout case beforehand. A more significant problem occurs if the running condition changes during operation, e.g. if the noise suddenly starts to increase or the gain changes. The Huffman tables need to be precalculated and loaded into the hardware. Should a table be loaded which does not exactly fit to the current running conditions, the compression factors will be even worse. This, together with the visible strong decrease of the compression factor by an increased noise contribution, rules out the option of a Huffman encoded raw data readout for the ALICE TPC.

3.1.3 Change of the Readout Scheme

After it was found that the ZS can not be applied in the SAMPA DSP due to the CM effect, at least not without significantly degrading the data quality, and after finding in addition that the alternative approach of compressing the differential raw data with Huffman encoding is not robust enough against detector effects such as an increase of the noise contribution or changing running conditions, the readout scheme had to be reconsidered. There is no additional option left for a data reduction, so the complete data volume of all SAMPAs of 16 Gbit/s per FEC has to be transmitted to the CRUs. An additional complication is, that since the volume of the data already exceeds the available bandwidth, there is no additional

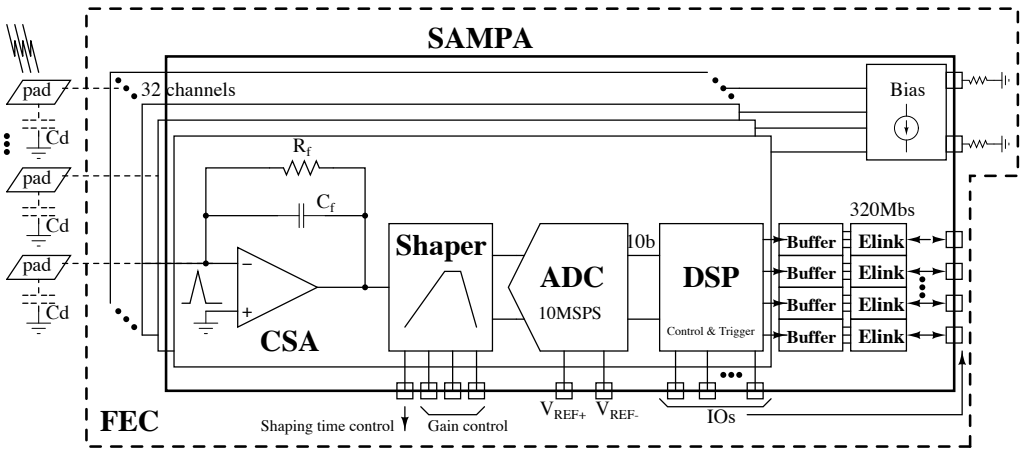


Figure 3.8: Block diagram of the SAMPA ASIC, showing the main building blocks. For each of the 32 channels, there is a CSA, followed by a shaper, a 10 bit ADC and a DSP, taken from [26].

space for a packaged data format with a header containing timing and status information. In order to be able to detect phase-shifts in the SAMPA ADC sampling clock, caused by single event upsets in the internal clock divider network, this clock is made available that it can be monitored. For this, the 10 bit ADC output bus of the SAMPA was extended by one further bit to 11 bit. This eleventh bit carries the ADC clock so that it is transmitted together with the data. Because the data of SAMPA 2 is shipped to two different receivers, as it can be seen in figure 3.1, this eleventh output must also be received by both, giving an effective width of the output of 12 bit for SAMPA 2. This results in a new bandwidth requirement of

$$(4 \times 11\,\text{bit} + 12\,\text{bit}) \times 32 \times 10\,\text{MHz} = 17.92\,\text{Gbit/s} \tag{3.4}$$

which is by chance exactly the bandwidth which is available by combining 4 GBTx ASICs of $4 \times 4.48\,\text{Gbit/s} = 17.92\,\text{Gbit/s}$. So the solution to be able to transfer all data is to either double the transmission bandwidth by using twice as many optical components, or to reduce the data volume by a factor of two e.g. by halving the sampling frequency. Since doubling the optical components would lead to substantial additional costs in the upgrade, the reasons for the 10 MHz sampling rate had to be reevaluated. Originally, the ALICE TPC was designed for a 5 MHz sampling rate to have roughly the same bin width in all three spacial directions [29]. This was changed later due to a slightly better performance of the ion tail cancellation filter with a 10 MHz sampling and a better sensitivity to cluster tails with an applied ZS. Since both facts will not matter anymore in the new system (no ZS can be applied and the GEM setup does not generate ion tails) the impact of reducing the sampling frequency on the physics performance was studied and presented in [33]. The simulations demonstrated that the Particle Identification (PID) performance as well as the tracking efficiency and momentum resolution are influenced to only a very small amount by the reduction of the sampling rate. Even with a low signal to noise ratio both sampling frequencies behave similarly. Based on this study it was decided to reduce the

	normal mode		split mode	
	serial links [9:5]	serial links [4:0]	serial links [9:5]	serial links [4:0]
cycle	ch [bits]	ch [bits]	ch [bits]	ch [bits]
0	0 [9:5]	0 [4:0]	16 [4:0]	0 [4:0]
1	1 [9:5]	1 [4:0]	16 [9:5]	0 [9:5]
2	2 [9:5]	2 [4:0]	17 [4:0]	1 [4:0]
3	3 [9:5]	3 [4:0]	17 [9:5]	1 [9:5]
4	4 [9:5]	4 [4:0]	18 [4:0]	2 [4:0]
⋮	⋮	⋮	⋮	⋮
28	28 [9:5]	28 [4:0]	30 [4:0]	14 [4:0]
29	29 [9:5]	29 [4:0]	30 [9:5]	14 [9:5]
30	30 [9:5]	30 [4:0]	31 [4:0]	15 [4:0]
31	31 [9:5]	31 [4:0]	31 [9:5]	15 [9:5]

Table 3.1: The two different transmission modes of the SAMPA in DAS mode. In normal mode, the 10 bit ADC values of each channel are provided one-by-one to the output ports. In split mode, the serial links are split in half, port 9–5 are used for channel 16–31 and port 4–0 for channel 0–15. Each 10 bit ADC value is therefore sent in two consecutive cycles, first the five LSB, then the five MSB. The pattern is repeated after 32 cycles in both modes.

sampling frequency by a factor of two to be able to transmit the raw data uncompressed and unmodified to the CRUs to apply the, for further processing necessary, baseline restoration there. In the following, the three main components of the readout and online processing chain are described in more detail with focus in the aspects which are most important for the readout of the TPC.

3.2 The SAMPA Chip

The main motivation for the development of the SAMPA was the change in the readout strategy in Run 3 towards a continuous readout. Therefore the presently used TPC FEE consisting of the 16 channel PASA ASIC (front-end amplifier and shaper) and the 16 channel ALTRO chip (10 bit ADC and DSP) was developed further. The new SAMPA ASIC integrates now 32 channels of the whole processing chain, which is indicated in figure 3.8. Each of the 32 paths consist of a positive/negative polarity CSA, a shaper, a 10 bit ADC supporting up to 20 Msamples/s whose data is then fed into a DSP. This DSP is capable of doing additional processing on the digitised signals like applying different baseline correction filters or compression algorithms. The SAMPA operation can be adapted with programmable parameters so that the chip can be used by two different detector systems, the muon chambers of the spectrometer and the TPC. [26]

The SAMPA can be used either with the integrated DSP enabled (and all its data processing capabilities) or in DAS mode. The TPC will use the latter one to overcome the issues of the not-applicable ZS due to the CM effect. In this mode, the raw ADC samples of the

words 0–7:	A	B	A	B	A	B	A	B
words 8–15:	A	A	B	B	A	A	B	B
words 16–23:	A	B	A	B	A	B	A	B
words 24–31:	A	A	B	B	A	A	B	B

(a) SAMPA synchronisation pattern in normal mode.

words 0–7:	A	A	B	B	A	A	B	B
words 8–15:	A	A	A	A	B	B	B	B
words 16–23:	A	A	B	B	A	A	B	B
words 24–31:	A	A	A	A	B	B	B	B

(b) SAMPA synchronisation pattern in split mode.

Figure 3.9: The SAMPA synchronisation pattern in DAS mode, using A for 0x2B5 and B for 0x14A, consists of 32 10 bit words. Figure (a) shows the pattern in normal mode while figure (b) shows it in split mode.

32 channels are multiplexed to the ten output pins and sequentially transmitted. Because the digital circuitry of the DSP is not needed it is powered down to reduce the power consumption. In this operation mode, two transmission modes are available which are using the ten available serial links in a different way, the normal mode and the split mode. In normal mode, the 10 bit ADC values of each channel are provided one-by-one to the output links. Starting with channel 0 in the first cycle, channel 1 in the next cycle and so on until channel 31 is reached. Then it is channel 0 again, but from the next time bin. In the split mode, the serial links 4–0 are used for channel 0–15 while the links 9–5 are used for channel 16–31. This is shown in table 3.1. Since the ADC value remains a 10 bit word, it must be split across two consecutive output cycles, always the five LSB first, then the five MSB as it is shown in the last two columns of the table.

In both transmission modes, the eleventh serial link is used to provide the internally generated ADC clock. With this it is possible to observe eventual phase shifts in the ADC clock, caused by single event upsets in the SAMPA internal clock divider network and one can act accordingly, e.g. by resetting all SAMPAs and resynchronise all FECs of the TPC. It has to be noted that there is no fixed relation between the clock on this port and the data on the other ports. That is why it can not be used to mark a specific channel. The channel must be reconstructed in a different way. The SAMPA sends a continuous stream of data in the DAS mode. To be able to identify channel 0 (and with that all following channels), a synchronisation pattern is used. This synchronisation pattern is sent once at the very beginning of the readout after the SAMPA receives a reset signal. The pattern consists of 32 10 bit words, thus the complete pattern has the same length as a readout cycle of all 32 channels. Two complementary values (0x2B4 and 0x14A) are used to build the pattern, which switch in the way shown in figure 3.9. The top part

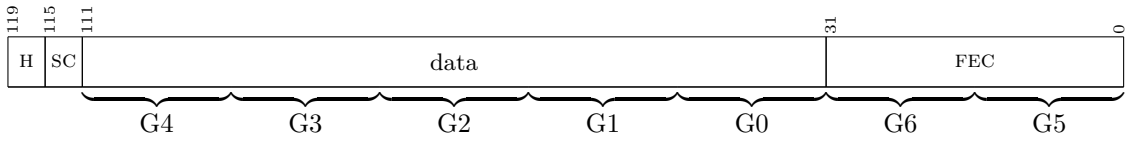


Figure 3.10: The GBT protocol, consisting of a 4 bit header (H), a 4 bit Slow Control (SC) field with 2 bit for the GBTx internal control and 2 bit for the external control of the GBT-SCA, a 80 bit data field and 32 bit for the optional automatic Forward Error Correction (FEC). The different data groups of the wide bus mode are indicated below.

shows the switching in the normal mode, the bottom part in the split mode. Although the patterns look quite different in the two modes, after de-interleaving the pattern of the split mode, they are both identical. The synchronisation pattern is not only used to identify the very first channel but also to synchronise the data from all SAMPAs of all FECs. Since they receive the reset signal at the same time (the signal propagation latency is deterministic in both directions), they also start all at the same time with sampling and sending the data [38].

3.3 The GBT System

The experiments at the LHC and other, e.g. future colliders, require high data rate links which can sustain high radiation doses. The GBT project [39] addresses this issue by providing a radiation hard on-detector ASIC, implementing a 4.8 Gbit/s bi-directional optical link between experiment and the CR. The counter part of the system is located off-detector in an environment without radiation and consists of an FPGA, programmed to be compatible with the GBT protocol, implementing the interface to further off-detector components. The on-detector components which are relevant for the TPC readout are the GBTx and the GBT-Slow Control Adapter ASIC (GBT-SCA). The GBTx is a serialiser/deserialiser chip. Its task is to provide the interface to the detector FEE and the encoding and decoding of the data into the GBT protocol. The serialised data is then transmitted at 4.8 Gbit/s. The GBT-SCA provides the Slow-Control (SC) interface. The chip is connected to dedicated pins of the GBTx and implements commonly used control busses like I^2C or JTAG. It can also be used to monitor environmental variables like temperatures and voltages [39].

3.3.1 The Transmission Protocol

The GBT protocol is a 120 bit wide frame, subdivided into four fields as shown in figure 3.10. 4 bit are reserved for a head and also 4 bit for the Slow-Control (SC) interface. The 4 bit SC field is further subdivided into 2 bit for the internal control of the GBTx itself and 2 bit for the external control of a SC interface, e.g. to connect the GBT-SCA via those dedicated output pins. Then there is a 80 bit wide data field and finally a 32 bit wide field for the optional automatic Forward Error Correction (FEC). On the receiving side, the frame is updated with a frequency of 40 MHz. Three different frame formats are available for the data transmission with the GBTx [40]:

GBT frame format: This is the default frame format. The 32 bit FEC field is used for an automatic forward error correction of the remaining 88 bit, based on a Reed-Solomon encoding. It was chosen in a way to provide a high level of error correction that is able to deal also with bursts of up to 16 consecutive wrongly received bits. The encoding is done before serialisation and decoding after deserialisation. In this way, any transmission errors or single event upsets can be corrected. This reliability in the transmission is important especially while transmitting control and trigger signals. Therefore only the 80 bit data field can be used for user data, resulting in a bandwidth of

$$80\,\text{bit} \times 40\,\text{MHz} = 3.2\,\text{Gbit/s}.$$

The TPC will use this frame format for the down-stream path, to configure the SAMPAs and send control signals to the FEC to use the advantage of the automatic error correction for the control path.

Wide bus mode: For applications were the bandwidth is more important than the possibility of an automatic error correction, the 32 bit FEC field can also be used for additional user data. This increases the available bandwidth by 40 % compared to the GBT frame format to

$$(80\,\text{bit} + 32\,\text{bit}) \times 40\,\text{MHz} = 4.48\,\text{Gbit/s}.$$

The TPC will use this format for the up-stream data path. Here, the bandwidth is most important and since raw ADC values without Huffman encoding are transmitted, single bit-flips are tolerable.

8B/10B frame mode: For completeness, also the third mode is mentioned although it is not foreseen to be used in the TPC readout scheme. On users request, the 8B/10B frame mode was added for the up-stream transmission. The 120 bit frame is divided into twelve 8B/10B* words of which eleven are available for user data since the first one is needed for the synchronisation on the receiver side. This leads to a bandwidth of

$$11 \times 8\,\text{bit} \times 40\,\text{MHz} = 3.52\,\text{Gbit/s}$$

which is only marginally higher than the GBT frame format with the disadvantage of the omitted forward error correction. The advantage of this mode is the reduced amount of resources needed for the FPGA implementation on the receiving side because the error correction is skipped.

3.3.2 The SAMPA Data within the GBT Frame

The electrical interface between the GBTx and the Front-End Devices (FEDs) is realised via so called eLinks. Each eLink consist of three signal lines, a clock line driven by the GBTx, a data down-link to transmit data from the GBTx to the device and an up-link to deliver data from the device to the GBTx. The setup of those lines, especially the data rate,

*In an 8B/10B coding, a 8 bit word is transmitted as a 10 bit binary string to achieve DC-balancing [41].

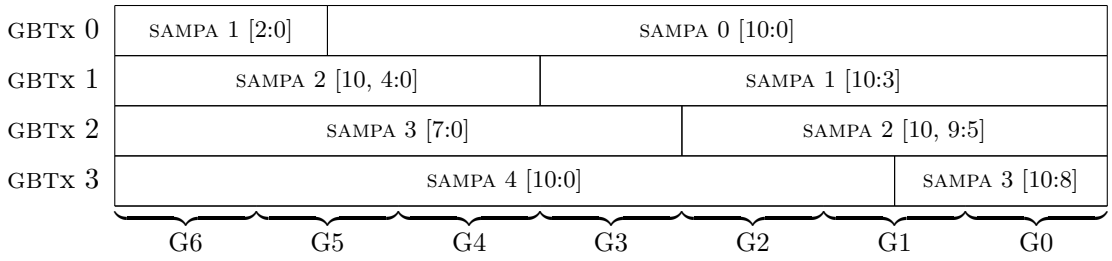
group	GBT frame bits
0	[47:32]
1	[63:48]
2	[79:64]
3	[95:80]
4	[111:96]
5	[15:0]
6	[31:16]

Table 3.2: The bits of the GBT frame of the individual data groups in the wide bus mode [40].

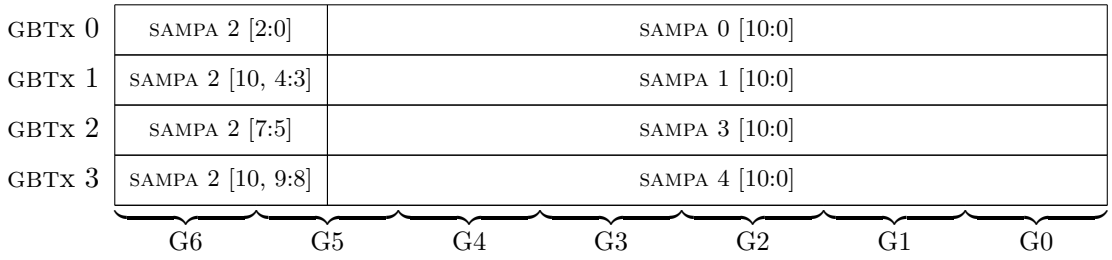
is programmable on a per group level and the number of eLinks available in each group depends on the data rate. Each group consists of four eLinks for a data rate of 160 Mbit/s. If the data rate is doubled, the number of eLinks is halved and vice versa.

The mapping of the input eLinks into the GBT frame is fixed and depends on the used frame format and the configured data rate. For the wide bus mode, this mapping between the eLink groups and the GBT frame is shown in figure 3.10. The exact bits of the frame belonging to the individual groups can be found in table 3.2. The TPC will use a data rate of 160 Mbit/s between the GBTx and the FED (the SAMPA chip). Therefore each group consists of four eLinks. Since the connections between the eLinks and the SAMPA ports are hardwired on the FEC, the location of the data of the individual output pins of the SAMPA within the GBT frame is purely determined by the layout of the FEC and therefore fixed during the design phase of the Printed Circuit Board (PCB). The connections were done in a way that an easy and straightforward decoding is possible. At the time when the discussion about the PCB layout took place, it was not yet decided to half the sampling rate for the TPC from 10 MHz to 5 MHz. However, the raw data readout was already settled. To cope with the expected data rates, a FEC version with 4 GBTx and a data rate of 320 Mbit/s between the GBTx and the FEDs was discussed, where the layouts figure 3.11a and figure 3.11b are coming from. For better readability, the mapping between the SAMPA and the groups is shown instead of the GBT frame bits. The ordering of the first version in figure 3.11a is straightforward on the PCB connection side. Port 0 of SAMPA 0 is connected to group 0 of GBTx 0 and then it is simply counted upwards with the port numbers of the SAMPA (and then with the SAMPA ID) on the one side and with the groups of the GBTx (and then with the GBTx ID) on the other side. The only exception is the eleventh port of SAMPA 2 which is routed to GBTx 1 as well as GBTx 2. This port transmits the ADC sampling clock which is needed for monitoring purposes. Since the two GBTx groups 0/1 and 2/3 are foreseen to be connected to different CRUs, this clock needs to be transmitted via both paths to be able to monitor the quality of the data independently in both CRUs receiving the data from SAMPA 2.

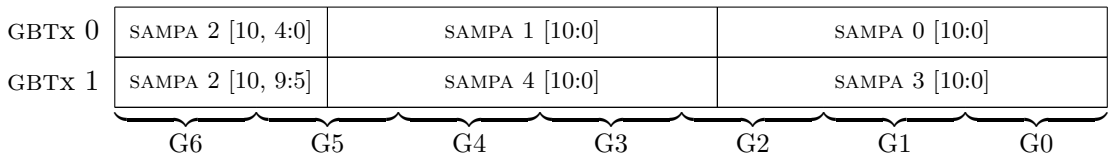
In version two, the ordering is done in a way that the GBT frame from the individual GBTx chips look very similar. This simplifies the decoding significantly. Since SAMPA 2 is anyway a special case because of the eleventh eLink, it was decided to split only the ports of SAMPA 2 across different GBTx chips, the data from the ports of all other SAMPAs are



(a) Mapping of all SAMPA ports into four GBT frames. The ordering is straightforward on the connection side, it was started with port 0 of SAMPA 0 connected to group 0 of GBTx 0 and then just continued counting upwards with the ports of the SAMPAs on the one side and with the groups of the GBTx on the other side. The only exception is the eleventh port of SAMPA 2 which is routed to GBTx 1 as well as GBTx 2.



(b) Mapping of all SAMPA ports into four GBT frames. The ordering is done in a way that each GBT frame look very similar to the others. Since SAMPA 2 is anyway a special case because of the eleventh port which needs to be routed to the GBTx 0/1 group as well as GBTx 2/3, it was decided to split only the ports of this chip across different GBTx. The ports of all other SAMPAs are sent by only a single GBTx chip.



(c) Mapping of all SAMPA ports into two GBT frames. After the readout frequency was reduced to 5 MHz there was enough bandwidth available to transfer the data of 2.5 SAMPAs via a single GBTx. Still the SAMPA 2 remains a special case. Otherwise version (b) was adapted to this solution.

Figure 3.11: Different versions of how to map the data from all five SAMPAs into the GBT frames. Version (a) and (b) originate from a time where the 10 MHz readout of the TPC was still the baseline but the raw data readout was already settled. Version (b) was then adapted to (c) after it was decided to reduce the readout frequency by a factor of two. The data groups of the GBT frame (G0–G6) which were shown already in figure 3.10 are aligned for a better readability.

kept within a single sampling chip. The layout of the SAMPA 2 connections is chosen in a way that the two CRUs receive a similar pattern. The decision was then made in favour of the second version because of two reasons:

1. Each frame of the different GBTx looks very similar which simplifies the decoding. To decode the frames of version 1, four different decoders would have had to be written for the different formats. SAMPA 2 is still a special case, but also this format is similar when comparing the two GBTx groups 0/1 and 2/3.

2. This format is more failsafe in case of phase-shifts of the different sampling clocks in the individual GBTx. These are still four individual GBTx ASICs, sampling the data from the SAMPAs independently. In version 1, three out of the five SAMPAs are split across two GBTx while in version 2 it is only the data of SAMPA 2, reducing the danger of data loss.

After the reduction of the sampling frequency for the TPC from 10 MHz to 5 MHz, the basic principle of the second layout was kept and adapted for a factor two less in data volume and therefore only two GBTx ASICs on the FEC. The resulting layout is shown in figure 3.11c. The frame from GBTx 0 looks exactly the same as from GBTx 1, only the origin of the content is different. Thanks to the split mode of the SAMPA, which was already introduced in section 3.2, even the mapping for SAMPA 2 looks exactly the same in both frames, making it possible to use the same decoder without any changes to decode the frames from both GBTx chips of each FEC. The reduction in readout frequency by a factor of two went along with a reduction in the data rate between the GBTx and the FED by the same factor. As a consequence, now each group contains the data of four eLinks instead of only two. With that, the data of 2.5 SAMPAs fit into one frame instead of only from one SAMPA and a quarter of the ports from SAMPA 2.

To fill the user data of the GBT frame of 112 bit, the GBTx concatenates multiple *time bins* of the input ports into a single frame. For the 320 Mbit/s case, eight time bins of each eLink would have been put into a single frame. For the 160 Mbit/s case, this is reduced to four. The exact layout of how the individual time bins are filled in the GBT frame and how the decoding is done will be discussed in subsection 5.2.1.

3.4 The Common Readout Unit

The data of the TPC FECs is received by the CRUs. Those units act as the interface between the on-detector systems, the Central Trigger Processor (CTP) which provides trigger information and the LHC clock and the computing farm for the Detector Control System (DCS), further processing and data storage [42]. The CRU was originally developed by LHCb for their readout upgrade in perspective of LHC Run 3 under the name PCIe40, but will be used in ALICE as well. The main component for the processing logic is an Intel Arria10 FPGA, which is one of the most powerful FPGAs currently available on the market, with 1 150k Logic Elements (LEs) [43, 44]. A complete overview of the available resources of the FPGA is given in table 3.3. The most important resources for the later development are beside the huge amount of LEs, the amount of memory which can be stored in the M20Ks of 54.260 kbit and the number of DSP blocks

Product line		GX 1150
LEs		1 150k
ALMs		427 200
Register		1 708 800
Memory (kbit)	M20Ks	54 260
	MLABs	12 984
Variable-precision DSP blocks		1 518
18 × 19 multipliers		3 036
PLLs	Fractional Synthesis	32
	I/O	16
17.4 Gbit/s transceivers		96
GPIO		768
LVDS pairs		384
PCIe hard IP blocks		4
Hard memory controller		16

Table 3.3: Resources of the Intel Arria10 (10AX115S3F45E2SG) FPGA of the CRU, taken from [45].

of 1 518. The memory corresponds to 2 713 individual M20K RAM blocks. An image of the CRU is shown in figure 3.12, where the FPGA is nicely visible in the centre of the card.

The CRU is, as the name suggests, a common solution for many of the detectors in ALICE to interface the data acquisition system. The card provides up to 48 bidirectional optical links of which the TPC will use up to 20 to connect to the FEE. The number of links, which is equivalent to the number of connected FECs, used in the individual *readout regions* is given in table B.1 and ranges from 15 to 20. It must be noted that each CRU receives the data of only half a FEC, as already shown in figure 3.1, or phrased the other way around, each FEC is read out by two different CRUs. Since the FEC needs to be controlled by one single master, the relationship between a FEC and its two CRUs is not symmetric. One is the master which implements the up-link (for data readout) and the down-link (for control) via the VTRx and GBTx 0, while the other CRU is the slave, implementing only the up-link to receive data via the VTTx and GBTx 1. The communication with the FECs is done using the GBT protocol in wide bus mode. Therefore, the maximum data input rate to a single CRU sums up to a total amount of

$$20 \times 112\,\text{bit} \times 40\,\text{MHz} = 89.6\,\text{Gbit/s} \tag{3.5}$$

which needs to be either compressed or transferred via the PCIe bus to the host machine, the FLP. Here, the data will be further processed and sent via a high performance network connection to the Event Processing Nodes (EPNs) where the data of multiple FLPs is combined and a calibration and the tracking can take place. Another optical link is implemented to receive the trigger information from and send status information to the CTP. This is the Trigger, Timing and clock distribution System (TTS) link, providing the baseline LHC clock as well as the experiment wide Heart Beat (HB) signal and further

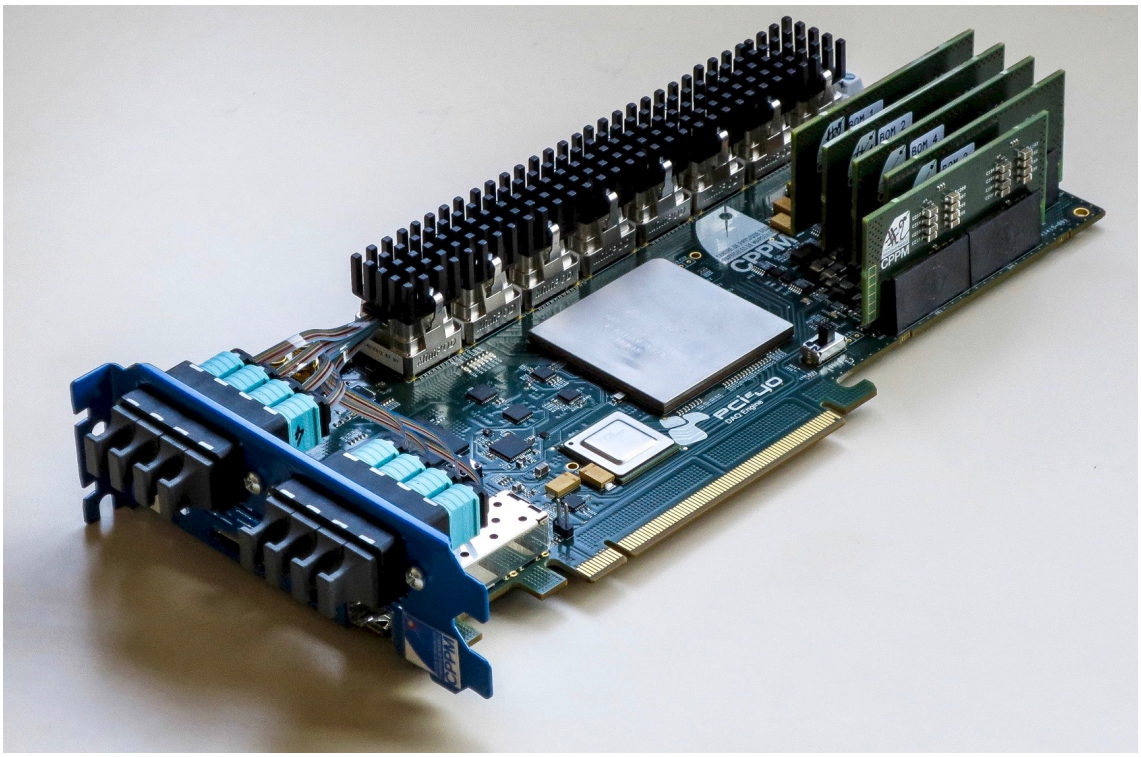


Figure 3.12: Image of the CRU version 1. The heat-sink of the FPGA was removed so that the Arria10 is visible. The optical components of the 48 bidirectional links are located below the still mounted heat-sinks on the back. The vertical PCBs on the right side are needed for the power supply of the card and were redesigned in future versions of the CRU. This photo was kindly provided by [46].

trigger and timing signals. The HB signal is used to synchronise all electronics of the whole experiment and is issued every 89.4 μs.

The interface between the host machine and the CRU is PCIe generation 3 with 16 lanes providing a practical sustainable bandwidth of 90 Gbit/s [42]. This means that the TPC will be able to also dump the raw data to the FLP, at least for short time periods. This possibility to examine the raw GBT frames in software will be needed for calibration and debugging purposes. Though, in normal data taking, the output data rate of the CRU is expected to be at least a factor five less. This reduction is achieved by a Cluster Finder (CF) running on the FPGA. This CF achieves the compression factor of five by first applying an intrinsic ZS, because only physically relevant information is taken into account, and second by converting the data into the Cluster Data Format (CDF). Since the average pad occupancy is expected to be at most 30 % [5], the ZS will give a compression of at least 3.3, while the conversion into the CDF gives another factor of $^{250}/_{160} \approx 1.56$, because the 250 bit of all contributing ADC values to the cluster will form a single cluster word with a size of 160 bit. Details can be found in section 5.3. This rough estimation leaves out the possibility of overlapping clusters, nevertheless, it gives a first idea about the maximum expected output data rate of

$$89.6\,\text{Gbit/s} \times 0.3 \times \frac{160}{250} = 17.2\,\text{Gbit/s} = 2.15\,\text{GB/s} \tag{3.6}$$

per CRU after applying the CF. This compression factor depends on the occupancy in the TPC. If the occupancy increases, more clusters are found and the data rate increases as well. The factor is further enhanced by additional postprocessing steps in the FLP by a reformatting of the data and a Huffman encoding.

In general, there are different readout modes foreseen to be implemented in the CRU. Each mode serves a different purpose. The default mode is the clusterised data output, mainly used in normal physics. Here, the data volume is the lowest while keeping all physically relevant information. For debugging and calibration tasks, it must also be possible in the final system to write out the raw GBT frames or data from the intermediate steps of CF processing to the FLP, coming along with the price of an increased data volume [42].

Location of the CRUs

The CRUs will be located off-detector in the CRs. Together with the FLPs they will be located in CR 1 [47], which is the uppermost one. Those rooms are placed in the shaft which is going down to the experiment, as can be seen in figure 3.13. Since they are sitting behind the shielding, they are outside of the radiation area. This facilitates the development of the hard- and firmware, as no special attention has to be paid to a radiation-hard design.

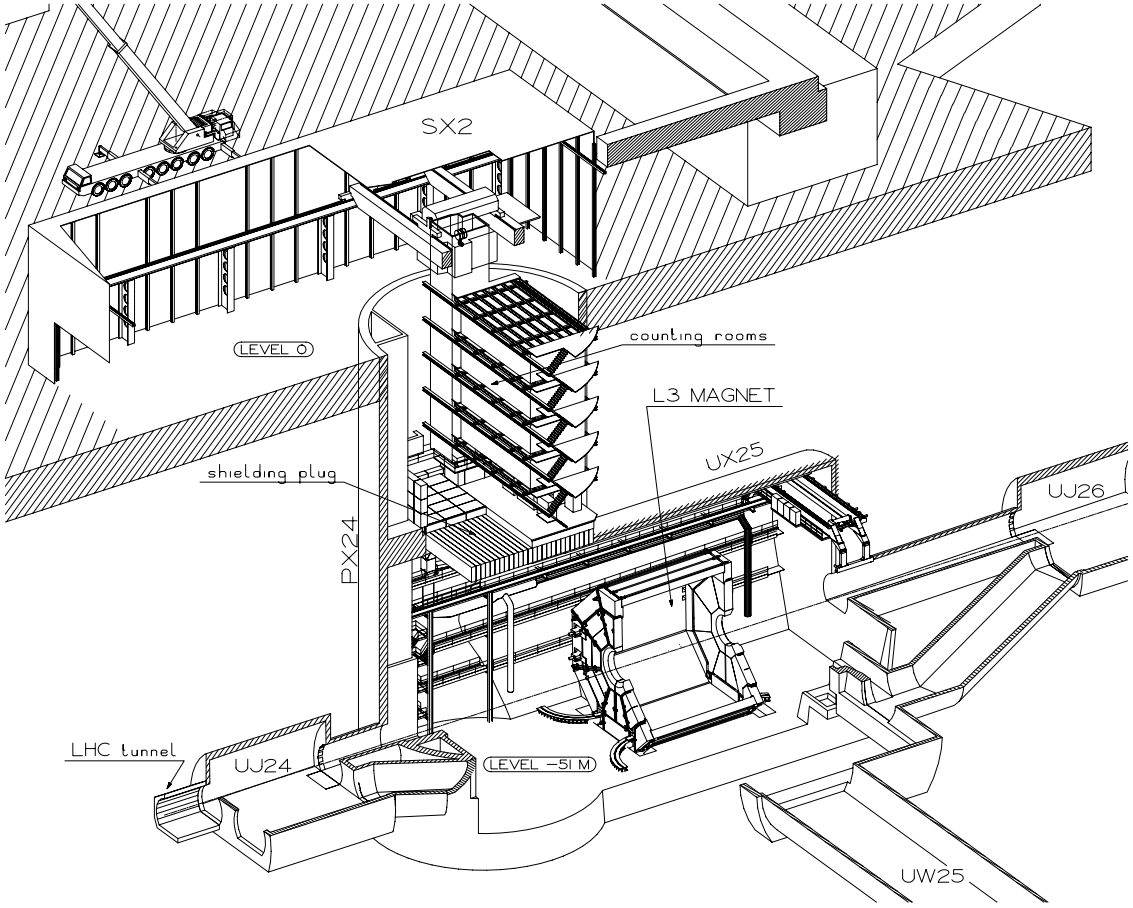


Figure 3.13: Layout of the ALICE underground area. The L3 magnet is shown as well as the CRs, taken from [29].

Chapter 4

The CRU Firmware

Before starting with the Cluster Finder (CF), which is the core of the processing logic of the TPC, an overview is given of the general layout of the FW design for the CRU. Since many different detectors in the ALICE experiment with individual requirements will use the CRU in Run 3, the FW design structure must be able to cope with this diversity. Some of the detectors need the CRU just to interface the data acquisition and the Detector Control System (DCS) — they do not need the processing capabilities of the available FPGA — while others need very specialised functionalities like the cluster finding and additional preprocessing steps as it is the case of the TPC. So the FW must be structured in a modular way with clearly defined interfaces. This chapter explains first the basic layout of those modules and then describes the interfaces.

4.1 A Modular Firmware Concept

In order to cope with the variety of different requirements, the FW is designed in a modular way. Modules which will be needed by most of the detector teams are developed centrally and can be used by the individual detectors if they fit their needs. The detector specific part is then combined in the User Logic (UL) which falls completely into the responsibility of each individual detector team. A schematic layout of the main building blocks of such a design layout is given in figure 4.1. As can be seen, the modules besides the UL are mainly interface modules and, not shown here, some small helper modules for individual tasks like Clock-Domain Crossings (CDCs) or the configuration of the UL. There are four interfaces which must be implented, first the one to the Central Trigger Processor (CTP), called the Trigger, Timing and clock distribution System (TTS) interface. This module provides the trigger information and the baseline clock with a frequency of 240 MHz (called the TTCrx clock in the following) for the design and delivers status information back to the CTP. The connection to the CTP is done via TTC-PON links and goes through the Local Trigger Unit (LTU). The LTU is a hardware interface module, provided by the CTP to have a uniform interface to all the detectors. The second interface, which is split into two separate ones, is the one through the PCIe-bus with the First Level Processor (FLP). One part is the DMA engine, taking care of shifting the big data volumes of the detector readout from the CRU to the FLP memory. The other part is the configuration and DCS path, also realised

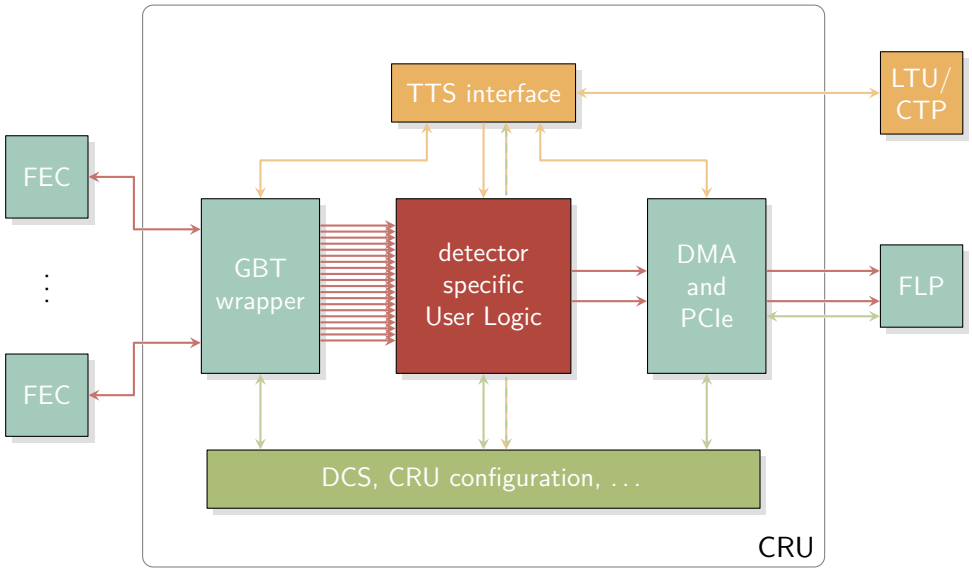


Figure 4.1: The main blocks of the CRU FW. They can be roughly categorised into the data path (blue and red), which is going from the FECs on the left side via the GBT wrapper, the detector specific User Logic and the DMA engine to the FLP on the right side, the configuration and DCS modules shown in green and trigger and clock distribution in yellow.

through PCIe. The fourth and last interface is towards the Front-End Electronics (FEE). Since most of the detectors which use the CRU will also use the GBT system and with that the GBT protocol, there is a wrapper available to encapsulate the individual GBT core modules of each single link into a bigger element. The GBT core modules are developed by the GBT group together with the corresponding ASICs. This wrapper implements a uniform interface and takes care of the CDC of the individual links into the common TTCrx clock. The individual GBT cores use internally a clock that is recovered from the data stream to deserialise and decode the data to be able to provide the frame format of the GBT protocol. Since the next module in the data path is the UL which combines the information from several links (even if it is just forwarded to the FLP's memory without any changes) the data has to be in a common clock domain. Thus it makes sense to implement the crossing already at the very beginning.

Clock-Domain Crossings

Before continuing, first a few general comments about CDCs are given. Whenever there is a signal crossing two domains with an asynchronous or unrelated clock, one has to deal with the effect of metastability. In a metastable state, the output of a register is not clearly defined. In FPGAs, or rather all digital devices, the registers have defined signal timings. With that it is possible that the input signal is captured correctly and the output is produced accordingly. The important timings here are the register setup time (the minimum time the input must be at a stable state before the clock edge) and the register hold time (the minimum time the input must still be stable after the clock edge). If those

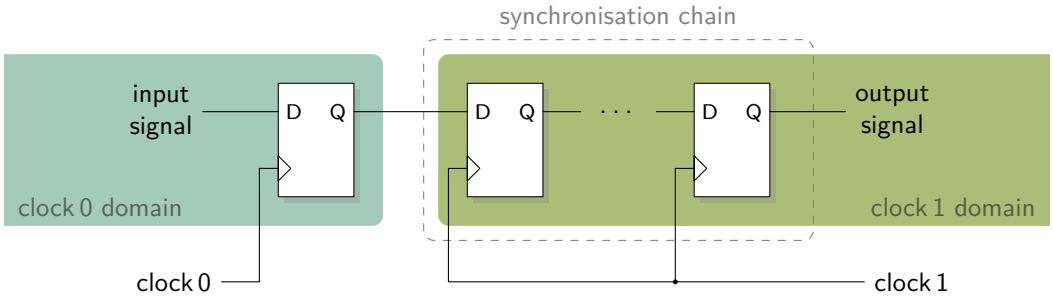


Figure 4.2: Synchronisation register chain. The input signal is synchronised from the clock 0 domain into the clock 1 domain with a set of successive registers.

timings are violated by the signal transition, the signal might be sampled incorrectly and the output of the register might be metastable. If this output signal is then used further, its state is not deterministic and can lead to unintended effects in the logic. In a synchronous design, the signals must always meet the register timings to avoid metastability. Usually the fitter which compiles the code takes care of achieving all timing requirements and reports paths that violate the conditions so that the developer can examine the logic again [48].

Metastability issues mostly occur for signals crossing domains with different, unrelated or asynchronous clocks. In such a case it can not be guaranteed that the timing requirements are always met by the signals. There are various different techniques to minimise the failures due to metastability issues, depending on the actual use case. The most commonly used method to transfer asynchronous signals is via a synchronisation register chain. This is a sequence of registers where all registers in the chain are clocked with the same clock (or a phase-related one) except for the very first register which is located in the unrelated clock domain. Each register in the chain, except for the last one, fans out to only a single other register. This concept is shown in figure 4.2 where the first register is in the domain of clock 0 and all others in the domain of clock 1. The length of the chain can be extended for a better metastability protection but is required to having a length of at least two registers. The first register in the chain will have an unpredictable, eventually metastable, output which is then recovered by the second one in the chain, assuming there is enough time for the metastability to settle down. Otherwise, more registers have to be added to improve the synchronisation behaviour but also adding additional latency to the signal propagation time. This concept is simple and works pretty well for clocks with either the same frequency or when the frequency of the receiving clock domain is faster than the one from the sending domain. It has to be noted that this kind of synchronisation should be used only for individual control signals and not for data buses with several correlated bits. For a rising edge in the data input signal, the three possible outputs of the first stage of the synchronisation register chain are either low, metastable or already high, depending on the relation between the arrival time of the signal and the sampling clock at this very moment. So if one would try to synchronise a set of signals via individual synchronisation register chains, some of the signals might be sampled earlier than others of the same set of signals which messes up the output although all signals were synchronised correctly, because the individual latencies and, with that, the states can be different. For

synchronisation applications with several correlated bits one should realise the CDC in a different way. To name just two standard approaches, there would be a handshaking mechanisms or the synchronisation via dual-clock FIFOs.

In a handshaking mechanism, the correlated data signals at the input are kept constant until the message arrives that they were correctly synchronised into the target clock domain. For this purpose an additional control signal is used, informing the receiving side that new data is available. After the data signals are sampled in the target domain, another control signal is sent back to the source domain to release the data there. With this approach only the two control signals have to be synchronised into the respective clock domains, for example via synchronisation chain registers, while ensuring that an arbitrary wide data bus is correctly synchronised. Since the control signals have to go forth and back over the domain, there is some, eventually not negligible, latency for the handshake.

A dual-clock FIFO is a FIFO were the input ports can be clocked with a different clock than the output ports. With this, the data can be transferred from one clock domain into another. The data is then written and read from a RAM while internally the available read and write addresses are properly synchronised from one clock domain into the other (for example by using Gray-Code counters*) and the flags for both interfaces, like full and empty flags, are generated directly in the correct clock domain. Those dual-clock FIFOs are usually provided as soft (synthesisable code, provided in a hardware description language) or hard (not changeable) IP cores by the vendors, ready to be used.

4.2 Interfaces to the User Logic

It is very important to clarify the interfaces to the UL before going into the details. Those interfaces are the connections to the outside world and must be fixed at an early state of the development process, since many details of the further implementation depend on this. Therefore, it is explained in detail in the following. For the data path, the important interfaces are to the GBT wrapper to receive the GBT frames and to the DMA engine to send the processed results. The interface to the TTS module provides a base clock and all necessary trigger information. In the following, the word input is used for paths going into the UL, while output describes paths going out of the UL.

4.2.1 GBT Wrapper

```
-- GBT down-link (CRU -> FE)
gbt_tx_ready_i  : in  std_logic_vector(23 downto 0);
gbt_tx_bus_o    : out t_cru_gbt_array(23 downto 0);
-- GBT up-link (FEE -> CRU)
gbt_rx_ready_i  : in std_logic_vector(23 downto 0);
gbt_rx_bus_i    : in t_cru_gbt_array(23 downto 0);
```

Listing 4.1: VHDL interface to the GBT wrapper for 24 links.

*The Gray-Code, named after Frank Gray, is a different representation of the binary numeral system in such a way that successive values differ by only one bit. Because of this, slightly different timings in the individual bits do not lead to short-lived wrong states, the value is changed only slightly later or earlier.

The clock domain of the interface to the GBT wrapper is the already mentioned TTCrx clock. The interface itself consists of two almost identical parts, the TX part for the down-link (CRU → FEE) and the RX part for the up-link (FEE → CRU). For both of the paths there is an input vector providing a status bit for each link. This bit indicates the readiness of the specified link for either a down-link communication or to receive data. In addition there is the data bus for both, up- and down-link. These buses consist of an array of records, one per link, containing all the relevant information. Each record contains two individual bits to indicate the validity of the current data. One flag for the general status information and one flag which is valid one out of six Clock Cycles (CCs) to indicate that the data can be used. This ratio comes from the ratio of the two involved clocks, the 240 MHz in which the interface is implemented and the basic 40 MHz clock of the GBT protocol. Additionally, there is the 4 bit Slow-Control (SC) field available and a 112 bit vector combining the data field and the FEC field. Since the TPC is using the GBT in wide bus mode, all the 112 bit are relevant. It has to be mentioned that the ordering of the data within this vector follows the group ordering and not the bit ordering of the GBT frame (see figure 3.10). This means that the normal data field is located in the bits [79:0] and the additional 32 bit of the FEC field can be found in the bits [111:80]. That is a very important detail for the usage of the data in wide bus mode within the UL [49].

4.2.2 DMA Engine

```
-- Endpoint 0
FCLK0 : out std_logic;
FVAL0 : out std_logic;
FSOP0 : out std_logic;
FEOP0 : out std_logic;
FD0   : out std_logic_vector(255 downto 0);
-- Endpoint 1
FCLK1 : out std_logic;
FVAL1 : out std_logic;
FSOP1 : out std_logic;
FEOP1 : out std_logic;
FD1   : out std_logic_vector(255 downto 0);
```

Listing 4.2: VHDL interface to the data path wrapper.

The interface to the DMA engine is implemented via a data path wrapper in between. Therefore, the DMA engine is not directly accessible from the UL. The data path wrapper selects the data streams which are transmitted to the FLP (could be the output of the UL or directly the individual GBT links) and does the flow control. Nevertheless, the actual hardware is still reflected in the interface to the wrapper by providing the two PCIe endpoints of the CRU individually.

For both endpoints the interface is implemented as a dual-clock FIFO. In this way the clock used in the UL, or at least the clock used to write the output data, is independent of the clock used to transmit the data via the PCIe bus. The user has only a basic FIFO interface to write data from the UL to the host machine: a clock `FCLK0/1` needs to be provided which is used to write to the FIFO, a valid flag `FVAL0/1` to enable the writing and

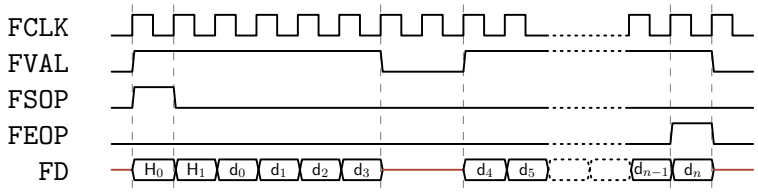


Figure 4.3: The transmission protocol towards the data path wrapper. The data stream consists of two header words, followed by the actual data. The SOP flag must be high for the very first word of a package, while the EOP marks the very last one. The valid flag is used to specify when the data is to be used, based on [49].

FD[31:0]	FD[63:32]	FD[95:64]	FD[127:96]	FD[159:128]	FD[191:160]	FD[223:192]	FD[255:224]
0	1	2	3	4	5	6	7

Figure 4.4: Mapping of the data bus into the FLP memory. The 256 bit data words are chopped into pieces of 32 bit of which the least significant one appears first in memory.

the data bus FD0/1 containing a 256 bit data word which is written. There are no full or empty flags provided as they usually can be found in a FIFO interface since the data path wrapper checks internally the level of the FIFO and informs the CTP via a dedicated path in case of an overflow to synchronously drop the data across all CRUs. Two more bits need to be set, the Start-Of-Packet (SOP), in the interface description marked with FSOP0/1, and the End-Of-Packet (EOP), in the interface description marked with FEOP0/1. They are needed by the protocol shown in figure 4.3, which must be complied with. As usual, the valid flag marks when the 256 bit data word is valid and is used as a write enable. So only those data words with the valid high are transferred to the host memory. Therefore, the words do not have to be concatenated, a valid sequence may also have gaps in between. The SOP marks the beginning of a new packet, starting with the first word of the in total 512 bit wide Raw Data Header (RDH), while the EOP marks the end of the packet after which a new packet must be started, again with the SOP. Everything in between belongs to the same header, regardless of how often the valid signal has toggled [49]. A detailed description of the data header and its fields is given in appendix C.

The RDH is actually a sequence of four 128 bit words as can be seen in figure C.1. Since the bus towards the data path wrapper (and via the PCIe) has a width of 256 bit the headers H_0 and H_1 (the first two data words in figure 4.3) must be composed of two RDHs words each. It must be noted that the data is represented in *little endian*, so the bus FD[255:0] is mapped into the FLPs memory as a sequence of eight 32 bit words in the order shown in figure 4.4. Thus, requiring that RDH 0 appears in the memory before RDH 1, the lower bits of H_0 must belong to RDH 0 and the higher bits to RDH 1, which is maybe counterintuitive when just looking at the protocol. The same is true for H_1 with RDH 2 and RDH 3.

It is also possible that a package contains no data at all and consists only of the two header words. Then, the SOP flag is set as usual for H_0 and the EOP must be set for H_1. This might be useful to finalise a series of packets belonging to the same Heart Beat (HB)

trigger via the stop bit of the corresponding header field. More details about the HB can be found in subsection 5.3.5.

4.2.3 TTS Interface

```
TTC_RXCLK    : in std_logic;
TTC_RXRST    : in std_logic;
TTC_RXREADY  : in std_logic;
TTC_RXVALID  : in std_logic;
TTC_RXD      : in std_logic_vector(199 downto 0);
```

Listing 4.3: VHDL interface to the TTS module.

The main signals coming from the Trigger, Timing and clock distribution System (TTS) are the clock and the trigger signals as well as global control commands. This is also reflected in the interface towards the corresponding module. The baseline clock for the UL, the 240 MHz TTCrx clock, is received from the `TTC_RXCLK` port. There is also the possibility to send a global reset, `TTC_RXRST`, and a ready flag to indicate the overall validity of the signals, `TTC_RXREADY`. The data bus `TTC_RXD` delivers the trigger pulses and bunch crossing information to synchronise the readout of all the CRUs from all detectors. `TTC_RXVALID` marks, when this bus is ready and the content can be used. The meaning of the individual bits of the data bus is described in [50]. The main signal, currently used by the UL from the TTS, is the HB signal which is located in bit 1 of the `TTC_RXD` bus [49].

Chapter 5

A 2D Cluster Finder for the TPC

The following chapter covers the main topic of this thesis, the development of the CF for the TPC. This CF will run on the FPGA of the Common Readout Unit (CRU) and reconstructs the charge clusters in real-time during the readout. This is the most challenging and resource consuming part of the Firmware (FW) design. There are various steps before the actual charge clusters can be found on a representation of the TPC pad plane in the CRU, like the decoding of the GBT frames, resorting of the individual pads based on the configuration of the readout card and the Baseline Correction (BLC). All those individual modules are covered in the following sections.

5.1 Overview of the TPC User Logic

The User Logic (UL) is the detector specific part in the CRU. For the TPC it implements mainly a CF. But there is more which needs to be done in order to provide usable ADC values to the CF network and to store the final clusters in the memory of the First Level Processor (FLP). An overview of the main building blocks is given in figure 5.1. The processing chain can be subdivided into two parts, a data preparation part and a cluster reconstruction part. They are logically decoupled and placed after each other. There is one fundamental principle behind the overall design of the UL: keep the data as separated as possible, do not merge paths unnecessarily. The TPC UL needs to process the data of up to 1 600 individual pads. Each pad provides a 10 bit ADC value, which will be a 14 bit Fixed-Point (FP) number, in the following shown as 10.4 FP number with 10 bit for the integer part and 4 bit for the decimal part, after the BLC. The four additional bits are needed for a sufficient precision of the correction. This sums up to 16 000 (22 400) paths which need to be routed together once they are merged. To reduce the effort for the fitter during the compilation of the design and to achieve timing closure, the individual parts which can be kept separated must also be kept separated. This is realised by doing the preparation up to the sorting for each input link individually. The links are completely separated and do not influence each other. Since clusters need to be found within single rows of the pad plane, the processing is done after the sorting for the rows independently. To reduce the complexity further for the single CF instances, each row is subdivided into smaller row-segments with a width of only a few pads. Here, each CF instance can find

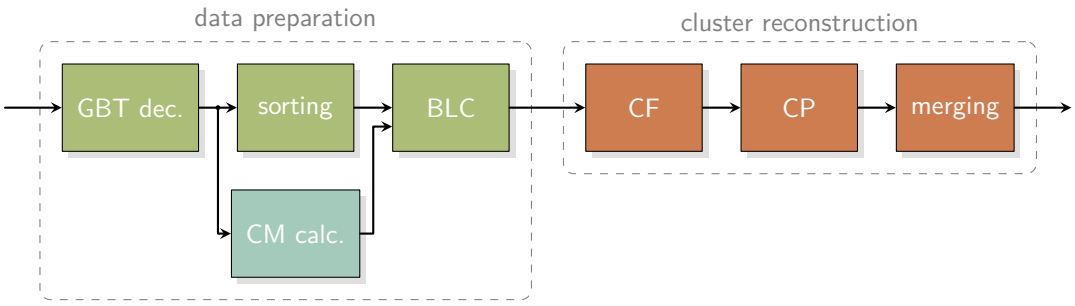


Figure 5.1: A simplified block diagram of the TPC UL, showing the main building blocks, subdivided into the data preparation and cluster reconstruction parts. The data preparation part consist of the decoding of the GBT frames, a sorting algorithm to reassemble a pad plane representation and the BLC for which the CM contribution must be calculated. To reconstruct the clusters, they must first be found with the CF after which the CP can calculate the cluster properties. In the end, the data from all CP instances must be merged to transfer them to the FLP.

clusters autonomously in its individual pad plane range. The data is merged only at the very end were it is absolutely necessary to write all clusters to one of the two output FIFOs of the Direct Memory Access (DMA) engine. An inevitable crossing occurs in the sorting module where the data of all links have to be combined to reassemble a pad plane representation from which the rows are extracted and separated again.

5.2 Data Preparation

To prepare the data in a way that charge clusters can be found, three basic steps need to be done. First, the GBT frames must be decoded. On average eight GBT frames are needed to get all channels of one time bin. Afterwards, the channels must be sorted. The data arrives on each link from a different Front-End Card (FEC) in the order of the SAMPA channels. To reassemble a representation of the pad plane, the mapping of the individual SAMPA channels to a specific pad position needs to be applied. This process is very costly because the mapping is different for each FEC in each readout region hence for each CRU in a sector. There are two basic approaches to overcome this issue, either leaving the mapping completely free (or at least as free as necessary, that one of the ten different mappings can be selected) via the configuration of the CRU, or having a fixed mapping for the ten different regions and compiling ten versions of the FW. The first approach leads to huge routing matrices, since in the worst case, each of the 1 600 input channels could end on a different pad in each of the ten regions. With a 14 bit number, this are 224 000 possible paths which all have to be realised in hardware and made selectable.

On the other hand, having the mapping fixed reduces this number significantly. However, one has to put way more effort into maintaining the whole system. After a single change, ten different FWs have to be compiled. This introduces first computational complications as the compilation time for a single FW for the Arria10 is around 10 h as soon as the FPGA becomes relatively full. This can be targeted with a parallel compilation on different

machines. Though, one has to keep in mind that a rather big machine is needed to achieve those compilation times. Intel recommends on its website [51] to have at least 18–48 GB of physical RAM to compile a FW for the Arria10 and the machine which achieved the compilation time of 10 h had more than 16 CPU cores, which is the maximum number the software is currently able to utilise for the overall compilation procedure [52]. Then again, these are just technical details which could easily be solved if the need exists. The bigger issue is that in fact ten different FWs must be compiled, with the danger that for some of them it is maybe harder to achieve timing closure because of a more complex logic and a different routing. This would perhaps require a separate treatment of the designs in also other aspects, than just the different mapping. This makes it very hard to achieve a homogeneous system in the end. Therefore, it was decided to have of a more complex design and a configurable mapping which fits all regions with the advantage to have just one FW. How this is achieved is described in subsection 5.2.2.

After the sorting is done, the baseline must be corrected. The correction includes three different components, a pedestal subtraction to subtract the offset of the electronics, a gain correction to compensate for gain variations in the detector and the Common Mode (CM) correction which was already discussed. The pedestal value is different for each pad and needs to be subtracted from the ADC values. Its origin lies in the electronics, while the CM is due to the capacitative coupling of the individual pads and results in a global downwards shift of the baseline and therefore has to be added to the ADC value. The gain correction, however, is a multiplicative adjustment and corrects for non-uniformities of the amplification in the Gas Electron Multiplier (GEM) stack. The BLC is done only after the sorting because it increases the data volume. A 10 bit integer value is expanded to a 10.4 FP number, which would require the sorting of 40 % more signal paths if applied beforehand.

In parallel to the sorting, the CM calculation is done. This module calculates the average ADC value of all pads without a signal peak. This does not require a sorting in advance because all pads are summed up, independent of the location. The peak exclusion is done as a function of the SAMPA channel instead of the pad. In this way, the peak is detected only in time direction but it was seen in the studies about the CM correction that this is sufficient [33, 53]. Before the summation can be done, the pedestal value needs to be subtracted as well. This can also be done as function of the SAMPA channel, but it has to be assured that the same value is used in this part of the design as in the actual BLC module. The gain should not be corrected before since the origin of the CM is the signals after the amplification in the GEM stack, so the gain variation is part of the CM signal.

The processing is done in a pipelined way. The data comes in sequentially, channel after channel (pad after pad subsequent to the sorting), and is processed in this order. There is no buffering needed in between. Due to this design approach, the data flow is completely deterministic and a time information can be assigned only later, during the peak fining, when it is needed. As a reminder, there is no time information sent together with the raw data from the FECs, this needs to be added within the CRU. If the time would have to be added at the very beginning, one would need to propagate this information properly throughout the whole design together with the data. However, since the latency is deterministic, there is only a constant offset which can be added also later in software. The CM calculation is an exception in this sense because here at least two time

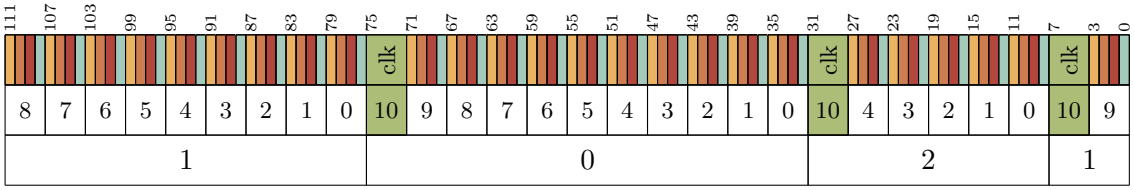


Figure 5.2: Detailed mapping of the SAMPA output into the GBT frame (without header and SC). The first line shows the individual bits of the output, with the time ordering highlighted (yellow first, blue last). The same scheme applies also for the three clock fields marked in green. The second row shows the SAMPA eLink output number and the SAMPA chip ID is given in the third row. The SAMPA informations are given for even region numbers.

bins are needed for the peak finding and the exclusion. Then a buffering is needed and done locally. Since this module is not part of the actual processing chain — it receives a copy of the ADC values, does the CM calculation independently of the other logic and provides a single value to the BLC module where it is applied — this is not an issue. The only condition is that the result of this module, the average ADC value of all pads without a signal peak for each individual time bin, arrives at the right time in the BLC module.

5.2.1 Decoding of the GBT Frames

The very first module of the TPC UL is the GBT decoder. This module is instantiated for each input link separately, it thus receives the data from one half of a FEC, or from 2.5 SAMPAs. In section 3.2 it was already discussed that the TPC will use all SAMPAs in the same configuration, the Direct ADC Serialisation (DAS) mode with the split mode transmission, to be able to decode the data from all sources in the same manner. Therefore, the ADC sampling clock of the three source SAMPAs as well as the five Half-Word (HW) sequences are contained in each GBT frame. Figure 5.2 combines the information of figure 3.10 and figure 3.11c and indicates the purpose of the individual bits of the frame. The figure shows the mapping for the even regions. The odd regions are similar, but SAMPA 0 is replaced with 3, SAMPA 1 with 4 and the eLinks 0 to 4 of SAMPA 2 are replaced with the eLinks 5 to 9. The ordering of the eLinks within the frame is purely determined by the FEC layout. It depends only on how the SAMPAs are connected to the input groups of the GBTx ASICs. It can be seen that each eLink contributes 4 bit to the frame. These are temporally consecutive output cycles of the SAMPA where the MSB (marked in yellow) is the first one and the LSB (marked in blue) the last one. The same applies for the clock fields which are marked in green to guide the eye. All bits of a SAMPA with the same colour belong together and form in total five HWs. The rearrangement to combine the same coloured bits to form the individual HWs is done in the *HW assembling* block of figure 5.3. Afterwards, each of the five HW sequences can be analysed independently.

The fields containing the 5 MHz sampling clock of the SAMPAs are extracted separately and monitored by a dedicated module, again independently for each of the three sources. The input ports of the GBTx ASIC are sampled with a 160 MHz clock. If a 5 MHz clock

is sampled with 160 MHz, the result will be a periodic pattern with a length of 32 bit where the signal is expected to be low for 16 bit and afterwards high for also 16 bit. Is this pattern now chopped into pieces with a length of 4 bit to transmit it via the GBT frame, the resulting (correct) sequence can only be one of the four possibilities shown in figure 5.4. To monitor the ADC clock, it is sufficient to look for the pattern with the rising edge (the second field in each sequence) and check that the following sequence is as expected. This can easily be implemented in a simple Finite State Machine (FSM). The FSM remains in the ground state until one of the first transition patterns is seen, locks to the corresponding sequence number and just passes through the following eight states. In each state the expected pattern is checked and if the input deviates, the FSM returns to the ground state. If all states are passed through once successfully, the ADC clock was correctly recognised. After that, the FSM continues to look for the next pattern of the same sequence and any deviation is reported as an error. A higher level of Detector Control System (DCS) monitoring modules can then act accordingly, e.g. by sending a synchronous reset signal to all the FECs and restarting the readout after accumulating some critical number of failures.

Decoding the other HWs to reassemble the ADC values is a bit more tricky. As already mentioned in section 3.2, the SAMPAs in DAS mode send a continuous stream of data without any header information. The only way to tell which channel is currently received is the synchronisation pattern (figure 3.9) at the very beginning. So this pattern must be recognised and afterwards the channel can be determined by counting the number of received values. The channel number follows the ordering given in table 3.1 for the two HW sequences of the split mode. The detection of the synchronisation pattern is realised in the same way as the monitoring of the ADC clock: a FSM goes through all its eight states and checks if the expected pattern is received. The FSM needs only eight states instead of 32, which is the length of the synchronisation pattern, because four HWs are received at once (the four different colours of figure 5.2), analogously to the four temporally consecutive bits of the ADC clock. This has the same consequence that the synchronisation pattern can start with one out of the four simultaneously received HWs. The simplest case is when the pattern starts at position zero, which is shown in figure 5.5 as sequence 0. Then the four 5 bit HWs contained in one GBT frame form two 10 bit ADC values. In all other cases the content of two GBT frames is needed to reassemble the two ADC values. The sequences 0 and 2 look very similar in that sense, since here, also four HWs are contained within one GBT frame which belong to two complete ADC values (HW_{n-1} and HW_n). The issue in this case is the very first frame after a reset where the ADC value $n-1$ is the last part of the synchronisation pattern. In this case, the decoder could provide only one value instead of two. This would shift all following outputs by one channel. Since the output of the decoder module should always be the same, independent of the location of the synchronisation pattern in the GBT frame, to simplify the further processing, the content of two GBT frames need to be merged to form a reliable output sequence. This decoding of the channels, meaning that the detection of the synchronisation pattern and merging of the HWs to the correct ADC values is done for each of the five streams individually, as can be seen in figure 5.3. Each of the *channel decoder* will therefore provide two ADC values for each incoming GBT frame, together with an ID for the channel number. Additionally, more status information is provided about the detection of the synchronisation pattern,

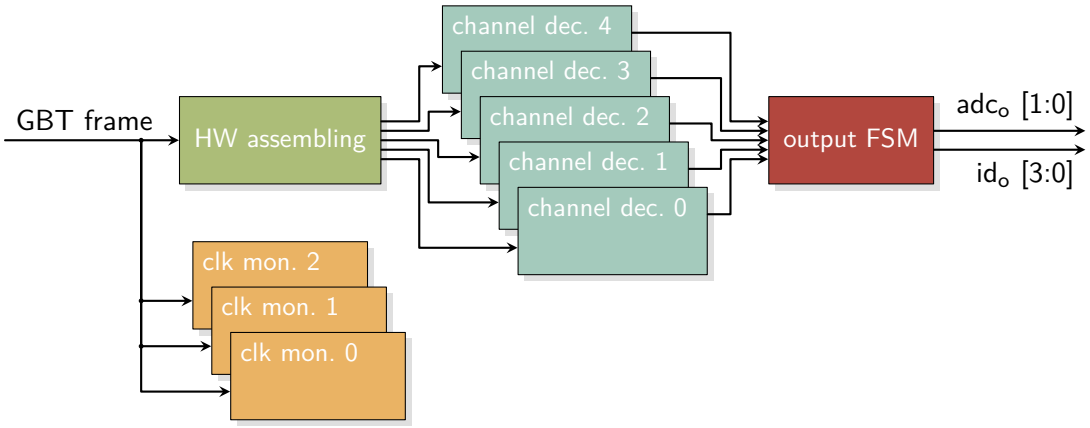


Figure 5.3: Block diagram of the GBT decoder. The *HW assembling* does the rearrangement of the individual bits of the GBT frame to form the individual HWs which are then analysed by the *channel decoders*. This module looks for the synchronisation pattern, combines two HWs to an ADC value and assigns a channel number. The FSM at the output takes care of a serialised data stream. In parallel, the ADC sampling clocks of the SAMPAs is monitored by the corresponding modules.

Sequence 0:	... 0000 1111 1111 1111 1111 0000 0000 0000 0000 1111 ...
Sequence 1:	... 0000 0111 1111 1111 1111 1000 0000 0000 0000 0111 ...
Sequence 2:	... 0000 0011 1111 1111 1111 1100 0000 0000 0000 0011 ...
Sequence 3:	... 0000 0001 1111 1111 1111 1110 0000 0000 0000 0001 ...

Figure 5.4: The four valid ADC clock sequences. Each sequence consist of eight groups with 4 bit each, which is periodically repeated. A valid one must contain 16 times a 1, followed by 16 times a 0. The phase can therefore be one of the four shown cases.

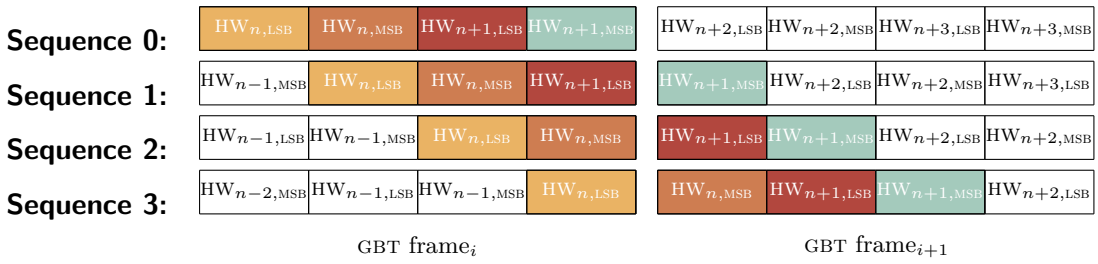


Figure 5.5: The four possible HW sequences. A GBT frame contains always four HWs, so only the four cases shown are possible. Everything else would just be a shift by one complete frame.

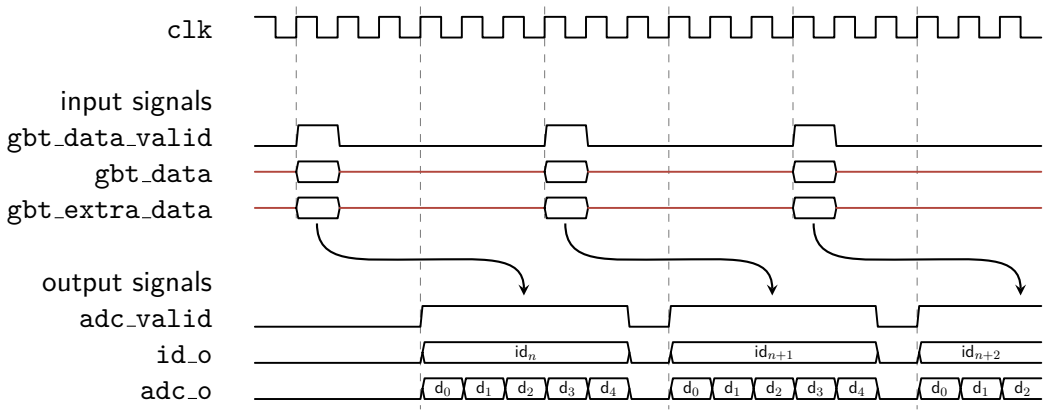


Figure 5.6: Data interface of the GBT decoder. The input is the payload of the GBT frame, the 80 bit of GBT data and the 32 bit extra data from the FEC field. The data output (there are also other ports, e.g. for monitoring and configuration) consists of a data port, delivering two ADC values at once and an ID port, flagging the data according to table 5.1.

in particular the start position of the pattern to determine which of the four sequences was present.

Each of the five HW streams contain only 16 of the 32 channels of a SAMPA. Since always two ADC values are decoded at once, a 3 bit ID ranging from 0 to 7 is sufficient to unambiguously mark those channels. Together with the information about the HW stream number and the decoder number, which corresponds to the link number and therefore to a specific card, the exact channel can perfectly be identified. The mapping between this 3 bit ID and all the channels of a FEC is shown in table 5.1. Each of the two subtables presents the mapping for one of the two output ports, table 5.1a for port 0 and 5.1b for port 1. The first column shows the ID. The other five columns show the respective SAMPA–channel combination for all the five HW streams, marked with d_0 to d_4. The tables show this combination for both, the odd and the even regions. The even regions receive the data from SAMPA 0 to 2, shown in orange, while the odd regions get the data from SAMPA 2 to 5, shown in blue. For the first four columns, only the SAMPA number changes between the two region types, the channel number stays the same. This was achieved by connecting SAMPA 3 and 4 to GBTx 1 in the same way as SAMPA 0 and 1 are connected to GBTx 0 (see the discussion about the FEC layout in subsection 3.3.2). The last column contains the mapping for SAMPA 2. The data of this chip is split between the two involved regions, the even regions receive the channel 0 to 15 while the odd regions get channel 16 to 31. By knowing the ID together with the number of the HW stream (and from which region the CRU receive its data) the SAMPA–channel combination can be obtained from this table. With the channel and the stream number plus the region, the pad number can be identified from which the data originates which is important for the sorting in the next subsection.

Some additional comments about the final interface of the GBT decoder, shown in figure 5.6, especially about the output signals. The input signals are defined by the

id_i \ d_i	0		1		2		3		4	
	SAMPA	ch	SAMPA	ch	SAMPA	ch	SAMPA	ch	SAMPA	ch
0x0	0/3	0	0/3	16	1/4	0	1/4	16	2	0/16
0x1	0/3	2	0/3	18	1/4	2	1/4	18	2	2/18
0x2	0/3	4	0/3	20	1/4	4	1/4	20	2	4/20
0x3	0/3	6	0/3	22	1/4	6	1/4	22	2	6/22
0x4	0/3	8	0/3	24	1/4	8	1/4	24	2	8/24
0x5	0/3	10	0/3	26	1/4	10	1/4	26	2	10/26
0x6	0/3	12	0/3	28	1/4	12	1/4	28	2	12/28
0x7	0/3	14	0/3	30	1/4	14	1/4	30	2	14/30

(a) Content of $adc_o[0]$ as a function of the ID and the data cycle. The CRUs of the even regions receive the orange SAMPA channels, while the CRUs of the odd regions receive the blue ones.

id_i \ d_i	0		1		2		3		4	
	SAMPA	ch	SAMPA	ch	SAMPA	ch	SAMPA	ch	SAMPA	ch
0x0	0/3	1	0/3	17	1/4	1	1/4	17	2	1/17
0x1	0/3	3	0/3	19	1/4	3	1/4	19	2	3/19
0x2	0/3	5	0/3	21	1/4	5	1/4	21	2	5/21
0x3	0/3	7	0/3	23	1/4	7	1/4	23	2	7/23
0x4	0/3	9	0/3	25	1/4	9	1/4	25	2	9/25
0x5	0/3	11	0/3	27	1/4	11	1/4	27	2	11/27
0x6	0/3	13	0/3	29	1/4	13	1/4	29	2	13/29
0x7	0/3	15	0/3	31	1/4	15	1/4	31	2	15/31

(b) Content of $adc_o[1]$ as a function of the ID and the data cycle. The CRUs of the even regions receive the orange SAMPA channels, while the CRUs of the odd regions receive the blue ones.

Table 5.1: Content of the GBT decoder data output. The output has two fields, providing each one ADC value per CC. The content of port 0 is shown in (a) and of port 1 in (b). Since each CRU receives the data of only one half of a FEC, the content is different for the even (orange) and odd (blue) regions. Please note that for d_0 to d_3, the channel number is the same for all regions, but the SAMPA changes from 0 to 3 and 1 to 4, while the data for d_4 comes always from SAMPA 2 but the channel numbers have an offset of 16 for the odd regions.

underlying GBT protocol. There is the 80 bit wide `gbt_data` bus and in addition the 32 bit wide `gbt_extra_data` bus containing the data from the FEC field. They are updated with a frequency of 40 MHz. So in a 240 MHz clock domain, only one out of six Clock Cycles (CCs) is utilised. If now the decoded ADC values would be simply written to the output ports after they are ready, this would be a waste of resources. The output bus would have a width of 10×10 bit plus additional IDs which are all utilised only one sixth of the time. To overcome this issue, a small FSM is placed before the output of the decoder (see last block in figure 5.3) to implement a more serialised output. A maximum of six CCs would be available to provide all the ten channels which are contained in one GBT frame on average because the input is a continuous stream and the output must be done when the next data arrives. A good mixture between having a uniform output and saving most resources is to provide only two ADC values at once but for five consecutive CCs. This is then the final output interface of the decoder. The overall decoding introduces a latency of three CCs after which the decoded values are provided for five CCs in a deterministic order, labeled by d_0 to d_4 in figure 5.6 and table 5.1. As a natural basis for this disaggregation serve the five HW streams, meaning d_0 contains the lower channel numbers from SAMPA 0 (or SAMPA 3 for the odd regions), d_1 the higher channel numbers of SAMPA 0, and so on. The arrows in figure 5.6 indicate that the content of one GBT frame is found in the following output sequence. This is only true for the special case when the synchronisation pattern is found at position zero for all the HW streams. Otherwise, the frame before will also be used to form the output.

5.2.2 A Two-Stage Sorting

After the GBT frames have been decoded, the input channels need to be sorted. The ordering in which the data arrives is based on the channel ordering of the SAMPAs. The clusters must be found in a later stage on neighbouring pads. This means, a mapping between the individual channels of the SAMPAs on each FEC to a specific pad location must be implemented and applied to reorder the arrival of the values to the subsequent modules.

The pad location is given by a row number within a region and a pad number within a row. Those two quantities vary strongly between the ten regions, the number of rows between 12 for the outermost region and 18 for region number four, and the number of pads between 66 for the first row of the innermost region and 138 for the last rows of region nine. More about the details of the pad plane layout can be found in appendix B. As an example for the mapping of a SAMPA channel to a pad location an excerpt of the Inner Readout Chamber (IROC) pad plane is shown in figure 5.7. The small red rectangles, in which the corresponding SAMPA channel is written, represent the single pads. The vertical blue lines, which are not necessarily straight, mark the border of a FEC. All pads within one vertical slice arrive through the same optical link and are decoded by the same GBT decoder. The regions are separated by the horizontal straight blue line which is just visible at the upper edge of the figure. The general layout is the same as in the figures in appendix B where the SAMPA ID is given instead of the SAMPA channel number. Within the sorting module the transition must be made, from the up to 20 input links, which correspond to vertical slices in the pad plane, to the 18 rows which are horizontal. In the excerpt can be seen already by eye that the assignment of a certain channel to a row or to a pad within a row

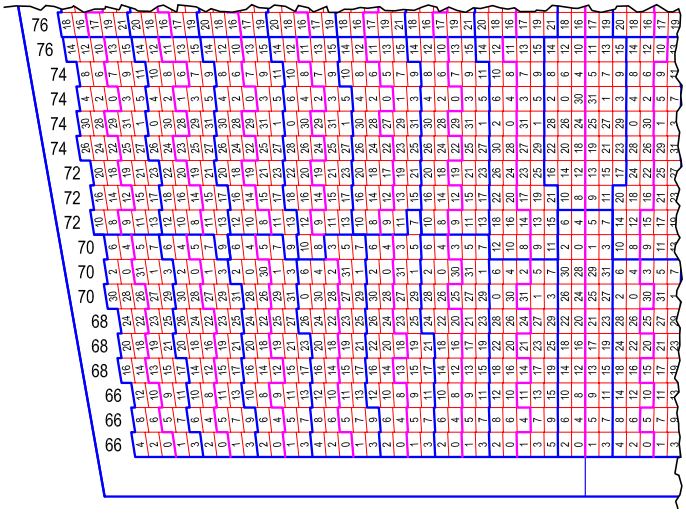


Figure 5.7: Excerpt of the IROC pad plane with the corresponding SAMPA channels written on the individual pads, taken from [34].

is different for the individual FECs, and even more diverse if one compares the mapping between different regions. Unfortunately, it does not follow a pattern, but was chosen instead to fulfil other criteria, e.g. to have similar trace lengths between the pad and the connector. Therefore, the mapping must be configured individually for each input channel of which there are up to 1 600 in one CRU. There are three requirements for the sorting module:

1. It must finish in time. The readout is a continuous one and the data can not be buffered. With a clock frequency of 240 MHz, only 48 CCs are available to sort all the 1 600 channels until a new readout cycle starts.

2. The resource consumption must not be unreasonably high, there is still the CF exercised later in the FW which is expected to be the largest consumer of resources.

3. The sorting must be configurable during runtime in a way, that the mapping for all the ten different regions can be achieved with the same FW.

Different approaches have been examined to determine whether they meet these conditions. The simplest method was to use the ID delivered by the decoder together with the source link to just look-up the target row and the target pad. This information is then used to store the ADC value in a 2-dimensional array (18 rows $\times$ 138 pads/row = 2 484 pads) from which the value can be taken for further processing. This approach is in principle very fast because only the look-up latency has to be taken into account. Also, the configurability is given because the mapping depends only on the content of the Lookup-Tables (LUTs) which can be filled with arbitrary values. However, the resource consumption is totally off, simply due to the high combinatorics. Each of the 20 $\times$ 2 output ports of all the decoders would be able to write into all of the 2 484 possible pads. This gives in total close to

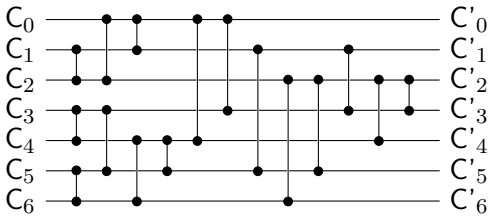


Figure 5.8: Illustration of a sorting network for seven elements. The vertical interconnections represent a comparison of the two involved numbers with a possible exchange afterwards, if necessary. Independent of the order of the input values C_x, the output C'$_0$ to C'$_6$ are always sorted.

10^6 paths which need to be realised and multiplexed for 10 bit ADC values. This number could be reduced by additional boundary conditions like limiting the possible pad range within a row for the individual FECs. Combining the mapping from all regions, it can be found that e.g. link 0 transmits only the pads 0 to 6 and link 1 the pads 4 to 13 and so on. Also, the possible target row can be restricted by knowing that e.g. SAMPA 0 and 3 are always connected to the lower row numbers. However, besides the effort to code all those limitations, the combinatorics stays very high and the module would still consume a majority of the resources which rules out this approach.

Another method considered was to use a *real* sorting algorithm to achieve the correct mapping. A set of sorting networks [54] can sort the channels according to the row and pad numbers which are looked-up for each channel. The working principle of a sorting network is indicated in figure 5.8. Each input number, in our case the pad number, is represented as a horizontal wire. They are compared with each other in a predefined ordering and swapped if those two are not ordered. In this way, an arbitrary sequence of pad numbers is sorted. However, a network for all the 138 possible pads of a row (assuming each row is sorted independently) would be quite huge, since the complexity increases with $\mathcal{O}(n \cdot \log^2 n)$ [54], and the final design would need 18 of those. To reduce the complexity of the individual networks, it is possible to split up the big sorting network into smaller ones which pre-sort the channels of only one link. Those pre-sorted segments then need to be merged in a second stage. With this, the number of inputs of one network is decreased dramatically to only seven, which is the highest number of neighbouring pads delivered by one link for all possible FEC locations in a sector. An optimal network, i.e. the smallest number of comparisons required, needs only 16 comparisons for seven inputs. The network shown in figure 5.8 is such an optimal network for seven inputs, generated with the Bose-Nelson algorithm [55]. Then, the resulting row segments still have to be merged in a second stage to complete the full pad row. However, the disadvantage of this approach is that all the channels need to be provided in parallel, instead of sequentially as it was implemented for the GBT decoder in order to save resources. In addition, not only the 10 bit word of the ADC value is needed, but also a 6 bit number to identify a row between 0 and 17 and a 8 bit number for a pad between 0 and 137. Even if the row number is omitted because a predefined pipeline is used for each row, the bits per channel is almost doubled, which leads in the end to big routing matrices and with that to a very high resource consumption.

The final solution keeps this two-stage approach with a pre-sorting on the FEC level and a subsequent merging of the row segments but replaces the sorting network with a RAM cell. The sorting can also be achieved by writing the ADC values to pre-defined addresses of the RAM and reading them again in a configurable order. This allows to use RAM cell of the FPGA, which already provides the routing matrices without additional resources. With that, the serialised output of the GBT decoder cab be used as an advantage. The Arria10 has two types of embedded memory, the 20 kbit Memory Blocks (M20Ks) which are dedicated blocks of 20 480 bit memory resources, and the Memory Logic Array Blocks (MLABs) which are 640 bit wide and are made up of ten Adaptive Logic Modules (ALMs), i.e. from logic resources [45]. The MLABs can be configured as simple dual-port RAMs, giving one write and one read port, whereas the M20Ks can be configured also as true dual-port RAMs, giving two independent write and read ports. Due to the continuous data stream to the sorting module of two ADC values at once for five out of six CCs, the true dual-port mode is needed to be able to deal with both values at once. Therefore the M20K was chosen for this purpose even though the amount of stored data of 80 channel $\times$ 10 bit/channel = 0.8 kbit is way smaller than the available space.

However, having one RAM in true dual-port mode is not sufficient in this case. If the two available ports are continuously used for the writing process, there is no time to read the data back. So in principle a quad-port RAM would be needed, two ports for the writing and two ports for the reading processes. To have in total four ports available, two RAM cells can be used. The data must be stored via the two ports always in the same RAM and can not be split across two independent RAMs because of the reading sequence. Having the data split would require that also during the read sequence, in each cycle one value is read from each of the involved RAMs. This strongly limits the freedom of possible output sequences whereas having all the data in the same RAM allows for an arbitrary reading sequence. Therefore, those two dual-port RAMs must be implemented in a ping-pong RAM configuration. While one RAM is being written to, the other RAM is being read. After a completed cycle they are swapped and read from the first and written to the second one. Such a ping-pong RAM is the core of the sorting module, shown in figure 5.9. Each of the two ports is dedicated to one of the GBT decoder output ports, RAM port A receives the data of $\mathrm{adc_o}[0]$ and port B of $\mathrm{adc_o}[1]$. The two respective write addresses are calculated in a deterministic way from the delivered ID and the data cycle in the following way:

$$\mathrm{add}_A = (\mathrm{d}_i \cdot 16) + (\mathrm{id}_i \cdot 2) \tag{5.1}$$

$$\mathrm{add}_B = (\mathrm{d}_i \cdot 16) + (\mathrm{id}_i \cdot 2) + 1, \tag{5.2}$$

where d_i ranges form 0 to 4 and id_i from 0 to 7, as can be seen in table 5.1. In this way it is ensured that the data from the same channel of the same SAMPA is always stored at the same address, for all the FEC positions in all regions. To apply the sorting, those addresses have to be asserted to the two read address ports in the correct order. With this, pad after pad can be read from the RAM, where the data from port A is always used first by combining the two values. This sequence of two addresses is always the same for a specific FEC location and can therefore be stored in a LUT during the configuration of the CRU at runtime.

So far, the output of the ping-pong RAM would be just a sequence of channels, as the input but in a different order. To complete the sorting in two dimensions, row-breaks are

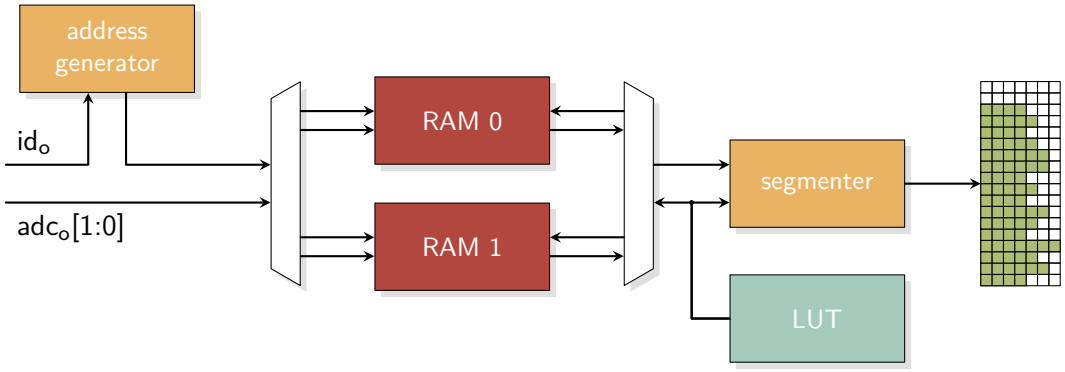


Figure 5.9: A configurable pre-sorting module based on a ping-pong RAM. The write address is generated from the ID marking the SAMPA channel while the read addresses are stored in the correct ordering in a LUT. The values of the read sequence are then divided into short row segments, indicated as 80 green filled pads in the reduced pad plane.

needed at the correct location. For this, a two bit command is appended to the set of two read addresses which is used by the *segmenter*, schematically shown in figure 5.9, to split up the sequence of channels into short row segments. The length of the segments varies between four and seven pads, depending on the row and FEC location. The two bits are needed because three cases are possible: the row must be changed after the first of the two simultaneously read pads, after the second pad, or not at all in this read cycle. To have the behaviour of the *segmenter* completely controlled solely by this command, the fourth available bit combination is used to mark the very last read cycle after which this module is reset and starts again with the first pad in the first row. In this way, the sorting is completely flexible and depends only on the content of the LUT. Since a maximum of 7 adjacent pads and 18 rows are read by one FEC, the port width of the module can be limited to 18 times 7 pads. The second output bus of the pre-sorter is a bit-mask with a width of 18, marking first the validity of the segment output bus, and second identify to which row the current segment belongs to. Also here, the segment output is sequential, one segment after the other via the same port in order to spare resources on unnecessary parallel outputs which are rarely used.

As a next step, those variable-length segments that belong to the same row but come from different links need to be merged to reassemble the complete pad rows. It turned out that it is impractical to have the full pad rows with a length of in total 138 pads each, at least there is no advantage in having them. On the contrary, keeping the pad row segmented allows for a more efficient implementation of the baseline correction module and cluster finder. However, the segmentation needs to be changed from a variable to a fixed length. It was seen that the optimal size is six pads, for two reasons. First, the baseline correction is mainly done with Digital Signal Processors (DSPs) which can be configured to process two streams in parallel (more in subsection 5.2.4). By having three DSPs utilised, all six pads of the segment can be processed in parallel without using a DSP only half. The second reason is that each cluster finder instance has six unique and four shared pads (more details are discussed in section 5.3). Having the segmentation set to six has the advantage that each instance needs the data of only two consecutive

segments. So the task of the row merger module is now to realise the transition from the processing of each input link individually, to a joint processing of all links combined, by merging the segments belonging to the same row from all links. One way to realise this is to implement all sequential mappings and make them selectable via two parameters, the region number which is configurable during runtime, and the row number within the region which is a *generic* parameter of the module. Having the latter one as a generic allows to instantiate the same module — containing all the mappings — once for each of the 18 rows per region and letting the compiler remove the respective irrelevant parts. This means that this module is not completely configurable at runtime, but the correct mapping for the relevant region can be selected during runtime, fulfilling all requirements. The complete mapping for all regions consists of more than 4 000 assignment statements. Since it is almost impossible to manually write this error-free, it was generated automatically using the ALICE O^2 framework (section 6.3). The original source files which were also used to build the real pad plane are used here to provide the correct mapping for simulation and reconstruction purposes. During the assignment, there are two empty pads (filled with an ADC value of 0) added to the *left* side of the row, below pad zero, and four empty pads added to the *right* side of the row. The two pads on each side are necessary to be able to find peaks also at the borders (more in section 5.3) and the additional two on the right side to complete the segmentation of six ($2 + 138 + 4 = 144$, which can evenly be divided by six). Since only the outermost rows of region nine have 138 pads in the real system, there are some unused pads with their signal set to zero by default. The output ordering is a bit unintuitive, it starts with the segment containing the highest pad numbers and goes down to the segment with the lowest pad numbers. That is still a remnant from the time when the CF was not yet completely worked out and was introduced to unravel the processing. Since this order is not really important for further processing — it only has to be implemented correctly in the following modules — it was never changed.

5.2.3 Common Mode Calculation

The CM effect is the main reason why the readout scheme had to be changed with respect to the original Technical Design Report (TDR). The correction algorithm to be implemented has already been introduced in subsection 3.1.1, in summary: the mean ADC value of all pads, with the signals excluded, must be subtracted from every pad for each time bin. This algorithm can well be separated into two steps, first the calculation of the mean value, which is done in this module, and second the subtraction of this value from the individual pads, which is done together with the other corrections in the BLC module. How the CM calculation is realised is shown in figure 5.10. The input to this module are the outputs of the 20 GBT decoders and the output is one value for every 5 MHz readout cycle. As a first step, the peak exclusion takes place. For this, two consecutive time bins must be buffered to be able to exclude the whole peak for the best performance. It must be noted that the peak detection and exclusion is only necessary in time direction since it is done for all pads anyhow. So there must be no sorting beforehand to group together neighbouring pads. That is why this calculation can take place in parallel to the sorting algorithm which was described in the previous subsection. To find a peak, the difference between two consecutive time bins is calculated and if this exceeds a configurable threshold, a rising

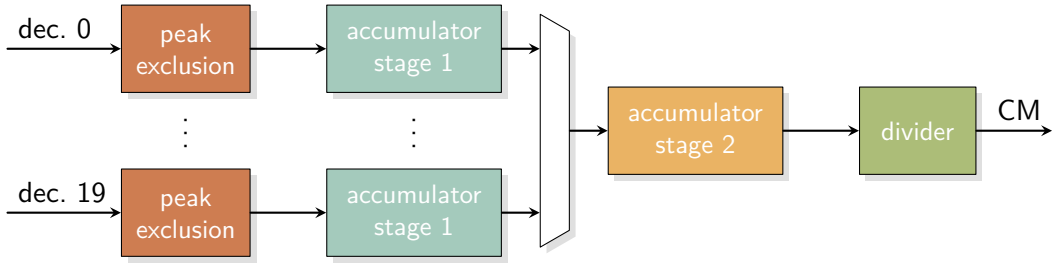


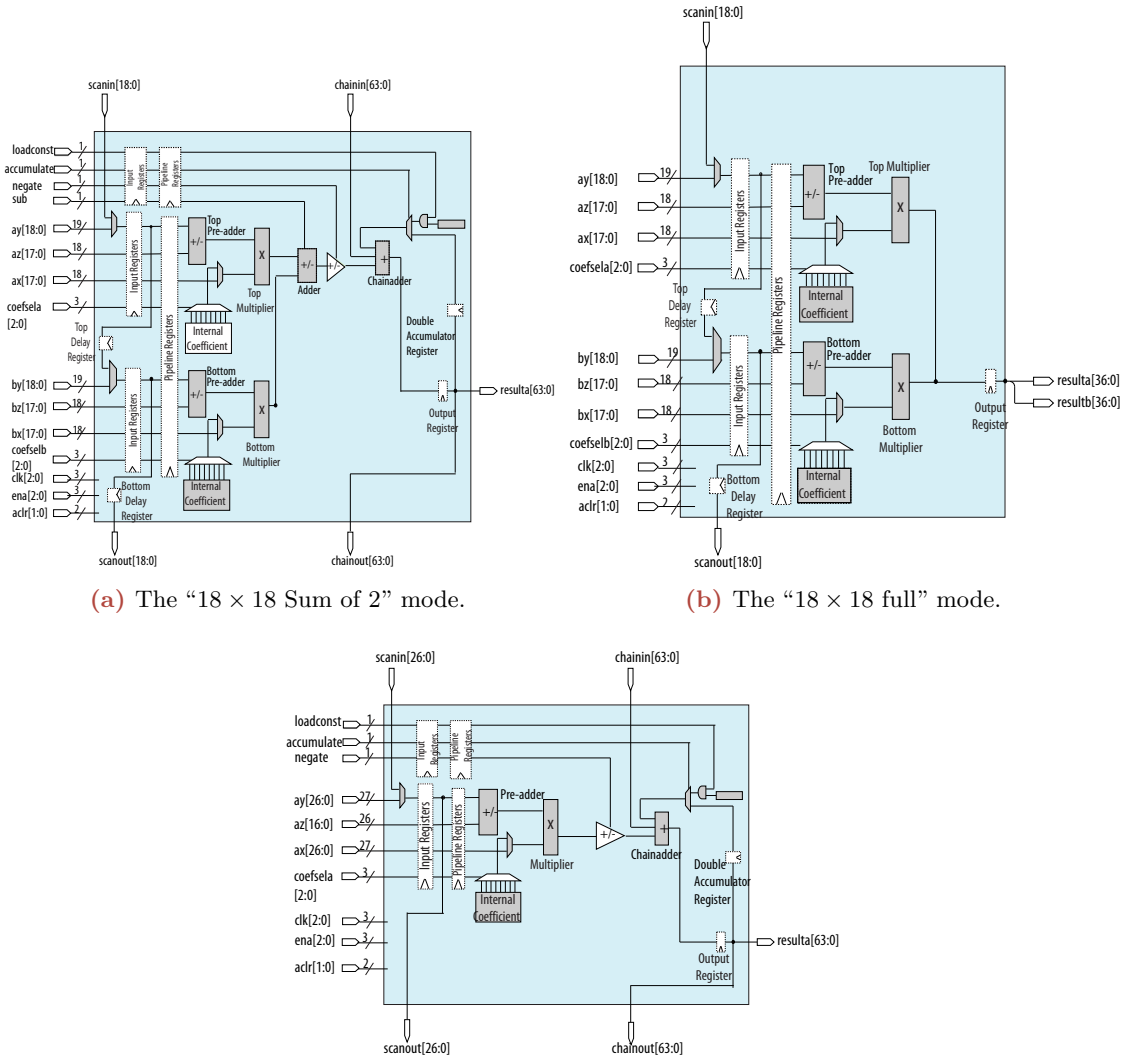
Figure 5.10: Block diagram of the CM calculation module. A peak exclusion module and a first accumulator stage is implemented for each if the input links. A second accumulator stage then combines the results and a divider normalises the value to the number of contributing pads.

edge in the signal is detected. If this rising edge is followed by a falling edge, again via the difference of two consecutive time bins, the peak is found and can be excluded. Since three time bins are needed to be able to decided that a peak is present, two time bins need to be buffered.

After the exclusion, the first accumulator stage takes place. Here, the charge of all surviving pads of one time bin is summed up and the number of contributors (number of surviving pads) is counted. Before summing them up, the pad-wise pedestal value needs to be subtracted. This is not important for the peak detection since this additive contribution cancels out at the subtraction of two time bins of the same pad, but would bias the accumulated charge in an indeterministic way because of the excluded pads. If no pad would be excluded, just the sum of all pedestal values could be subtracted from the final accumulated charge. A corresponding accumulator could also be implemented by hand. However, in order to save resources one should use for this kind of tasks the provided DSP blocks. The utilised Arria10 FPGA contains in total 1 518 DSPs, of which some can be used at this place. The Arria10 native FP DSP IP core, which is able to carry out arithmetic operations like adding and multiplying of FP numbers, will be used. A detailed description of the IP core and all its functionalities is given in [56]. Important for the application is that it can be configured for different operating modes. The so called "18×18 Sum of 2" mode, of which the block diagram is shown in figure 5.11a, fits perfectly to the requirements of this module. There are two parallel input groups — the GBT decoder output is two ADC values at once — each having first a *Pre-adder* available which can be used for the pedestal subtraction. In this case, the *Multiplier* which comes next for each input can be set to one and therefore be ignored. Next, the *Adder* calculates the sum of both pedestal subtracted input values and the *Chainadder* in the end can be used to accumulate the charge of a complete readout cycle. This single DSP core takes care of all the arithmetic operations needed for the first accumulator stage, without any additional resources, by applying the equation

$$\text{resulta} = \sum ((\text{ay} - \text{az}) + (\text{by} - \text{bz})) \tag{5.3}$$

to the input signals. The names of the variables were taken from the block diagram so that they can be mapped directly to the individual ports. Also, the bit width of the DSP



(a) The "18 × 18 Sum of 2" mode.

(b) The "18 × 18 full" mode.

(c) The "27 × 27" mode.

Figure 5.11: Block diagram of the Arria10 native FP DSP IP core in three different configuration modes, all three taken from [56].

is sufficient because at most 80 channels are summed up for the individual links, each a 10 bit ADC value. This means that a 17 bit number can hold the maximum possible value of $80 \cdot 1\,023 = 81\,840$. Even after adding four additional bits for the FP decimals, the result is significantly below the maximum output width of the DSP of 64 bit.

In the second stage, the results from the individual links need to be further accumulated. A different DSP configuration was chosen, the "27×27" mode shown in figure 5.11c, to compensate for the increased bit width. This configuration provides only one input which can therefore be up to 27 bit wide, sufficient to accumulate the 21 bit numbers of the previous stage. In this case, the *Pre-adder* can be used to add up two values which are then further accumulated by the *Chainadder*. A small FSM is written to first assert the results of always two links to the inputs *ay* and *az*, and afterwards reuse this DSP to also sum up the number of contributors of each of the links. There is enough time to accomplish these two task sequentially because each accumulation needs only 10 CCs, 20 links divided by two simultaneous operations. Also here, 48 CCs are available to process the complete readout cycle.

The last step in the calculation of the CM value is the division of the accumulated charge by the number of pads which contributed to the value. A soft IP core provided by Intel is used for this. Those IP cores delivered by the vendor offer usually a more efficient logic synthesis compared to a self-written VHDL module. All modules provided to dedicated tasks, of which one is a divider, are described in [57]. They can be used by instantiating the corresponding module, being configured through several generics to adapt for the design requirements. Since there is plenty of time at this stage (again up to 48 CCs), the divider is configured to use many pipeline stages for the devision of a 26 bit number by a 11 bit number to reduce the resource consumption.

The implementation of the module is a shared task. The development of the general concept is part of this thesis while the actual implementation in VHDL, which is still work in progress, is carried out by colleagues from the Nagasaki Institute of Applied Science in Japan. After the implementation is carried out, the total latency needs to be determined. Should it be larger than the latency of the sorting and merging modules the data needs to be delayed so that the CM value of the correct time bin is used in the BLC module. This can be done in the ping-pong RAM of the pre-sorting. Currently only 80 out of the 2 048 available addresses are used for the sorting. By having an offset between the write and read addresses, up to $\lfloor 2048/80 \rfloor = 25^*$ time bins could be buffered here without additional resources.

5.2.4 Baseline Correction

The last module of the data preparation part is the BLC module. Three contributions need to be corrected for, first the gain non-uniformity, second the much discussed moving baseline, the CM effect, and third the pedestal value of the ADC of the electronics. While the latter two corrections are additive, the first one is multiplicative. Since these distortions are applied to the real signals in this order, they have to be corrected in reversed order, first the pedestal value, then the CM and last the gain. This can also be summarised in the

*Those brackets symbolise the *floor*-function, $\text{floor}(3.3) = \lfloor 3.3 \rfloor = 3$.

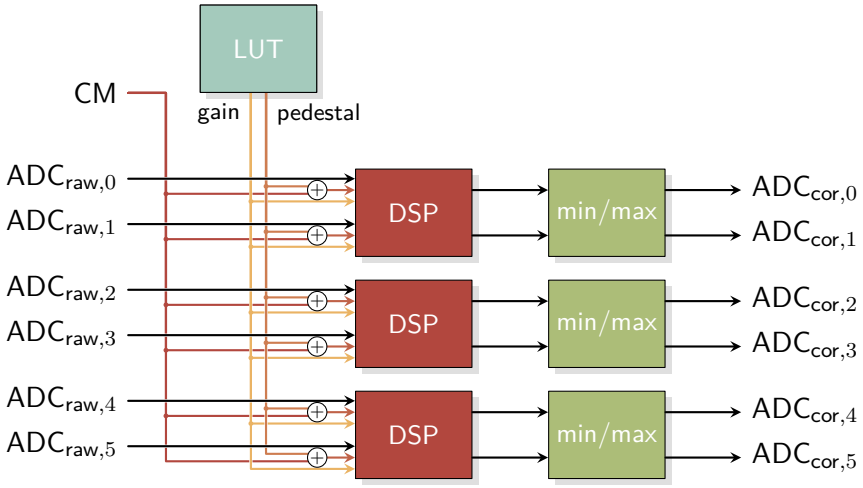


Figure 5.12: Block diagram of the BLC module. The six parallel input values are processed by three DSP blocks. The gain correction factors and pedestal values are stored in a LUT. Before using the pedestal value, the CM contribution is added. After the correction is done, a module ensures that 0x0000 $\leq$ ADC$_{cor}$ $\leq$ 0x3FFF is always true.

equation

$$\mathrm{ADC}_{\mathrm{cor}} = (\mathrm{ADC}_{\mathrm{raw}} - c_{\mathrm{ped}} + c_{\mathrm{cm}}) \cdot c_{\mathrm{gain}}, \tag{5.4}$$

where ADC$_{cor}$ is the corrected ADC value, ADC$_{raw}$ the raw value from the Front-End Electronics (FEE), c_{ped} the pedestal value, c_{cm} the CM value and c_{gain} the gain correction factor. It would be beneficial in terms of resource consumption if also this arithmetic operations, which needs to be applied to each individual pad, could be carried out by a DSP. Indeed, there is another configuration of the DSPs available which implements almost exactly this equation and then also on two independent inputs. The configuration is called the "18 $\times$ 18 full" mode of which a block diagram is shown in figure 5.11b. Only the two additive components, c_{ped} and c_{cm}, have to be combined beforehand. Besides that, there is again the *Pre-adder* which is used for the additive correction and this time the *Multiplier* is used for the gain correction. Also, in terms of port widths fulfils the DSP the needs. The input is the 10 bit ADC value, converted beforehand to a 10.4 FP number, which is also the expected output precision.

To be able to process the six pads of the row segment from the sorting and merging module simultaneously, three DSP cores are needed. This is also shown in the block diagram of the BLC module in figure 5.12. Each of the three DSPs processes two ADC values in parallel. The needed gain correction factors and pedestal values are stored in a LUT during configuration from which they are read, depending on the input segment number. Although the number of words in the LUT is 24 and thus quite low (one per segment), the width of each is rather high because in total six sets of gain factors and pedestal values need to be read at once. The precision of those factors are chosen to be the same as the resulting corrected ADC value, therefore they are 10.4 FP numbers as well. The module was designed in a way that each correction can individually be enabled or disabled. With the corresponding configuration it can be decided during runtime which correction should be turned on or off.

A further process is connected downstream of the calculation, checking the results. Since the CF is designed to work with positive 10.4 FP numbers, it must be ensured that the output values of the BLC module are exactly this. By subtracting too big pedestal values, it could happen that the result of the calculation turns negative. This is a rather easy task to filter out because negative numbers are naturally represented as two's complement[†] in an FP (or integer) arithmetic. So only the MSB of the result needs to be checked. If this bit is one, the complete value is set to zero. On the other hand, by adding a big CM value or multiplying with a gain correction factor above 1, a possible overflow can occur, meaning the result is bigger than the maximum possible value in 10.4 FP representation of $1\,023 + \frac{15}{16} = 0\text{x}3\text{FFF}$. For this, all bits between the MSB of the result and the MSB of the interesting 14 bit of the FP number need to be checked. If any of the bits is one, an overflow has occurred and the result needs to be set to the highest possible value. The BLC module is correcting only the baseline, meaning that the ADC value is corrected and transformed from a 10 bit integer into a 10.4 FP number. The structure of the interface, the segmentation into 24 times 6 pads and the valid signals, is kept the same as from the merger module.

5.3 Cluster Reconstruction

After the data is prepared, finally the reconstruction of the charge clusters can be done. This process is subdivided into three parts, the CF which implements a true 2-dimensional peak finding, the Cluster Processor (CP) to calculate the cluster properties and to generate the Cluster Data Format (CDF). As a last step, there is a merging network which combines the output of the individual CPs into four final FIFOs. From here, the cluster data are then written to the DMA FIFOs to transfer them to the host machine for further processing and forwarding to the Event Processing Nodes (EPNs) for the online tracking and storage.

As already mentioned, the clusters will be searched for in each pad row individually. This results in the ideal case in one cluster per row for a track traversing the whole TPC, giving the corresponding number of space points for the later tracking algorithm. But before starting to implement the CF algorithm, one has to clarify what actually needs to be found and processed. Especially the size of the clusters in pad and time direction need to be investigated. By fixing those parameters at an early stage, the whole development and implementation process can be simplified.

5.3.1 Determining the Cluster Size

It was found and stated already in the original TPC TDR that for a GEM based readout system, the broadening of the electron cloud is dominantly given by the diffusion during the drift time [29]. The spread by the GEM intrinsic properties is therefore negligible and only the diffusion contribution needs to be taken into account in the calculation of the expected cluster size. The spread of the electron cloud in longitudinal and transversal direction is then the product of the diffusion coefficient $D_{T/L}$ and the drift length l_{drift}:

$$\delta_{L/T}(l_{\text{drift}}) = D_{L/T} \cdot \sqrt{l_{\text{drift}}}, \tag{5.5}$$

[†]To calculate the two's complement of a number, one has to invert the individual bits and then add 1.

gas mixture	drift velocity v_d (cm/μs)	diffusion coefficent D_L ($\sqrt{\text{cm}}$)	D_T ($\sqrt{\text{cm}}$)
Ne-CO_2-N_2 (90-10-5)‡	2.58	0.0221	0.0209
Ne-CO_2 (90-10)	2.73	0.0231	0.0208
Ar-CO_2 (90-10)	3.31	0.0262	0.0221

Table 5.2: Diffusion coefficients and drift velocities for electrons in different gas mixtures, evaluated at an electric field of 400 V/cm, based on [5].

taken from [29]. These diffusion coefficients for electrons in both directions were determined for different gas mixtures and are summarised in table 5.2, together with the drift velocities. The coefficients for the transverse and longitude directions are very similar for each of the shown gas mixtures. They are all CO_2-based mixtures of which the one with added Nitrogen, the Ne-CO_2-N_2 (90-10-5)‡, is the baseline gas mixture for the TPC in Run 3. The width of a cluster in pad direction can then be estimated for this gas composition by calculating

$$\sigma_T = \delta_T = 0.0209\,\sqrt{\text{cm}} \cdot \sqrt{250\,\text{cm}} \approx 0.3305\,\text{cm} \tag{5.6}$$

for the maximum drift length of the TPC of 250 cm if the cluster was generated near the central electrode. Taking a 3σ interval into account, a width of the cluster of 1 cm must be covered to include most of the charge. With the smallest size of the individual pads of 4.16 mm in the IROC and the largest size of 6.08 mm in the OROC 2 [34], the maximum cluster width is then spread across ± 2.4 pads around the peak in the IROC and ± 1.6 pads in the OROC. So the size of a cluster can safely be fixed to 5 pads, containing in the worst case for the IROC 99.84 % of the charge. For the OROCs, a cluster width of 4 pads would also be fine but this introduces two complications. First, one would need to implement two different sizes for the cluster which makes the whole implementation more complicated. Secondly, if the number is odd, the centre is uniquely determined and the cluster extends two pads in both direction. With an even number of pads, one would need to compare the pads next to the centre and add the fourth to the corresponding side with the larger neighbour, which is in this case an unnecessary complication. Therefore, the size in transversal direction is fixed for all ROCs to 5 pads.

For the extension in the time direction, two more contributions need to be taken into account in addition to the diffusion: the track inclination angle and the shaping time of the electronics. Tracks with a higher inclination angle will deposit charge more along the drift path of the electrons. This extends the cluster size in time direction. The mean width of the clusters in longitudinal direction can be calculated with the equation:

$$\sigma_L^2\left(r, l_{\text{drift}}\right) = \frac{1}{v_d^2}\left(\delta_L^2\left(l_{\text{drift}}\right) + \frac{\tan^2\lambda(r, l_{\text{drift}}) \cdot L_{\text{pad}}^2\left(r\right)}{12}\right) + \sigma_{\text{ele}}^2 \tag{5.7}$$

$$= \sigma_{\text{det}}^2\left(r, l_{\text{drift}}\right) + \sigma_{\text{ele}}^2, \tag{5.8}$$

‡ Must be normalised to 100 %. This statement refers to a mixing ratio between Ne and CO_2 of 9:1 to which 5 % of N_2 is added. The notation is given in this way for easier comparison with other gas mixtures which have no or a different N_2 admixture.

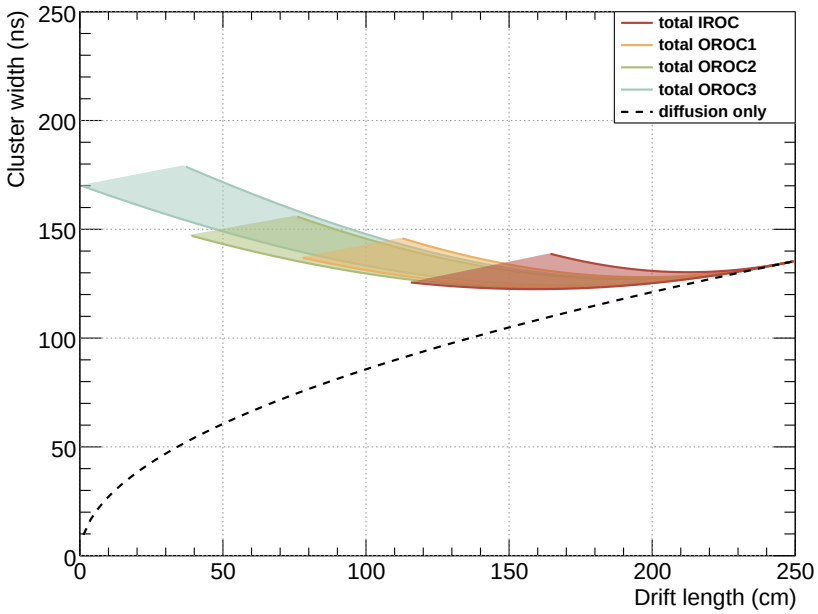


Figure 5.13: Extension of the clusters in time direction as a function of the drift length, calculated using equation 5.7 without the contribution of σ_{ele}. The equation was evaluated at all the ROC boundaries and the pad lengths given in table 5.3 for the baseline gas mixture. The acceptance shown is limited to the relevant region of $|\eta| < 0.9$, based on [5].

taken from [5], where $\delta_L(l_{\text{drift}})$ is the spread due to the diffusion during the drift, $\lambda(r, l_{\text{drift}})$ the inclination angle of the track with respect to the pad plane, $L_{\text{pad}}(r)$ the length of the pads in the different ROCs and σ_{ele} the approximate sigma of the semi-gaussian signal output of the electronics. The dependency of the inclination angle on the drift length and the radius is a pure geometrical effect. Clusters with a long drift length were generated close to the central electrode and therefore originate from tracks with a smaller inclination angle. Tracks which point towards the IROC — to a smaller radius — have a larger pseudorapidity and therefore also a greater inclination angle compared to those which point towards the OROCs. The contribution of the electronics depends on the shaping time of $t_{\text{FWHM}} = 190\,\text{ns}$ and can be approximated by dividing by a constant factor of 2.4: $\sigma_{\text{ele}} \approx t_{\text{FWHM}}/2.4 \approx 80\,\text{ns}$ [5]. The detector component σ_{det} of the equation is plotted in figure 5.13 for the baseline gas mixture as a function of the drift length. The dashed line is the contribution of the diffusion term, which increases with increasing drift length. The sum of this diffusion term and the angular contribution is plotted for all the ROCs individually in different colours. With a decreasing radius (going from OROC 3 to IROC), the inclination angle increases and with that also its contribution to the total cluster size. The inclination angle is particularly large for clusters that are created close to the readout planes and therefore have a short drift distance. Only the relevant pseudorapidity range of $|\eta| < 0.9$ was taken into account for the plot, which is why the drift distances for e.g. the IROC (shown in red) start from 115 cm to 164 cm. For the OROCs the same η range corresponds to a larger drift length. The different radii and pad lengths which were used in the calculation are given in table 5.3. Requiring that a 3σ range of the signal is contained within 5 timebins

ROC	region	radius inner (cm)	radius outer (cm)	pad length (mm)	pad width (mm)
IROC	0	84.85		7.5	4.16
	1			7.5	4.20
	2			7.5	4.20
	3		132.1	7.5	4.36
OROC 1	4	134.7		10.0	6.00
	5		168.7	10.0	6.00
OROC 2	6	170.8		12.0	6.08
	7		206.8	12.0	5.88
OROC 3	8	208.9		15.0	6.04
	9		246.4	15.0	6.07

Table 5.3: Geometric parameters of the pad plane of a TPC sector. The given radii correspond to the bottom of the lowermost and the top of the uppermost pad row in the individual ROCs [34].

of the 5 MHz sampling clock, the condition $3\sigma_L \leq 2.5 \cdot 200\,\text{ns}$ must be fulfilled, or

$$\sigma_{\text{det}} \leq \sqrt{\left(\frac{2.5}{3} \cdot 200\,\text{ns}\right)^2 - (80\,\text{ns})^2} \approx 146.2\,\text{ns}, \tag{5.9}$$

which is the case for IROC and OROC 1 as can be seen in the figure. Even for the maximum cluster length of 180 ns in OROC 3, the 5 timebins would still contain

$$\frac{2.5 \cdot 200\,\text{ns}}{\sqrt{(180\,\text{ns})^2 + (80\,\text{ns})^2}} \approx 2.54\sigma_L \tag{5.10}$$

of the signal, or 98.89 % of the charge. It is therefore also valid in time direction to limit the cluster to 5 time bins. To summarise, the surrounding 5×5 matrix around a peak in pad and time direction contains even in the worst case more than 98.73 % of the charge. Since the intrinsic energy resolution of the GEM system is ∼12 % [5], the cluster size can be fixed to these widths because the loss of charge of 1.27 % due to this cut is negligible.

Increasing the acceptance to $|\eta| < 1.4$, which is also used sometimes for the TPC, leads to a maximum width of the cluster of 210 ns in time direction. By limiting the cluster to 5 pads in this case, already 2.8 % of the charge can not be taken into account. Although, this is still small compared to the 12 % energy resolution of the GEM system.

5.3.2 The Concept of the Cluster Finder

The basic approach of the cluster reconstruction is to find a 2-dimensional peak in pad and time direction in each pad row individually. With this approach, a 3-dimensional cluster finding is reduced into a 2-dimensional problem which simplifies the processing. Then the charge of the surrounding bins of a peak is used to compute the cluster properties

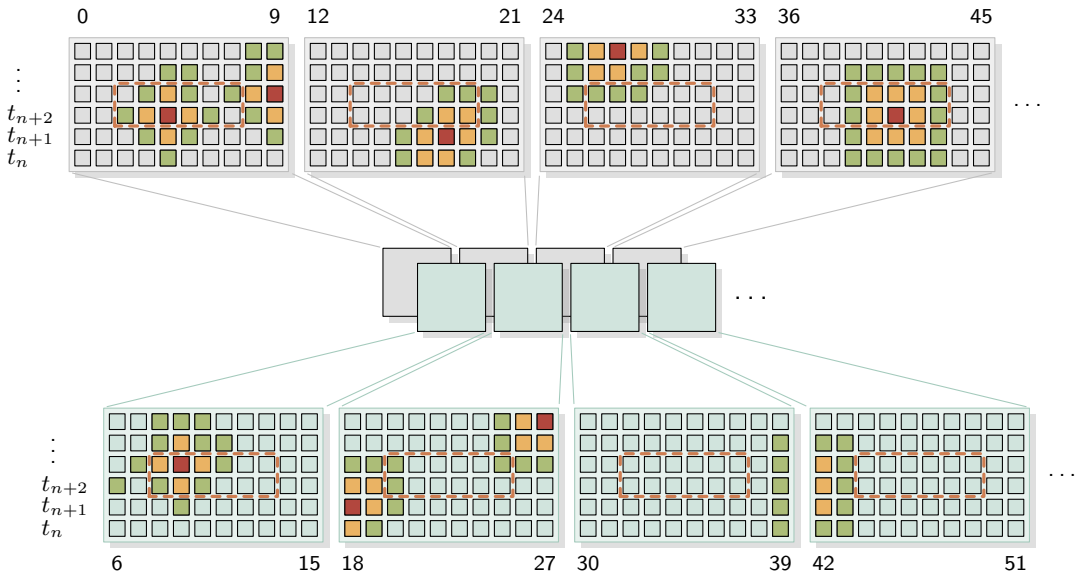


Figure 5.14: General concept of the cluster finding approach. Each of the CF instances gets only a short segment of the pad row (here ten neighbouring pads) in which they look for peaks. There are some double pads shown in the top and the bottom row (e.g. pads 6 to 9 for the first CFs in both rows) to compensate for the additional borders between the individual instances. The orange dashed rectangle marks the allowed positions for a peak to ensure that a cluster is found only once.

like the total charge, the positions in pad and time direction as well as the widths in those directions, assuming a gaussian distributed charge both in pad and time direction. It was seen before that one key element in the ability to handle these data rates is to keep the individual data streams as much separated as possible. The same is true for the clusterisation. So the general idea is to have many small, independent CF instances, each looking only at a small segment of the pad row for peaks. This basic concept is shown in figure 5.14. The pad row is subdivided into small pieces. Each instance receives time bin after time bin of only those pads. They are buffered locally and the individual instances look independently for peaks in their own region. Since 5 timebins need to be taken into account for the calculation of the cluster properties, at least this amount of data needs to be buffered. The Arria10 offers plenty of small memory blocks, the MLABs, which are excellently suited for this purpose. Each MLAB is formed from ten ALMs, which are configured as ten 32×2 bit memory blocks, giving one 32×20 bit simple dual-port SRAM block per MLAB. Assuming that always ten pads are processed by one CF instance, then a local storage of at least $5 \times 10 \times 14\,\text{bit} = 700\,\text{bit}$ is needed, which fits into two MLABs.

The subdivision of the pad row into small pieces introduces additional artificial borders between the individual CFs, which need to be compensated for. One way to deal with that is to have overlapping pads — pads which are present in more than one CF. For example, pads 6 to 9 are present in the top left as well as in the bottom left of the CF instances in figure 5.14. Though, it must be ensured that a peak is found only once, even if the charge is present in two CFs. This is achieved by restricting the allowed position of the

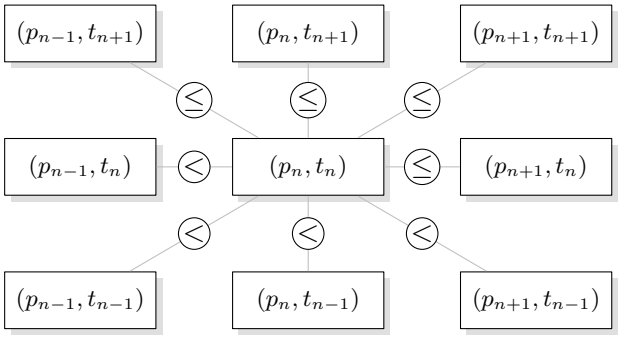


Figure 5.15: The definition of a peak. The value of the centre bin must be larger than the four bins on the bottom left part, but only larger or equal to the four on the top right part.

peak within the instances. This region for the peaks is indicated by an orange dashed rectangle in the figure. Those regions do not overlap so that the peaks are not found twice. The region ends in the first CF (top left) with pad 7 and starts in the second CF (bottom left) with pad 8 and so on. The number of double pads per CF is given by the cluster size. As it was described in the previous subsection, the width of the clusters is fixed to five pads. With this, it is clear that always two additional pads have to be added on both sides of the short segment to have the complete charge of a cluster contained in one CF. So the total width in pad direction clearly needs to be optimised in order to not waste too many resources for double pads while still having the time to find all clusters. Also the number of buffered time bins can be used for this optimisation problem. Since the processing time is an important restriction, the actual peak finding algorithm needs to be fixed before starting to optimise the size of the CF instances.

5.3.3 The Peak Finding

To find a peak, one first has to define what needs to be found. The most basic definition of a peak would be the ADC value is greater than the value of all the eight neighbouring bins. This definition has an obvious problem: if the value of two adjacent bins is the same, and their value is also higher than that of the other surrounding ones, then the peak is not found because neither of the two maximum values is higher than *all* the others. This can be avoided by introducing an asymmetry. In one dimension, a peak would then be defined as bigger than the left pad and bigger or equal than the pad on the right side. If this logic is applied to two dimensions, the pattern shown in figure 5.15 results. So a peak is found at the central pad if the value is bigger than the one of the pad on the left side and bigger than the values of all three pads of interest in the last time bin, and bigger or equal than the value of the pad on the right side and bigger or equal than the values of all three pads of interest in the following time bin. How exactly the pattern is arranged does not really matter, but it must be symmetric to achieve a constant bias in one direction which could be corrected later, if needed. Actually, it would also be possible to mirror the pattern for every second row so that the bias is automatically removed by combining the information of multiple rows. At a later stage, these comparison operations could also be adapted in a

way that not the ADC values are compared but rather the differences must exceed some threshold to suppress noise clusters.

To actually find such peaks, several algorithms were under consideration. The basic principle though was always the same, the data previously stored in the local RAM is read back value by value and searched for peaks. But the ordering is different for the different algorithms. One option was an adaption of the widely known *divide-and-conquer* approach [58]. Actually, the whole concept as it is shown in figure 5.14 is already a kind of divide-and-conquer approach, just at a higher level. The basic idea of the algorithm is to divide the problem into smaller instances of the same problem. Then they are conquered by recursively solving the subproblems by eventually further subdivisions until the problem is a very basic one and can easily be solved. The results of the subproblems need to be combined for the solution of the original problem . So this approach seems to fit very well to the problem of finding all clusters in an arbitrarily sized 2-dimensional plane. The plane is simply divided further into smaller areas until one reaches the smallest needed size of 3×3 to determine whether a peak is present or not. It turned out that applying this type of algorithm does not give any advantage in terms of processing time compared to simply going through the storage and check pad after pad, at least not if the latter one is done in an optimised way by buffering some pads temporarily. Since all clusters that could be in memory have to be found, there is no other way than to really look at all the pads, independent of the algorithm.

It turned out that the simplest solution is the best. The data is anyhow written pad after pad into the storage because the MLAB is only a simple dual-port RAM. So by adding a shift-register of appropriate size in front of the RAM, peaks can be found already before storing the data into the RAM. Since the pads arrive always in the same order, always the same elements of the shift-register needs to be compared with each other to achieve the pattern shown in figure 5.15. Furthermore, there are two more reasons speaking in favour of doing the peak finding in this way, already before storing the values. First, storing a *peak-flag* with which the peak can be identified later after reading it back, together with the data does not increase the resource usage. The MLABs are 32×20 bit RAMs in which 14 bit values have to be stored. So there are anyhow 6 bit available for each value to store additional information along with the ADC value of which one bit can be used to identify a peak. Second, by doing the needed comparisons sequentially, the number of needed comparators can be reduced from eight to four while still comparing each pad with all its eight neighbouring bins. This is the case because

$$a \geq b \equiv !(b > a), \tag{5.11}$$

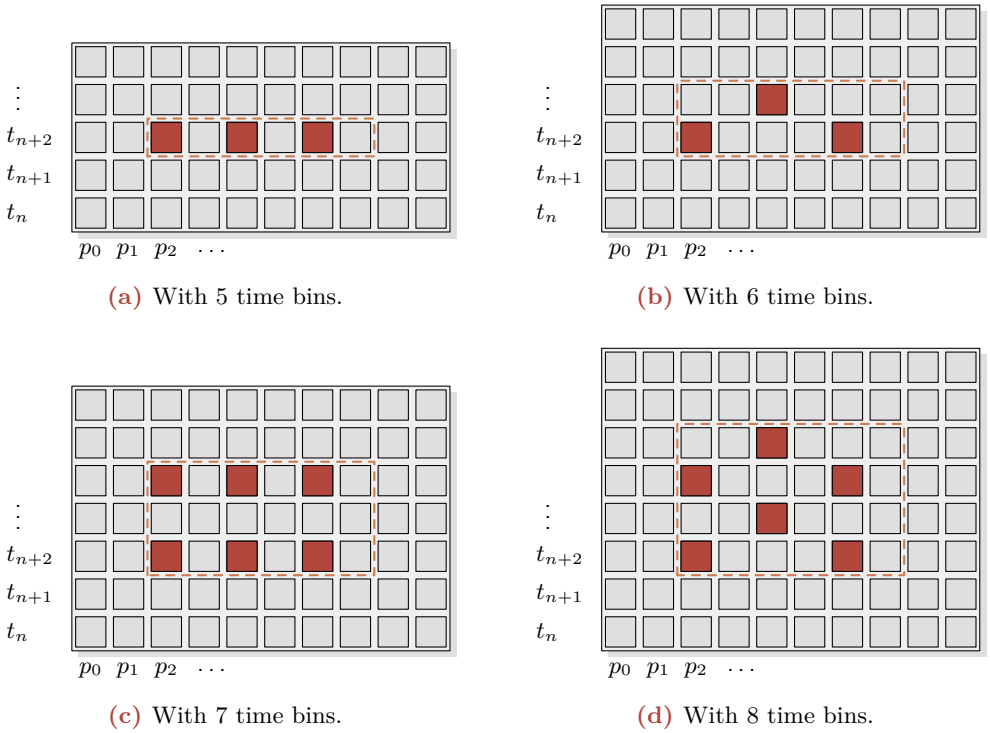
which can easily be proven by setting up a truth table for all possible relations between a and b. Coming back to the peak definition shown in figure 5.15: instead of evaluating e.g. the equation $\text{ADC}(p_n, t_n) \geq \text{ADC}(p_{n-1}, t_{n+1})$ it can be waited until the previous bin (p_{n-1}, t_{n+1}) is shifted by enough positions and becomes the new centre and uses the inverse of the equation $\text{ADC}(p_n, t_n) > \text{ADC}(p_{n+1}, t_{n-1})$ — which is evaluated anyway — as the result for the relation between those two bins. With this, all the $\geq$-operators can be replaced by the corresponding $>$-operators combined with an inverter. So the complete peak finding algorithm is reduced to a shift register and four comparators. This reduces the complexity

bit	content
[13:0]	ADC value
14	peak, maximum in all directions and value above peak threshold
15	ADC value above contribution threshold
16	minimum in pad direction
17	minimum in time direction
18	minimum in diagonal $(p_{n-1}, t_{n+1}) \Leftrightarrow (p_{n+1}, t_{n-1})$
19	minimum in diagonal $(p_{n-1}, t_{n-1}) \Leftrightarrow (p_{n+1}, t_{n+1})$

Table 5.4: Content of the data word stored in the CF memory. The 14 LSB are the original ADC value, the 6 MSB contain flags for the bin characterisations.

of the system significantly because all values are just shifted through the registers in the order they arrive and no intelligent ordering of read addresses must be developed to minimise the overall processing time while still ensuring that all clusters are found.

In addition, this approach also opens up the possibility of directly recording all minima together with the peak finding without additional latency. For a maximum, the only important information is whether the current centre bin is a peak or not. Thus, the information whether all relations of the peak definition are true and whether the ADC value is above a configurable threshold or not can be combined into a single bit. The information about a minimum must be propagated in a more differentiated way. First, the detection is analog to the detection of a peak in each direction. But instead of requiring that the relation must be true both times, it must be false. E.g. in pad direction the ADC value of the centre bin must be less or equal to the value of the left pad and less than the value on the right side. Again, this evaluation is done anyhow for the peak finding, the result is just reused in a different way. The difference here is, compared to the peak detection, that the results for the individual directions can not be combined because of the use case later on. It can be used at a later stage, to split nearby clusters. If two clusters are close to each other, their charge distributions will overlap and one of the pads will be a minimum, but not in all directions. With the definition in figure 5.15 four directions can be distinguished, in pad, time and two diagonal ones. The minimum flag for all those four options needs to be propagated along with the ADC value to be able to divide the charge correctly between the two clusters if wanted. In total, five of the six available bits are used for the peak and minimum flags. The last one will be used to indicate whether the charge of the pad is above another configurable threshold, the contribution threshold. The charge distribution of a cluster is in first order Gaussian shaped in pad and time direction [5]. Assuming such a distribution, it is clear that if the charge of the bin next to the central bin is below some small threshold, then the charge of the bin even further away should be ignored. The charge contribution from the Gaussian distribution will be negligible and only noise would be added. To do so, this flag can be used to avoid another comparison operation later. The total content of the 20 bit word, which is written to the local memory of the CF, is given in table 5.4. With that, it can later be checked with a single bit comparison whether this currently read ADC value is a peak centre, a minimum in some direction or should be excluded from the calculation of the cluster properties.



(a) With 5 time bins.

(b) With 6 time bins.

(c) With 7 time bins.

(d) With 8 time bins.

Figure 5.16: Maximum number of peaks within a CF instance for different number of time bins and always ten pads.

5.3.4 Width of the Cluster Finder Instances

It was already mentioned that the optimal size of a single CF instance still needs to be defined. The number of pads processed by one instance and the number of buffered time bins need to be chosen to not use too many resources — the CF grid will be one of the biggest components of the CRU FW — and to be able to process the input data continuously. Due to the continuous readout, it is not possible to implement something like a busy mechanism and interrupt the data taking for a short time to complete the processing of the current chunk of data. The time bins arriving with a frequency of 5 MHz need to be processed in time. With a baseline clock of 240 MHz, $\frac{240\,\text{MHz}}{5\,\text{MHz}} = 48$ CCs are available to complete each time bin.

Since the peak finding itself is done before buffering the data, the most time consuming part is to read all the 25 bins of the 5×5 cluster matrix after a peak is found. Simply reading all the values of one cluster from the RAM takes 25 out of the 48 CCs, more than half of the available time. If this takes that much time, then the first question which arrises is: how many clusters can occur in the region of interest? To answer this question, some examples with different CF sizes are shown in figure 5.16. The number of pads is kept constant at ten, here still for no particular reason, but the number of time bins increases from five to eight. The inner region in which a peak is allowed to be detected is again shown by an orange dashed rectangle and has always a distance of two bins to the borders. First, the focus is on the example with the five time bins in figure 5.16a. Five is the number of time

bins needed to complete a cluster of this size, so at least this amount needs to be buffered. Finding a peak means that the ADC value of one pad is higher than the value of all the surrounding ones (according to the definition in figure 5.15). Therefore, there must be at least one pad with a lower ADC value in-between two peaks. This means that in the shown case of five time bins and ten pads, at most three peaks can be found in the inner region with a length of six pads. Example positions are shown by red rectangles in the figure. To read those clusters, $3 \cdot 25$ CCs $= 75$ CCs would be needed in a simple implementation where always all corresponding pads are read from the RAM. This is far beyond the available 48 CCs, so this combination of ten pads and five time bins does not work.

Adding one additional time bin, as it is done in figure 5.16b, does not change the number of possible peaks within the inner region, but the available processing time is doubled. Instead of 48 CCs there are 96 CCs available because always two time bins are processed at once, and the array shifted afterwards by two time bins. So by adding another time bin, the processing of this amount of pads becomes possible all at once. Only after adding a seventh time bin (figure 5.16c), the number of possible peaks within the region is increased by a factor of two. Adding an eighth time bin (figure 5.16d) does again not change the maximum number of peaks. This can be generalised: it is preferable to have an even number of time bins processed (and stored) because then the number of peaks (and with that the biggest contribution to the overall porcessing time) stays the same compared to one time bin less, but the available number of CCs is increased. Already from this simple consideration it can be concluded that the configuration with six time bins could be an optimal case. It requires the least number of buffered data, and with that the smallest resource consumption, while having the advantage of an increased processing time available. The same considerations can also be done in pad direction with the conclusion that also here an even amount of pads is preferable. A complete analysis of all possible size combinations is shown in figure 5.17. For each combination, the available processing time was taken, meaning that the number of time bins within the central region multiplied with 48 CCs, and divided by the number of CCs needed to read out the maximum number of clusters. The second one is just the number of possible peaks multiplied with 25 CCs, 1 CC to read each bin. Any combination that would be realisable (with a ratio of ≤ 1) was filled in the histogram. Those steps of two can clearly be seen, both in time and pad direction, which were discussed before. Three different combination could be optimal (having the same number of pads but just more time bins is definitely worse), 5 time bins with 6 pads, 6 time bins with 10 pads and 7 time bins with 8 pads. Further increasing the number of time bins does not seem to improve the situation. The CF can not be made wider in pad direction, only more resources are needed to buffer more time bins. With this argument, the last combination can also be ruled out as the optimal one. Compared to the combination with 6 time bins, the CF is shorter in pad direction. This is the reason why more CF instances are needed to cover all 138 pads of a row ($\lceil 138/(8-4) \rceil = 35$[§] compared to $\lceil 138/(10-4) \rceil = 23$) and in addition one more time bin needs to be buffered. Having the CF shorter in pad direction does not only increases the number of CF instances (the resources needed per instance is at this point still to be seen), but also the number of doubled pads is increased. For the last combination, 7 time bins of 35×8 pads $= 280$ pads need to be stored, whereas for the second combination only 6 time

[§]Those brackets symbolise the *ceil*-function, ceil(3.3) $= \lceil 3.3 \rceil = 4$.

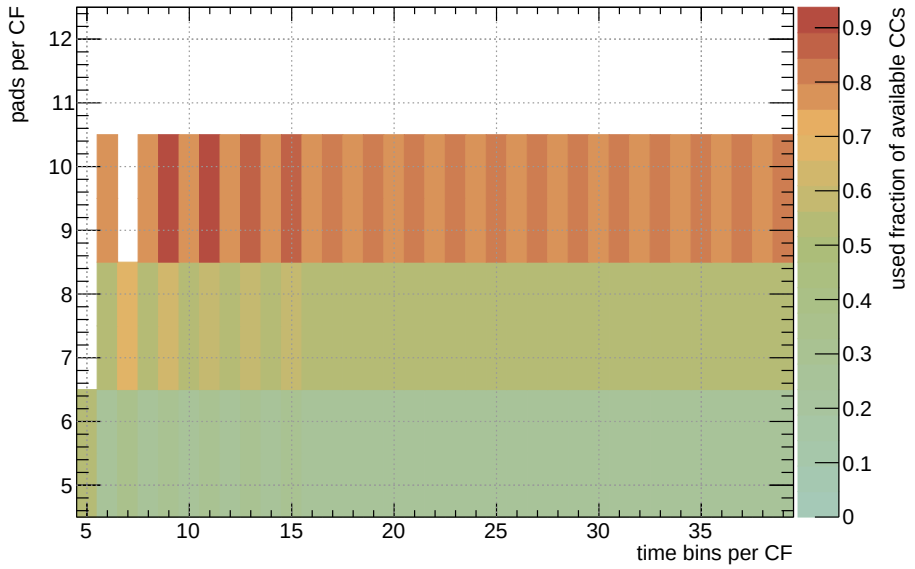


Figure 5.17: The colour code shows the used fraction of available CCs as a function of pads and time bins per CF. Only the possible combinations with fractions of ≤ 1 are filled.

bins of 23×10 pads $= 230$ pads are needed, which is a clear advantage. Doing the same estimation for the first possible combination yields to $138/(6-4) = 69$ CF instances needed to cover a full row and to 5 timebins of 69×6 pads $= 414$ pads which need to be buffered. Multiplying these numbers with the width of the ADC value plus the needed flags of 20 bit, one finds that $5 \times 414 \times 20$ bit $= 41.4$ kbit need to be buffered for the first combination in total and $6 \times 230 \times 20$ bit $= 27.6$ kbit for the second. To conclude, the second combination with 6 timebins and 10 pads is better in both, the number of CF instances and the size of the needed local storage, making this the optimal size.

5.3.5 The Cluster Finder Module

After it is clear now how the CF must be dimensioned and by which method the peak finding must be done, with the actual implementation can be continued. The central element of the CF is the RAM to store the data over 6 timebins. Before that, there is the *peak finder* located, consisting of a shift-register and the necessary comparators which adds the additional flags given in table 5.4 to the data word. This arrangement is schematically shown in the block diagram in figure 5.18. A *write controller* takes care of the correct shifting of the individual pads and calculates the write addresses for the RAM. Since ten pads need to be stored, it fits very well to use the decimal system for the read addresses. Then, the ones digit gives the pad position within the short row segment and the tens digit counts the time. To avoid unnecessary copy operations at the start of a new time bin, the RAM is used as a ring-buffer where the data is continuously written to, independently of the read process. To be able to decouple the write from the read processes, there must be enough space for two more time bins in the memory to always ensure a valid content in the memory. The two additional time bins are simply needed because the read controller has

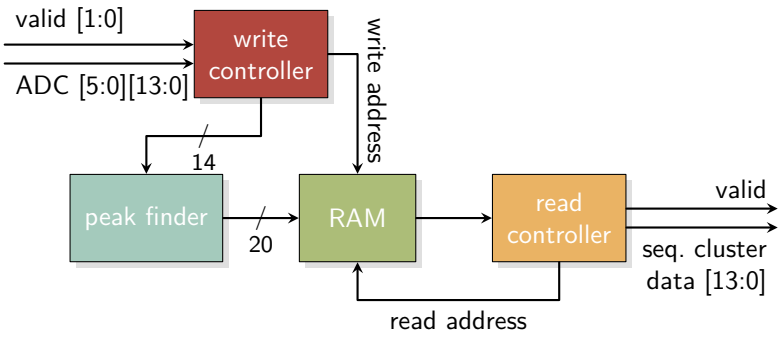


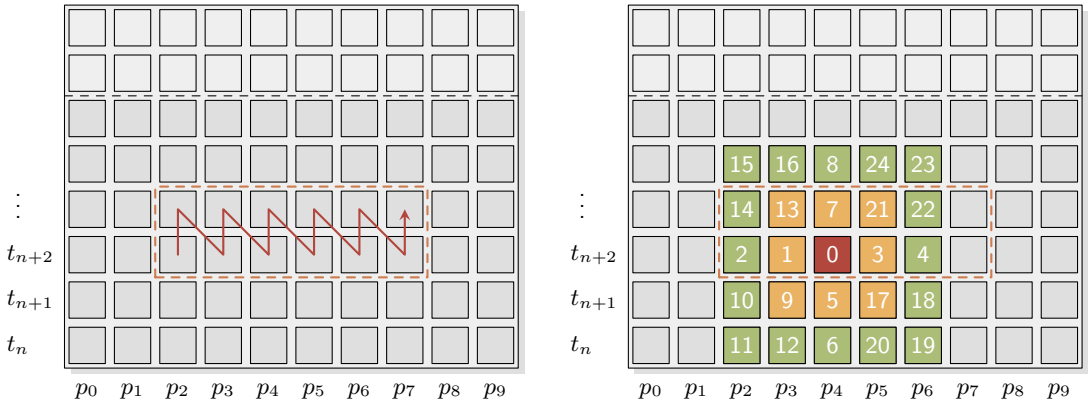
Figure 5.18: Block diagram of the CF module. A write controller takes care of the correct shifting, the peak finder and calculates the write addresses for the RAM. The read controller checks for peaks and reads all corresponding bins of a cluster from the storage.

to wait until two complete time bins were written to the RAM before it can start to process those. Since the continuous input stream does not stop while the reading is performed, the storage for two additional time bins is needed so that they can be written somewhere. The total amount of needed addresses in the storage therefore sums up to 80 which fits into three MLABs, each providing 32 addresses, instead of the previously mentioned two.

The *read controller*, on the other hand, takes care of the correct reading procedure. It is implemented as a FSM and ensures that all peaks are found within the inner region of the CF and that all bins belonging to the cluster are read in a predefined order. For this, the addresses of the pads within the inner region are asserted to the read address port of the RAM in the order shown in figure 5.19a. This zigzag pattern, starting from the bottom left and going to the top right, was chosen for optimisation reasons. The calculation of the address is not different compared to other patterns because an adder needs to be involved anyhow. In case a peak is found, two to three pads can be skipped for the search process of the next one — depending on the time bin in which the peak is found — simply due to the definition of a peak that two peaks can never be at two adjacent pads. If the pattern would be more simple, like always reading a complete time bin after the other, only one pad could be skipped or an additional logic group would be needed to keep track of the already found peaks to omit the pads in the next time bin which were already part of a peak.

After a peak was found, the remaining 24 pads need to be read from the memory and forwarded to the next module, the CP, to compute the cluster properties there. This order needs to be defined und must be always the same so that the CP knows which pad arrives when, relative to the centre. Since the output of the RAM is sequential, this is also kept for the output port of the CF, one 14 bit word after the other. The additional 6 bit of the peak and minimum flags are not needed in the CP, since the corresponding modifications of the ADC values, like setting the bin to zero if it should be ignored, are implemented in the read controller. This simplifies the implementation of the CP later on. The order in which the pads are read from the memory is illustrated in figure 5.19b. This order is based on two considerations:

1. Reading from the inside out. The inner pad (yellow) is always read first. Thus, if this pad has the flag *above contribution threshold* not set, the outer pad/s (green)



(a) The central peaks are found in this order. Going from the bottom left in a zigzag pattern to the top right.

(b) The data of a peak is read (and forwarded to the CP) in this order to exclude the outer (green) bins if the inner one (yellow) is below the contribution threshold.

Figure 5.19: Reading order of the CF. On the left side is shown in which order it is looked for peaks, while the right figure shows the order in which the individual bins of a cluster are read and forwarded.

is/are set to zero instead of the actual ADC value. It must be distinguished between two cases for the operation, the vertical and horizontal cross in which the inner pad has just one corresponding outer pad, and the four corners in which the inner pad has three corresponding outer pads.

2. Have as few time bin crossings as possible while reading the four corners. It must be ensured for each time bin crossing that the new address is still valid between 0 and 79. If adding or subtracting ten for the time bin crossing leads to an invalid address, it must be mapped back into this range. Since the memory is used as a ring-buffer, such cases will occur. This is not an issue for the previously discussed zigzag pattern. The start address is set to an even time bin address at the very beginning. Afterwards, there are only four possible cases for the start address at the bottom left: 2, 22, 42 or 62. In each of these options, the second time bin (address +10) is always within the valid address range.

To have all information of a cluster combined, four more items need to be attached to the sequential cluster data stream, the pad and time position of the central pad and the row number of this CF. Also, some additional flags are foreseen which are not used at the moment but can be utilised at a later time for example to flag clusters were the charge was split between two nearby ones. The pad position of the peak is a 8 bit number to cover the full range of 138 pads and is the sum of an external given pad offset of the CF instance and the pad within the memory. The time is counted internally in the CF whenever a new time bin is added and needs to be a 9 bit number. This time is measured within the so called Heart Beat (HB) frame which has a length of 89.4 μs. With the sampling rate of 5 MHz, this HB frame consists of 447 timebins. The externally set row number needs to cover the range from 0 to 17 and is therefore a 5 bit number. Finally, the flags are set to 8 bit due

item	content
0	ADC value 0
1	H0 (”000000000” & row[4:0])
2	ADC value 1
3	ADC value 2
⋮	⋮
24	ADC value 23
25	ADC value 24
26	H1 (flags[7:4] & ”00” & pad_peak[7:0])
27	H2 (flags[3:0] & ”0” & time_peak[8:0])

Table 5.5: Sequence of the CF output data stream. In addition to the ADC values of the individual pads, there are the row number, the pad and time position of the central peak and additional flags inserted as *header* words H0 to H2.

to the CDF which is described in subsection 5.3.7. This sums up to 30 additional bits, or three additional 14 bit data words which are inserted in the output stream. The sequence is given in table 5.5. The row number is sent as the second item after the ADC value of the peak because it is always available (it is supposed to be a constant after the initial configuration of the CRU) and can therefore be used to utilise this one CC. After the peak is detected, the correct read address for the pad marked with a 1 in figure 5.19b needs to be asserted to the RAM. Until the value is returned, one CC of latency has passed which is not wasted in this way. The pad position of the peak needs to be calculated first from the read address and is therefore available only at a later stage. Also, the flags are filled only while reading the content of the cluster. Those are appended together with the time information at the very end.

5.3.6 Optimising the Cluster Finder Grid

Before continuing with the processing of the cluster data, there is another optimisation to be done. It was seen that 23 CF instances are needed to cover all 138 pads of a row with maximum length and the individual regions of a sector have up to 18 rows. Multiplying those two numbers would indicate that 414 CF instances are needed to cover the maximum size area. However, it is possible to improve the numbers since not every region has those 18 rows and not every row — not even in the outermost region — has really the maximum number of 138 pads. All the CF instances which are actually needed are sketched in figure 5.20. The 18 rows are shown vertically and the 23 CF instances horizontally as grey boxes with a width of six pads corresponding to the number of pads within the inner region of each instance. The maximum pad number which is shown, was found by going through all the ten regions and taking the maximum number of pads for each individual row. Since clearly only those pads need to be covered, it would be a waste of resources to also instantiate CF instances for the remaining 38 white boxes, leaving only the 376 coloured CFs. However, there is still room for improvement. Those CFs are not needed all at the same time. Having a look at the pad planes of the regions individually (figure B.1 to

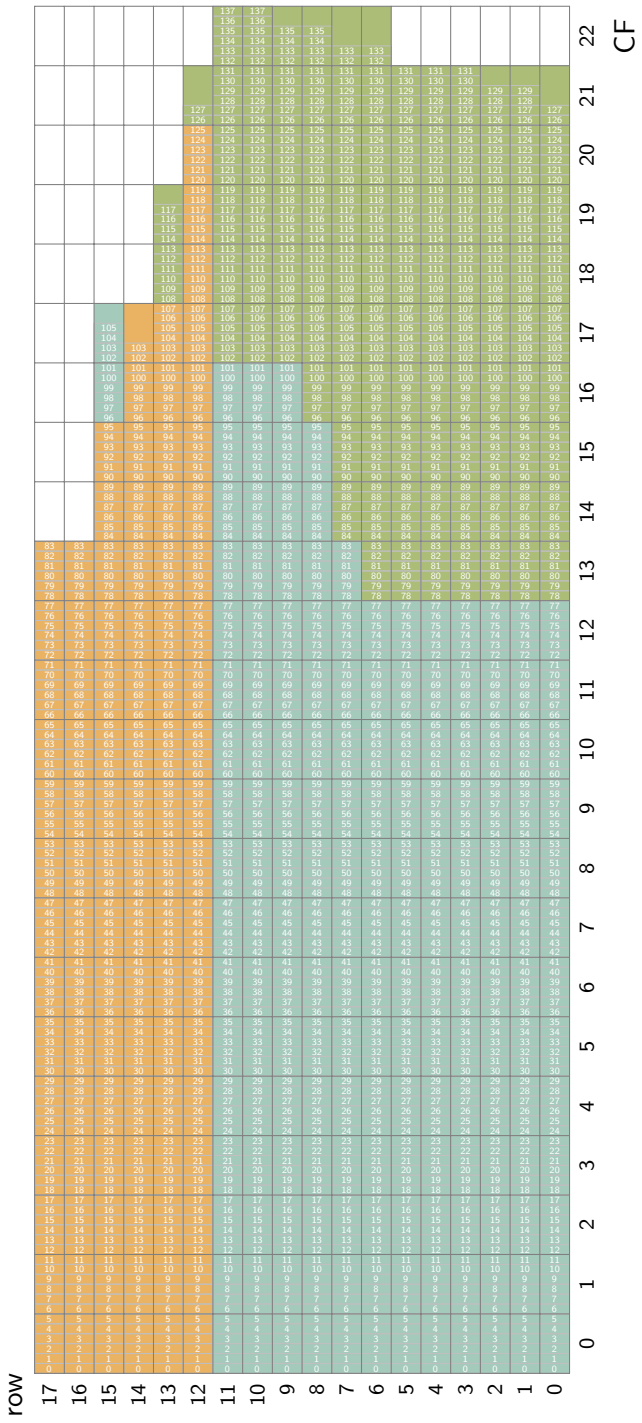


Figure 5.20: The CF grid covering all required pads. Each rectangle represents the inner region of a single CF instance. The 38 white instances are not needed, there is no region with the corresponding row with this number of pads. Those regions where the green marked fields are needed, the yellow ones are not and vice versa. So the 101 green marked CFs can be reused at the positions which are marked in yellow.

figure B.4), one finds that there are some with many rows and few pads per row and others with a smaller number of rows but therefore many pads per row. For example, region 4 has all the 18 rows but only a maximum of 84 pads in those rows, in contrast to region 9 with only 12 rows but therefore up to 138 pads. Those coloured boxes are not all needed in a single region. Therefore, it can be tried to instantiate a CF only once in the FW and use it in different configured regions for different areas of the pad plane. Requiring that one instance is used at most for two different positions — to keep the combinatorics as small as possible — the pattern shown in figure 5.20 can be achieved. The 101 instances which are marked in green can safely be reused in the yellow positions without influencing the functionality. These positions are never used both within a single region while the blue ones are needed in most of the regions. The exact mapping between the green and the yellow boxes, depending on the individual region, is not shown for better visibility.

With those two optimisation steps, the number of instances needed can be reduced by more than 33 %, from 414 to only 275 while requiring that a CF has at most two different data sources. This number of 275 is only 3 CF more than the absolute minimum of 272 CF which are needed in regions 6 and 7 to cover all pads available there.

5.3.7 Calculating the Cluster Properties

Five quantities need to be calculated for each cluster. First, the total charge

$$q_{\text{tot}} = \sum_{i=0}^{24} q_i \tag{5.12}$$

which is the sum over all ADC values of the 5×5 cluster matrix. Since the total charge depends on the energy deposit of the original particle, which is described by the Bethe-formula [27], it can be used for Particle Identification (PID). In the following, all summations over the index i are done from $i = 0$ to $i = 24$. The four further quantities are for the position information of the cluster. The actual position is given by the so called *centre of gravity* in pad and time direction which is the weighted mean. It is calculated in the following way:

$$\mu_x = \frac{\sum_i q_i \cdot x_i}{q_{\text{tot}}}, \tag{5.13}$$

where x_i stands for either p_i or t_i, the individual pad numbers and time bins of the charges q_i. The width of the position, given in units of pad and time bins, can be expressed by the standard deviation of the weighted mean:

$$\sigma_x^2 = \frac{\sum_i q_i \cdot (x_i - \mu_x)^2}{q_{\text{tot}}} \tag{5.14}$$

$$= \frac{\sum_i q_i \cdot x_i^2}{q_{\text{tot}}} - \mu_x^2. \tag{5.15}$$

The second form is easily derived by expanding the square in the first equation or by directly using the identity $\text{Var}(X) = \text{E}[(X - \text{E}(X))^2] = \text{E}(X^2) - [\text{E}(X)]^2$ [59]. Equation 5.15 is better suited for an implementation in an FPGA because the mean value needs to

be known only at the very end and not in each summand. In general, those sums are very well suited for an implementation in an FPGA because the accumulation can be split across several CCs. Although the summations are well suited, the multiplications, squares and divisions of equation 5.13 and 5.15 are not. Since each arithmetic operation must be reduced to basic bit operations, more complex calculations require a lot of resources for the implementation and time (in form of CCs) for the execution, especially when it comes to numbers with many bits. By expressing the position relative to the central peak, the calculation of the mean value can also be done in the following way:

$$\mu_x = \frac{\sum_i q_i \cdot x_i}{q_{\text{tot}}} \tag{5.16}$$

$$= \frac{\sum_i q_i \cdot (x + \delta x_i)}{q_{\text{tot}}} \tag{5.17}$$

$$= x + \frac{\sum_i q_i \cdot \delta x_i}{q_{\text{tot}}}, \tag{5.18}$$

where x is the pad or time position of the central bin and $\delta x_i \in \{\pm 2, \pm 1, 0\}$ is the distance between the peak and the individual charges. This reduces the multiplication to a bit-shift operation by one in case of a multiplication by two. The sign can be implemented either by calculating the two's complement of the eventually shifted charge or by using an adder which can also be configured as a subtractor for the calculation. An additional benefit is that also the number of summands decreases since there is five times a multiplication by zero involved after expressing the position relative to the peak. The same can also be applied to the calculation of the resolution:

$$\sigma_x^2 = \frac{\sum_i q_i \cdot (x + \delta x_i)^2}{q_{\text{tot}}} - \mu_x^2 \tag{5.19}$$

$$= \frac{\sum_i q_i \cdot \left(x^2 + 2x \cdot \delta x_i + (\delta x_i)^2\right)}{q_{\text{tot}}} - \mu_x^2 \tag{5.20}$$

$$= x^2 + 2x \cdot \frac{\sum_i q_i \cdot \delta x_i}{q_{\text{tot}}} + \frac{\sum_i q_i \cdot (\delta x_i)^2}{q_{\text{tot}}} - \mu_x^2. \tag{5.21}$$

By using equation 5.18, the calculation can further be reduced to:

$$\sigma_x^2 = x^2 + 2x \cdot (\mu_x - x) + \frac{\sum_i q_i \cdot (\delta x_i)^2}{q_{\text{tot}}} - \mu_x^2 \tag{5.22}$$

$$= x^2 - 2x^2 + 2x\mu_x - \mu_x^2 + \frac{\sum_i q_i \cdot (\delta x_i)^2}{q_{\text{tot}}} \tag{5.23}$$

$$= -(x - \mu_x)^2 + \frac{\sum_i q_i \cdot (\delta x_i)^2}{q_{\text{tot}}} \tag{5.24}$$

and again using equation 5.18 leads to

$$\sigma_x^2 = \frac{\sum_i q_i \cdot (\delta x_i)^2}{q_{\text{tot}}} - \left(\frac{\sum_i q_i \cdot \delta x_i}{q_{\text{tot}}}\right)^2. \tag{5.25}$$

It can be seen that the dependency on the central pad drops out completely and the resolution depends only on δx_i and $(\delta x_i)^2 \in \{4, 1, 0\}$, apart from the charges q_i and q_{tot}. The sum of the second part was already calculated for μ_x and can be reused.

Since all the clusters need to be accessed in the FLP anyway, for a reordering, aggregation and a compression before sending them over the network, it is possible to outsource parts of the calculation to the CPU of the FLP. In particular, the divisions by q_{tot} and the square root operation to get σ_x can be done in software more easily. Thus it was agreed that the summations, given in equation 5.26, are done as precalculations in the CRU:

$$q_{\text{tot}} = \sum_i q_i \qquad \begin{aligned} \mu_{p,\text{pre}} &= \sum_i q_i \cdot \delta p_i & \sigma_{p,\text{pre}} &= \sum_i q_i \cdot (\delta p_i)^2 \\ \mu_{t,\text{pre}} &= \sum_i q_i \cdot \delta t_i & \sigma_{t,\text{pre}} &= \sum_i q_i \cdot (\delta t_i)^2. \end{aligned} \tag{5.26}$$

The values are then shipped to the FLP where the final calculations, given in equation 5.27, are done during the data handling in the CPU:

$$\begin{aligned} \mu_p &= p + \frac{\mu_{p,\text{pre}}}{q_{\text{tot}}} & \sigma_p &= \sqrt{\frac{\sigma_{p,\text{pre}}}{q_{\text{tot}}} - \left(\frac{\mu_{p,\text{pre}}}{q_{\text{tot}}}\right)^2} \\ \mu_t &= t + \frac{\mu_{t,\text{pre}}}{q_{\text{tot}}} & \sigma_t &= \sqrt{\frac{\sigma_{t,\text{pre}}}{q_{\text{tot}}} - \left(\frac{\mu_{t,\text{pre}}}{q_{\text{tot}}}\right)^2}. \end{aligned} \tag{5.27}$$

In addition to those five quantities, also the position of the peak in pad and time direction needs to be included in the CDF, as well as the row number, the already mentioned flags and the charge of the peak, the so called q_{max}. The q_{max} depends also on the original energy loss of the particle and can therefore be used as a complementary method to identify the particle. The bit width needed for this value is in the order of the original 10 bit ADC value but one additional bit is kept for a better precision due to the already applied baseline correction. The widths of the flags, the row and the peak positions were already discussed. For the other five quantities, it was decided to keep the maximum precision in order not to bias the computations in the FLP. This means that for the total charge q_{tot}, to which up to twenty-five 14 bit values can contribute, a width of 19 bit is needed. For the precalculations of $\mu_{x,\text{pre}}$, this looks different. Going through the 5×5 matrix of the cluster, one finds that ten values will contribute with a factor of $|\delta x_i| = 1$ and ten values with a factor of $|\delta x_i| = 2$. The remaining five bins are not taken into account because of a factor of $|\delta x_i| = 0$. The maximum possible value fits therefore in a 19 bit number. One additional bit is needed to encode the sign, since the value could also be negative, giving in the end a 20 bit number. The $\sigma_{x,\text{pre}}$, on the other hand, has a factor of $\delta x_i = 4$ involved, this is why the maximum possible result is also a 20 bit number. However, this time without a sign, since all multiplications are done with positive numbers. Taking all these widths into account one finds that the CDF needs to have at least 140 bit. Since it is very convenient to work in software with 32 bit words, the size of a CDF was fixed to five words, giving a space of 160 bit for the cluster with 20 spare bits for future developments. How the individual numbers are distributed within these words is shown in figure 5.21.

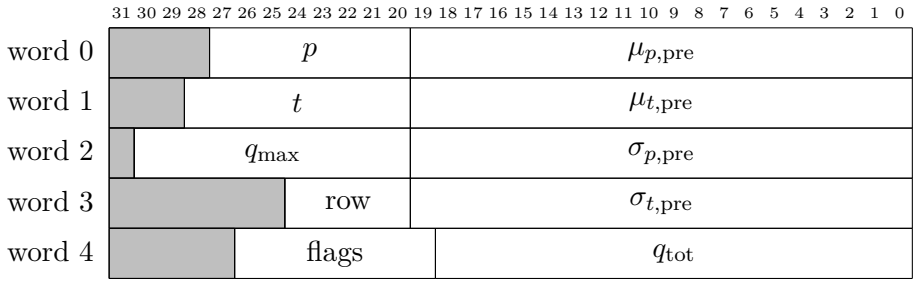


Figure 5.21: The Cluster Data Format (CDF), consisting of five 32 bit words, contains all relevant information about a cluster. The grey marked bits are currently unused and must be set to 0.

Since the multiplications are trivial now, again an accumulation problem remains. This kind of problem was solved before efficiently by utilising DSP blocks in the CM calculation. Since the individual sums of equation 5.26 are decoupled from each other, it makes sense to use one DSP block for each. In this case, again the "18 × 18 Sum of 2" mode of the available DSPs, shown already in figure 5.11a, has exactly the form which is needed. It has many inputs, the pre-adder can be used either as adder or as subtractor and it can accumulate internally a lot of data. The input port widths of 18 bit and 19 bit are also fine, even by shifting the 14 bit input ADC value by two bits for the multiplication by four and the output port width of up to 64 bit is also sufficient for each of the results of the sums. The question arises how to utilise all of the four input ports (two for both pre-adders) in order to use the DSPs efficiently. The solution is to put an appropriate FIFO in-between the CF and the CP. There are a few reasons speaking in favour of using an M20K for this. First, it is rather big which reduces the danger of lost clusters due to a buffer overflow. Second, and more important, those RAM blocks can be configured with different widths of the write and read ports. Having the read port four times as wide as the write port allows to read four ADC values in one CC. This factor of four fits perfectly the number of input ports of the DSPs. Even the number of words that belong to a cluster is 28, which can be divided without remainder by four. So such a configuration of having a mixed-width FIFO in the beginning to always read four values at once and then a DSP which can accumulate four values at once, fits perfectly. With this, the CP can also process the data four times as fast as the data is transmitted by the CF. Keeping this in mind, each CP instance can be used to process and collect the output of four CF instances. Since the data of all the 275 CFs need to be merged anyhow at some point to end up with two PCIe endpoints, this is a perfect first step of doing so. Then, additionally it reduces the resource consumption, especially the number of needed DSPs, if not every CF needs its individual CP. Having multiple inputs to each CP requires then again a FIFO for each input which are processed round-robin.

With this, the CP can be designed as shown in figure 5.22. Four input FIFOs are shown, each is filled by a different CF. The FIFOs are emptied by a *controller* which takes care that always a FIFO with some content is read. In addition, those which are almost full are preferred for the next reading cycle. It also feeds the individual DSPs, each responsible to accumulate the charges for a different sum and writes the formed CDF to an output FIFO.

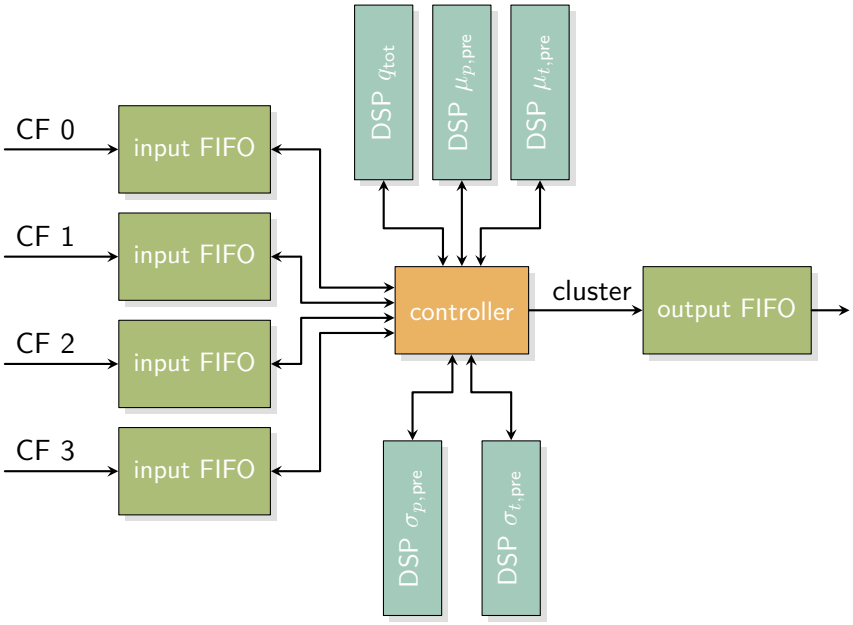
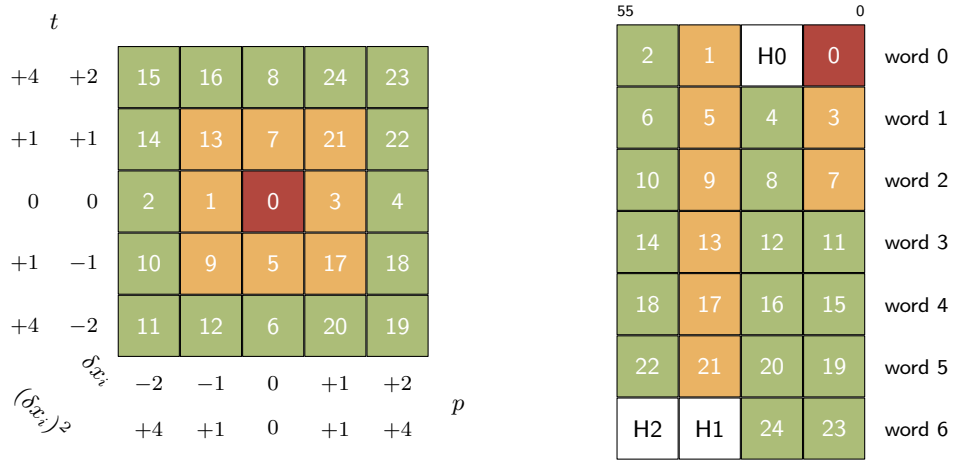


Figure 5.22: Block diagram of the CP. Each CF writes its data to a FIFO. A single CP instance reads from four of those round-robin, calculates the cluster properties with the help of DSP blocks and forms the CDF which is written to an output FIFO.

The controller itself is composed of two small FSMs, one for the reading process and one for the calculation and write process.

The read-FSM must ensure that always $28/4 = 7$ words are read from the same FIFO. Otherwise the charges of different clusters would get mixed up. Additionally it needs to always find the next FIFO with content to ensure an efficient processing while the FIFO which is almost full must be preferred. The FSM also takes care of the masking of the input FIFOs if a so called HB-pattern was seen, until this was seen in all input FIFOs. This is necessary to provide synchronicity by keeping all clusters belonging to the same HB-frame together. Each of the CF is working independently of the others. They all receive the HB signal which resets the internal time counter and triggers the generation of this special reset-pattern. It is a 56 bit wide pattern (4×14 bit) which is inserted into the normal data stream. Doing it this way makes it on one hand necessary to detect the pattern but on the other hand simplifies the data handling significantly because no additional path needs to be kept synchronously within the individual data streams. The controller is designed in a way that it could in principle handle an arbitrary number of input paths, one just needs to ensure that the data can be processed fast enough in order not to generate back-pressure to the CFs. The output of the CP is again a FIFO, making it easily possible to put the CP in an individual clock domain, faster than the baseline clock of 240 MHz. The Clock-Domain Crossing (CDC) is then done with dual-clock FIFOs on both sides. By increasing the frequency by $1/4$ to 300 MHz, one could add a fifth input path which would reduce the number of needed CPs from $\lceil 275/4 \rceil = 69$ to $\lceil 275/5 \rceil = 55$ and with that the needed resources. However, it was found that there are enough resources available for the current design and that therefore this optimisation is not needed at the moment. Keeping a lower frequency reduces the



(a) The cluster matrix with the δx_i and $(\delta x_i)^2$ next to the corresponding bins.

(b) The seven data words from the CP input FIFO, containing 4 ADC values each.

Figure 5.23: Mapping of the cluster data to the CP input FIFO.

effort needed by the fit and routing algorithms during the compilation procedure to achieve timing closure. It should be kept in mind that the module is written in this configurable way and the number of inputs can be set by a simple *generic* parameter.

The calculation-FSM, on the other side, must ensure that the input data to all four ports of the five DSPs is always valid. From the input FIFOs, always four ADC values are read at once, so they must fit to the requirements of the DSPs. For the q_{tot} summation, this is straight forward. The corresponding DSP is configured that all four input ports are summed up (both *Pre-adders* are configured for an adding operation) and accumulated over seven CCs. Only the header data, containing the additional cluster information, in word 0 and word 6 of figure 5.23b must be excluded by setting the input to 0 in those cases. This figure shows how the individual cluster bins are grouped into the seven words of the mixed-width FIFO output. Each word contains four ADC values. The readout order of the cluster charges is shown again in figure 5.23a. Starting from the centre, first the horizontal and vertical axes are read, afterwards the four corners in the shown order. The matching pre-factors for the mean and sigma summations (δx_i and $(\delta x_i)^2$) are written below for the calculation in pad direction and on the left side for the calculation in time direction. They become important now for the computation of the weighted mean. For the sake of simplicity, we concentrate on the pad direction. It works analogously in time direction. To calculate the sum, each of the pads in the columns above $|\delta x_i| = 2$ needs to be multiplied by two and then either subtracted from or added to the sum. To subtract a value, one could either calculate the two's complement which is then added, or use the available *Pre-adders* of the DSP in subtraction mode. Using the second option gives a DSP with two input ports which are always added to the sum and two ports which are always subtracted from the sum. This appears in principle to be perfectly fine since also half of the bins need to be added and the other half to be subtracted. However, to make the whole process work, there must be at most two values for the addition and two values for subtraction in each word because the setting of the *Pre-adder* must be known at compile time and can not be

δx_i	$(\delta x_i)^2$	word 0	word 1	word 2	word 3	word 4	word 5	word 6
(-2)	$(+4)$	2		⑩	⑪,14	15		
(-1)	$(+1)$	1		⑨	⑫,13	16		
$(+1)$	$(+1)$		3			⑰	⑳,21	24
$(+2)$	$(+4)$		4			⑱	⑲,22	23

(a) *Advancing* individual bins to fit the requirements of the CP processing in pad direction. The encircled numbers are shifted one word ahead (or the current word is delayed by one).

δx_i	$(\delta x_i)^2$	word 0	word 1	word 2	word 3	word 4	word 5	word 6
(-2)	$(+4)$		6		⑪,12		⑲,20	
(-1)	$(+1)$		⑤	⑨,10		⑰,18		
$(+1)$	$(+1)$			⑦	⑬,14		㉑,22	
$(+2)$	$(+4)$			8		⑮,16		㉔,23

(b) *Advancing* individual bins to fit the requirements of the CP processing in time direction. The encircled numbers are shifted one word ahead (or the current word is delayed by one).

Table 5.6: Mapping of the cluster data to the DSP ports.

changed during run time. Unfortunately, that is not the case, as can be seen in table 5.6a for the pad direction (and table 5.6b for the time direction). Going through the individual words from 0 to 6, the relevant bins — those not multiplied with zero — are sorted into the row with the correct pre-factors. The problematic ones are words 3 and 5 in pad direction, where four values are contained which all need to be subtracted or added. To overcome this issue, a shift register can be build with only two elements for the words. With this, it is possible to select from two consecutive words for the DSP input, making it possible to effectively *shift* some values to the previous word. By applying those shiftings, which are indicated by arrows, to the encircled bins, it can also be achieved that indeed in each word at most two values are contained which need to be subtracted and two which need to be added to the total sum. Further, it can also be achieved that only a single value needs to be multiplied by $+2$ and a single value multiplied by -2 in each word. This allows to hardcode the bit-shifting for the multiplications and assign them to the individual ports of the DSP. To summarise, with the trick of delaying the words by one CC, a DSP can be utilised were one input is used for the contributions with $\delta x_i = -2$, one port with $\delta x_i = -1$, one port with $\delta x_i = +1$ and one port with $\delta x_i = +2$, and thus compute $\mu_{p,\mathrm{pre}}$.

The same method can also be used for the $\sigma_{p,\mathrm{pre}}$ DSP with the only difference that the *Pre-adders* are not used for a subtraction and that the multiplication-bit-shift is by two instead of one. The calculation in time direction is analogous, only other bins (shown in table 5.6b) need to be assigned to the respective DSP ports. With this, all the needed cluster properties can be calculated in an efficient and resource-saving way.

The CF is required to insert a special pattern in the data stream upon receiving the HB trigger. After the CDF is formed, such an approach is not needed anymore. From here on, a single not yet used bit in the flags-field is used to identify a so called reset-cluster in the following logic. After all input FIFOs present the reset-pattern, such a reset-cluster

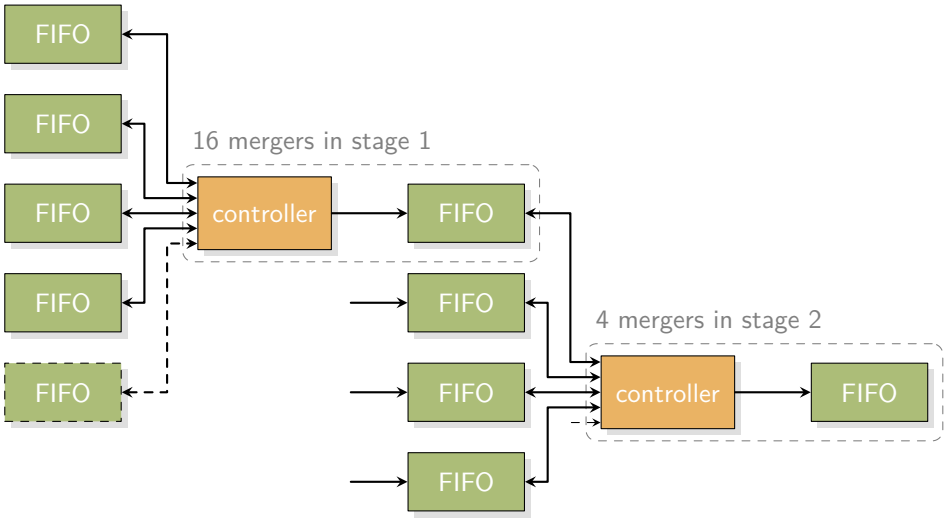


Figure 5.24: The FIFO merging network. The sixteen mergers of stage one working with four or five input FIFOs while the four mergers of stage two all have four input FIFOs.

is generated and written to the output FIFO to mark the transition from one HB to the next one.

5.3.8 Cluster Merging Network

The last component of the UL, before the data can finally be transmitted by the DMA engine, is the merging network. As described before, there is a huge CF grid consisting of 275 independent instances. A first step of merging the individual output data streams is already done with the CPs. Each of those modules handles the data from four CFs and merges the corresponding streams to one. But there are still 69 CPs left and only two PCIe endpoints. Those two PCIe endpoints have a bus width of 256 bit each. The data which needs to be transferred are clusters with a width of 160 bit. It can directly be seen that to utilise the available bandwidth to the host machine most efficiently, access to two clusters is needed at the same time to fill the available 256 bit. Since those two endpoints need to be independent from the point of view of the software in the FLP, a cluster can not be split and some part of the data written to one and another part to the other endpoint. Therefore, access to two clusters is needed for each of the endpoints or to four final output streams. That means, the 69 streams of the CPs need to be merged to four final streams. This does not need to be done in one step but can be implemented in multiple stages.

By reusing parts of the logic of the CP, small merger instances can be built which read the clusters from 4 or 5 FIFOs and write them into a single one. With those sizes, the merging is achieved in two stages ($69 \Rightarrow 16 \Rightarrow 4$), which is indicated in figure 5.24. The first stage of the merging network needs in total sixteen mergers, eleven instances working with four inputs and the remaining five with five different input FIFOs. The second stage consists of four final mergers, each combining the data of four streams.

A single merger consist of two parts, a controller and an output FIFO. The controller takes care of selecting the correct input and detection of the reset-cluster, after which

Family	Arria 10
Device	10AX115S3F45E2SG
Timing Models	Final
Logic utilisation (in ALMs)	362 946 / 427 200 (85 %)
Total registers	638 092
Total pins	342 / 960 (36 %)
Total virtual pins	0
Total block memory bits	26 633 212 / 55 562 240 (48 %)
Total RAM blocks	2 314 / 2 713 (85 %)
Total DSP blocks	416 / 1 518 (27 %)
Total HSSI RX channels	41 / 72 (57 %)
Total HSSI TX channels	41 / 72 (57 %)
Total PLLs	56 / 144 (39 %)

Table 5.7: Fit result summary of the complete FW with the TPC UL.

the corresponding input is masked until this cluster was seen in all inputs, identical to the logic in the CP. Then again, a new reset cluster is generated at the output and the emptying of the input FIFOs is continued. The module is written in a very generic way. The number of inputs is freely configurable (during compile time) which makes it easy to optimise the network according to the required bandwidth and fluctuations in data rate. All the FIFOs are dual-clock FIFOs which opens the possibility, again similar to the CP, to increase the clock frequency in this part of the UL to speed-up the merging if needed. The CDC is handled again by the FIFOs. In general, it is sufficient for the intermediate FIFOs to be rather shallow, they are only needed to implement the merging in a very convenient way. So if they run full the logic introduces back-pressure on the previous stages, including the CP. Therefore, it is sufficient if the very last and the very first FIFOs are deep: the very first ones to compensate for this back-pressure and the very last one to have enough data available to ensure that the DMA engine does not run empty and precious transmission time is lost. For those two cases, two different FIFOs are implemented in the module which can be selected with a generic parameter: a shallow one with 32 elements and a deep one with 512 elements. With that, the same module can be used for all stages. Since the controller works only with the status signals of the FIFO, like the full and empty flags, the merger module can easily be extended with differently sized FIFOs if at any time in the future it is seen that another size would be more appropriate.

5.4 Resource Consumption

After all modules are ready, the FW can be compiled for the CRU. To anticipate the outcome, the fit is successful and there are enough resources for all the logic of the TPC UL, although the FPGA is quite full with a utilisation of 85 % of the available Logic Elements (LEs). The fit summary as it is usually output by the software[¶] is shown in table 5.7. The design which was

[¶]Intel Quartus Prime 17.0 Pro edition was used.

Module	ALMs	%	M20Ks	%	DSPs	%
UL glue logic	21.2	0.0	0	0.0	0	0.0
Link MUX	2 875.6	0.7	0	0.0	0	0.0
GBT decoder	10 263.7	2.4	0	0.0	0	0.0
Pre-sorter	3 889.3	0.9	40	1.5	0	0.0
Row-segment merger	40 067.4	9.4	0	0.0	0	0.0
BLC	4 248.0	1.0	116	4.3	54	3.6
Clusteriser	167 905.4	39.3	906	33.4	362	23.8
↪ Glue logic	2 251.0	0.5	0	0.0	0	0.0
↪ All 275 CFs	119 219.5	27.9	0	0.0	17	1.1
↪ All 69 CPs	41 688.0	9.8	826	30.4	345	22.7
↪ All 20 FIFO merger	4 749.2	1.1	80	2.9	0	0.0
Readout gate	3 889.7	0.9	0	0.0	0	0.0
Configuration	10 024.0	2.3	0	0.0	0	0.0
$\sum$	243 184.3	56.9	1 062	39.1	416	27.4

Table 5.8: Resource consumption of the individual modules of the TPC UL. The modules of the clusteriser are listed separately.

fitted contains the complete TPC UL and also the common parts of the FW for 24 GBT links. The only missing module is the one for the CM calculation module since the implementation itself was not part of this thesis and the module is not yet ready. Therefore it could not be included in the fit. Although the FPGA is already quite full with a usage of 85 % of the basic LEs, the ALMs, there is still enough space for this additional module. Test compilations showed that the CM calculation module should not need more than a few percent of the ALMs, 21 DSPs for the accumulation and maybe some RAM cells to store the pedestal values.

Table 5.8 lists the resource consumption for the individual modules of the UL in a more differentiated way. For each group of modules, there is the number of used ALMs, the number of used M20Ks and the number of used DSP blocks given, together with the percentage share of all available resources in the FPGA. With this, it can nicely be seen which modules are the biggest consumers. First, there are three more modules listed which were not discussed before in detail, because their implementation is rather simple. That is a link Multiplexer (MUX) with which the 24 input links can be freely assigned to one of the 20 implemented processing paths. There are 24 input links used to provide a higher flexibility during the connection of the FEE and because one fibre trunk contains 24 individual optical fibres. The readout gate is also kept very simple: it composes the Raw Data Header (RDH) and combines the four final FIFOs to the two DMA endpoints (two times 2 $\Rightarrow$ 1) via a small FSM. The configuration module is somewhat more extended, as can be seen in the consumption of the ALMs but is also very simple. It provides the registers to read from and write to by instantiating the corresponding modules provided by the central team.

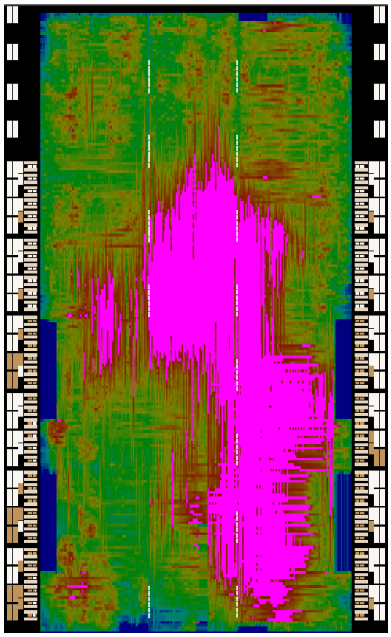
Otherwise, the result is as expected. There are two big consumers, the clusteriser and next the row-segment merger, while the other modules are in principle negligible in terms

of resource consumption. The 119 219.5 ALMs (or 27.9 %) which are needed for the CFs is first a huge number. However, one must consider that those resources are needed for 275 individual modules, leading to an average consumption of 119219.5 ALMs/275 = 433.5 ALMs per CF which is a reasonable number, keeping in mind that the CF consist of a shift register for a whole time bin, a comparator array for the peak finding, the local storage for eight time bins and the relatively complex read controller.

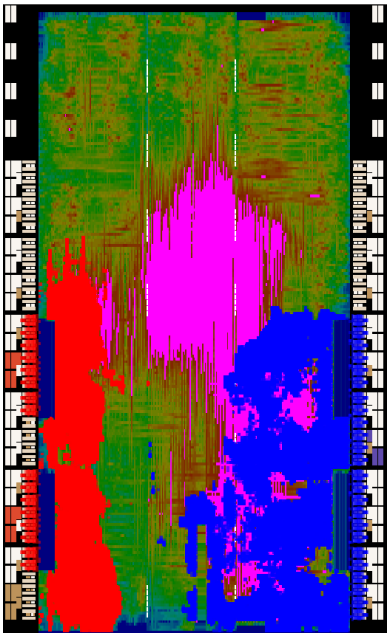
The next type of resources are the DSP blocks. The result is close to the expectation. The 18 BLC modules need three DSP blocks each, giving in total the shown 54. The 69 CPs use five blocks each to accumulate the five different sums which results in 345 needed DSPs. An interesting observation is that in 17 randomly distributed CFs a constant multiplication by 6 which is needed to calculate the pad offset was implemented by the fitter, using a DSP block. This was probably done to reduce the consumption of ALMs and might be a place for further optimisations. Since so far only 27 % of the DSP blocks are used (there are more than 1 000 still available), one could do the complete pad offset calculation (multiplying the externally given CF number within a row by 6 and subtracting 2) in all CFs in this way and save some ALMs with that.

In order to interpret the consumption of the M20Ks, it is important to note that in some cases the fitting algorithm can reduce the utilisation of ALMs by increasing the usage of M20Ks blocks. This is particularly the case when it comes to the implementation of memory. There is also the obvious case of the 20 pre-sorters, each using two M20Ks in a ping-pong RAM configuration, giving exactly the shown 40 blocks. The 116 used RAMs in the BLC modules are initially unexpected, as this means that on average each module uses 116/18 = 6.56 blocks, although it was not intended that the modules would use any at all. The only memory that should be used here is for the LUT and should be implemented in MLABs since they have only few entries. However, the configuration of the instantiated IP core for the RAMs was that it is explicitly left to the fitter to choose the type of RAM to be used. In this case, the algorithm used five M20Ks instead of MLABs for the implementation of this LUTs. The additional one or two RAM blocks are used in some of the modules for a 24 bit shift register which was implemented to delay the valid signals by five CCs until the BLC calculations are done. Both decisions of the fitter can be interpreted as an attempt to reduce the ALM consumption by increasing the use of M20Ks. The clusteriser is the biggest consumer also of those resources. Each of the final four merger used indeed four M20Ks as it should be, to implement a deep FIFO in this place. The width of 160 bit of a cluster requires to use four blocks, each with a width of 40 bit. However, also the FIFOs of the mergers of the other stage are implemented by using M20Ks. Again, the cores are configured so that it is up to the fitting algorithm to select the RAM type. Before going into the final system, one should maybe explicitly set the type here as well to the M20K because the fitter uses them to reduce the ALM consumption, so why not using all the advantages and make the FIFOs deeper also for the intermediate mergers. Having in mind that the maximum width of a M20K is 40 bit, then the consumption of the CPs is also no surprise. The output FIFO was intended to be a shallow one but also here are four M20Ks used by the fitter for each CP. The input FIFOs, in total one per CF, had to be a M20K because of the mixed-width configuration. Since the output port had to have a width of $4 \cdot 14\,\text{bit} = 56\,\text{bit}$, two blocks are needed. This gives in total $275 \cdot 2 + 69 \cdot 4 = 826$ blocks. In summary, there are no surprises and the overall resource consumption is reasonable.

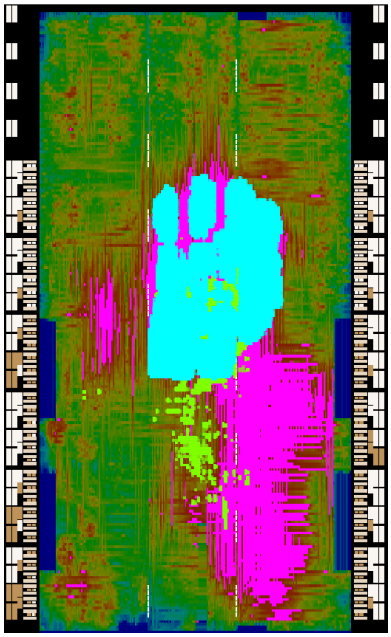
Figure 5.25 shows a visualisation of the used resources of the FPGA by the individual modules, generated with the *chip planer* tool of the Quartus software. As can be seen in figure 5.25a, there is a big region in the centre of the chip with a lot of interconnections, shown in pink. The utilisation of the routing resources is at these locations close to 100 %. The origin of those is shown in figure 5.25b and figure 5.25c. The part on the bottom right is due to the GBT wrappers where the used LEs are indicated in blue which are placed close to the IO-cells (Input/Output-cells). The part of the high routing utilisation in the centre of the FPGA comes from the row-segment merger, shown in cyan. The DMA engine (in red) was placed by the algorithm again close to the IO-cells. The green elements symbolise the pre-sorter logic which is located between the GBT wrapper and the row-segment merger. The complete top part of the FPGA is occupied by the clusteriser, shown in yellow, placed around the merger module.



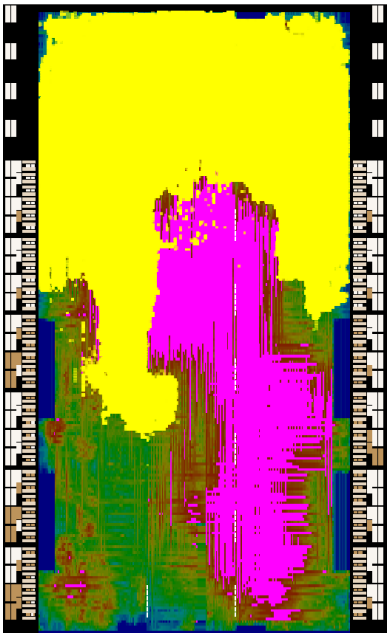
(a) Routing utilisation.



(b) Used LEs by the DMA engine in red and by the GBT wrapper in blue.



(c) Used LEs by the pre-sorter in green and by the row-segment merger in cyan.



(d) Used LEs by the clusteriser in yellow.

Figure 5.25: Visualisation of the utilisation of the FPGA. The different module groups are shown in different colours on top of the general utilisation of the routing resources.

Chapter 6

Performance and Validation

The majority of the validation of the User Logic (UL) must take place in simulation due to the timeline of the TPC upgrade program. To be able to test all modules together in a Common Readout Unit (CRU), the upgraded TPC would be needed: a drift volume which generates realistic clusters, a GEM amplification system and the new Front-End Cards (FECs) generating the GBT frames with the data content, read out by a CRU with the TPC UL included in the Firmware (FW). Further, to fully qualify the processing which was done in the CRU, the ALICE O^2 framework is needed for the decoding of the data stream from the CRU, tracking and Quality Assurance (QA) tasks. But since the TPC will not be ready for such tests before the end of 2019 [32], most modules can only be validated in simulation. The simulation results are discussed in section 6.1. However, there was a test beam campaign in May 2017 to test a preproduction Inner Readout Chamber (IROC) and six of the new FECs at the CERN Proton Synchrotron (PS). This was also used to test the first modules of the UL, the GBT decoding, and verify their correct functionality. The procedure and the limitations are described in section 6.2. Since the Cluster Finder (CF) algorithm was also implemented in software for the O^2 framework, the unavoidable differences between the two implementations and the performance are discussed in section 6.3.

6.1 Performance of the User Logic Modules in Simulation

A validation of the FW with a real system during the development phase was excluded already from the beginning simply due to the time scale of the upgrade program. The development of the UL was started well ahead to have the FW ready as soon as the detector is upgraded. With this it will be possible to read out the detector as soon as the GEM based ROCs are mounted and the electronics is installed. The disadvantage for the FW development is then clearly that one has to rely on a proper simulation. Therefore, great emphasis was put on the simulation of each individual module. An individual test bench is written for each element of the UL, testing all the provided functionalities and each expected source of error. Those test benches are then simulated with ModelSim [60] and the behaviour of the Unit Under Test (UUT) automatically compared to the expected ones, if possible. These simulations are used not only for the development and debugging, but are

also executed automatically without manual intervention after the code in the repository has changed. If even one of the simulations fails, an automatic build of the FW is prevented. This ensures that the provided FW is always fully functional and behaves as expected.

6.1.1 Decoding of the GBT Frames

The decoding of the GBT frames is the basis of all the processing steps afterwards. As already described in subsection 5.2.1, the decoder itself is quite a complex entity. It consists of three identical modules to monitor the ADC sampling clock, must reassemble the Half-Words (HWs) from the SAMPAs, decode the five individual data streams which implies a detection of the synchronisation pattern, and finally provides a defined output stream by merging the result of the five channel decoders. So in addition to the combined simulation, those smaller modules are simulated individually to reduce the complexity.

Clock Monitor

This module must provide two basic functionalities, first the general recognition of the ADC clock in all four possible phases. The module must lock to the rising edge and check that all subsequent patterns are as expected. Second, an error must be reported if the signal on the input port does not follow the correct pattern.

This can easily be simulated by feeding the right sequences into the module, one phase after the other. The module correctly locks to the detected rising edge of the input pattern. After a complete sequence, the *error* flag is taken down, indicating that the ADC clock is now detected as a valid one. During the switching between the phases an *error* is reported because the pattern is now a different one. Afterwards, the module correctly locks to the new one. This behaviour is also compared to a predefined pattern for the automatic test cases. In addition, random bit-errors are introduced which are also correctly found by the clock monitor.

Detection of the Synchronisation Pattern

The basis of the decoding of the channel number is the detection of the synchronisation pattern. Again, the pattern can occur in all four possible phases. So it must always be detected and the correct phase must be reported to the higher-level module. The approach is the same as for the clock monitor, the correct sequences are applied to the module, one phase after the other. The module reliably recognises every pattern and reports the correct phase together with the signal for a completed sequence.

Channel Decoder

The channel decoder combines the detection of the synchronisation pattern with the assembly of the 10 bit ADC values. This assembling needs the phase of the synchronisation pattern to couple the correct HWs together, also involving a buffering of the last received HWs for the case that the data is split across two GBT frames (compare sequences 1 to 3 of figure 5.5.) In addition, the channel numbers must be counted with which an ID is generated.

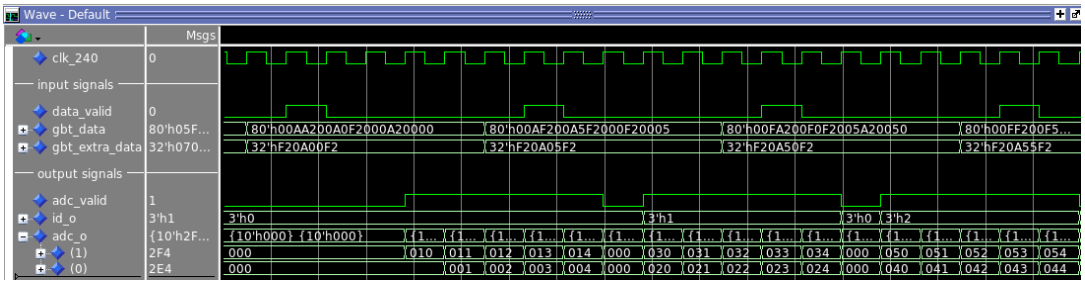


Figure 6.1: Simulation of a complete GBT decoder with ModelSim. The image shows the data interface of the simulated module (cf. with the timing diagram in figure 5.6).

To show the correct functionality, again all four phases are tested after each other. An HW stream is generated, consisting of the synchronisation pattern at the beginning and ADC values afterwards. Since it does not matter which ADC values are decoded, it is very convenient to use a counter for the ADC values for the automatised testing procedure. Each decoded value is then greater by one than the previous one. This allows an automatic verification of the reconstructed ADC value without keeping track of the input data.

The decoder reliably detects the synchronisation pattern and provides the counter values on the two output ports, together with a valid flag and an ID which indicates the channel number. By simply counting how often the valid flag was present, it is verified that the correct ADC value was decoded. Port 0 of the decoder then shows a number, always twice as high as the valid-flag-counter, and port 1 shows a number which is higher by one. Since the ID is in principle also just a counter form zero to seven, it can be verified with the decoded values. Both start at the same time with zero, that is why $\text{ID} = (\text{ADC}/2) \bmod 8$ must always be true and can therefore be used for the verification.

The complete GBT Decoder

After the individual components are successfully simulated, a complete GBT decoder can be validated. The input to a GBT decoder are, apart from the configuration, the GBT frames. They must be generated according to the description in subsection 5.2.1. For the automatic validation, again the same approach is used as for the individual channel decoder, a counter is embedded in the frame, replacing the ADC values. Besides the counter, the four LSB of each ADC value are used to identify the half-SAMPA so that the content of the output interface, presented in table 5.1, can be verified. An excerpt of the simulation with ModelSim is shown in figure 6.1. The same signals are displayed as in the timing diagram in figure 5.6, on top the used 240 MHz clock as a reference, below the input GBT frames consisting of the 80 bit wide data field of the GBT protocol and the 32 bit wide FEC field with the additional data of the wide bus mode. The valid-flag is used to sample the input data. The output signals are shown at the bottom, consisting of a valid-flag, an ID and the two ADC data ports.

As can be seen, the valid signal is active for five out of six Clock Cycles (CCs), exactly as expected. Second, the ID increases by one with each rising edge of the valid signal. The figure contains only the first three cycles since in a further zoomed-out version, the ADC

values would no longer be visible. The ID increases from 3'h0* to 3'h2 in the first three cycles. This is continued in the simulation until 3'h7 is reached and then starts again with 3'h0. The content of the data ports are shown in 10 bit hexadecimal numbers, so the last digit corresponds to the four LSB with the half-SAMPA number (d_i in table 5.1). It can nicely be seen that this number increases with each CC in which the data is valid. The other two digits show the transmitted counter which is the same for all five half-SAMPAs. Port 0 has always the even numbers and port 1 the odd numbers, both increasing with each new GBT frame and therefore with each valid-cycle of the data output. All this is checked in the automatic verification procedure and demonstrates the correct functionality of the GBT decoder.

6.1.2 Sorting Algorithm

The approach which was used for the validation of the GBT decoder needs to be improved in order to be used for the validation of the sorting algorithm, too. The general functionality of the two modules, the pre-sorting together with the merging of the segments, needs to be checked, as well as the configurations of the pre-sorting module. Those configurations, which are 40 times the two read addresses in the correct order together with the control command for the row-breaking, are mostly unique for each of the 182 half-FECs, only 20 randomly distributed configurations are by chance equal (e.g. the configuration for FEC 5 in region 1 is equal to the one for FEC 8 in region 3).

Since there are a lot of cases it would be quite difficult to write all checks to the test bench file without errors. Therefore, a slightly different approach was used. A small GBT frame generator was written as a FEC emulator. It embeds either the channel number or the SAMPA chip number or the FEC ID as the ADC value into the frames, depending on the setting. As a reminder, the SAMPA ID and channel number are different for the different regions as it was shown in table 5.1. Having this in place, together with the already validated GBT decoder, an input stream with a defined sequence is generated.

The Pre-Sorter

To validate the functionality of the pre-sorter and the configurations, two runs are performed for each individual configuration (for simplicity reasons the few duplicate configurations are not handled separately but are just simulated again), one run with the frame generator configured to send the SAMPA channels and a second one with the SAMPA chip number. The FEC ID is not needed because this part of the data preparation path is still done for each input link (and with that for each FEC) individually. The resulting segment of the pad plane is written to a file where it is checked in a second step with a small C-macro using the O^2 framework. As already mentioned, the correct mapping is part of the framework. The written files look like it is shown in figure 6.2, on the left side with the channel numbers at the pad positions and on the right side with the SAMPA chip ID. By confirming that all SAMPA channels from the individual SAMPA chips (therefore the two runs) are written to the correct pad positions, the overall functionality and the configured mappings are validated. Since this approach needs the O^2 framework which was not yet installed on

*Notation for the hexadecimal number 0x7 with 3 bit.

```
18 16 17 19 21  0  0
24 22 20 23 25  0  0
28 26 27 29 31  0  0
 2  0 30  1  3  0  0
 8  6  4  5  7  0  0
12 10  9 11 13  0  0
18 16 14 15 17 19  0
22 20 21 23 25  0  0
28 26 24 27 29  0  0
 2  0 30 31  1  3  0
 8  6  4  5  7  9  0
12 10 11 13 15  0  0
18 16 14 17 19 21  0
24 22 20 23 25 27  0
30 28 26 29 31  0  0
```

(a) The SAMPA channels.

```
2 2 2 2 2 0 0
2 2 2 2 2 0 0
2 2 2 2 2 0 0
3 3 2 3 3 0 0
3 3 3 3 3 0 0
3 3 3 3 3 0 0
3 3 3 3 3 3 0
3 3 3 3 3 0 0
3 3 3 3 3 0 0
4 4 3 3 4 4 0
4 4 4 4 4 4 0
4 4 4 4 4 0 0
4 4 4 4 4 4 0
4 4 4 4 4 4 0
4 4 4 4 4 0 0
```

(b) The SAMPA chips.

Figure 6.2: The content of the pre-sorter mapping files for FEC 0 in region 1. On the left side with the channel numbers of the SAMPA chips and on the right side with the SAMPA chip ID. The row numbers are indicated by the line numbers on the left sides, counting from the top down to the bottom (reversed order compared to e.g. figure 5.7). Each row segment has seven elements, filled by default with a 0 if no other value is set. With this information, each channel of a half-FEC is uniquely identified.

the build server on which the automatic testing procedures are executed, some manual interventions are needed. However, all necessary scripts and commands are included in the repository and can easily be run on a properly prepared machine.

The Row-Segment Merger

For the confirmation of the merger module, the same concept is used. However, the simulation must be extended substantially, since the merger is the first module (apart from the Common Mode (CM) calculation) which combines the data from all input links. Therefore, the simulation must contain twenty frame generators (this time with a different FEC ID setting for each instance), one for each input link. Afterwards twenty GBT decoders are needed and also twenty of the pre-sorter modules. This quite extensive setup is shown in figure 6.3. Up to this point, everything is already validated and can therefore be used. Since there is only one merger module in each CRU, the mapping must be validated for each of the ten regions separately. Within each region, the frame generators must be configured with the right FEC ID and the single pre-sorter instances need to be configured individually for the respective location. Having this, the simulation must run three times, one for each setting of the frame generator to be able to write the corresponding pad planes of the individual regions with the FEC ID, with the SAMPA chip and the SAMPA channel to a file. With those three informations the pad position is unambiguously determined and with that is the overall mapping procedure validated. This is again done with an external C-macro using the mapping information from the O^2 framework.

The reason why one has to run the simulation multiple times is because those three numbers do not fit into a single 10 bit value. The FEC ID is a number between 0 and 19

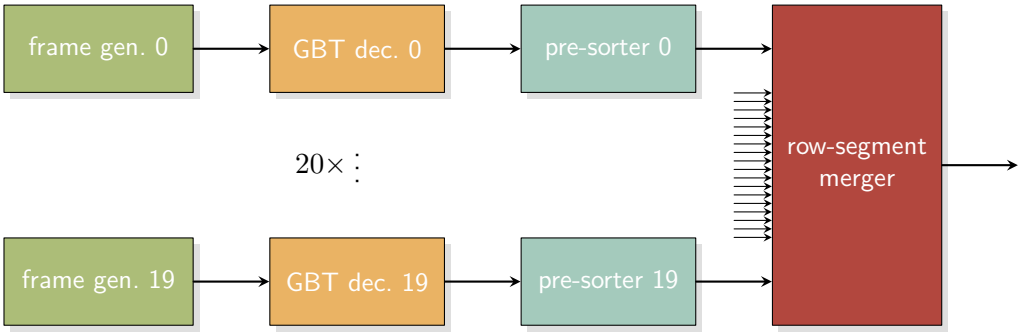


Figure 6.3: Setup of the row-segment merger simulation test bench. To fully qualify the merger, data for all the twenty input ports must be generated. To do this in a convenient manner, twenty frame generators are instantiated, together with twenty GBT decoders and twenty pre-sorters. The whole test bench consists of 61 modules.

(5 bit), the SAMPA chip a number between 0 and 4 (3 bit) and the SAMPA channel a number between 0 and 31 (5 bit). One could argue that at least the chip and channel numbers would fit into a single 10 bit value, which would reduce the required number of simulation runs for the pre-sorter. However, due to the additional FEC ID information needed to validate the merger module, the simulation of the merger module must be done multiple times. Having this in mind, it is not worthwhile to make the simulation more complex than absolutely necessary, always with the danger of remaining undiscovered errors, just to make it a bit more convenient.

6.1.3 Baseline Correction

The validation of the BLC module is straight forward. An input stream of randomly chosen values is generated on which all combinations of the three operations (additive pedestal subtraction and CM correction and the multiplicative gain correction) are applied. The individual corrections are enabled and disabled after each other so that all eight combinations are tested, as can be seen in the *config* field of figure 6.4. The result of the module is then compared to values which are calculated directly with the corresponding operators in VHDL. With that, the module is validated. This is also part of the automatic verification procedure.

6.1.4 The individual Cluster Reconstruction Modules

Since the CFs and the Cluster Processors (CPs) are the heart of the UL, great attention was given to the simulation of these modules. To be sure that the reconstructed clusters really correspond to the data, the testing procedure consists of several steps. First, a single CF instance is validated, it is checked that a single CP is calculating the cluster properties accurately and that the FIFO merger is able to combine the input streams to a single output stream. In a second step, the CF and CP modules are combined with increasing complexity. It is startet with one module of each to ensure that the protocol between them is working. Then the output of two CFs is processed by one CP which is then increased to five CF

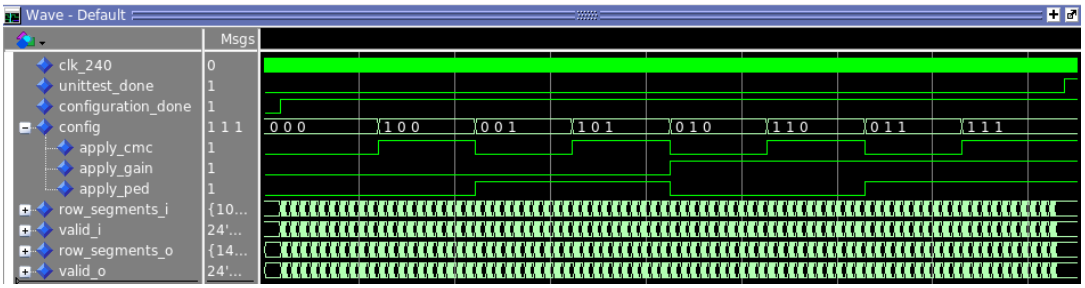


Figure 6.4: Simulation of the BLC module with ModelSim. Each combination of the three corrections is applied one after the other, verifying that all calculations are done properly.

instances, one more than in the final system to stress-test the CP. The next logical step is to increase the number of CPs to two (and with that the number of CFs to ten) which makes it possible to place a FIFO merger at the end. This setup is then further extended to 23 CF instances, which corresponds to a complete pad row with 138 pads, five CPs and a single FIFO merger. As a last step, the full clusteriser network is simulated, consisting of all the 275 CFs, 69 CPs and the two merging stages in the end, for all ten region configurations.

The Cluster Finder

The pattern which is fed to the CF instance contains several peaks to cover all possible cases. There are peaks on both sides outside of the central region to check that those peaks are ignored and not detected by this CF, they will be found by the neighbouring one. There are isolated clusters and clusters which are nearby to validate that those are correctly found as well. Also, a peak with a low ADC value is added to the test data to check that the corresponding threshold is taken into account to suppress small peaks. The contribution threshold is tested as well by introducing some small numbers next to the central peak in all directions. An additional feature which helps to verify the functionality is that the data sample has an odd number of time bins. By using this sample twice, shifted by one cycle due to the odd number of entries, it can be checked whether both of the two time bins that are processed together in the CF are treated equally.

Also, the output interface is validated since it is used to check if each individual bin of the found clusters is provided in the correct order by the CF. Finally, the content of the header fields (see table 5.5) are controlled as well. The configured row number must be transmitted in the field H0. The correct pad number, including an offset which is calculated with the configured CF number, must be contained in H1. The last header field H2 must contain the correct time bin of the peak, taking also several Heart Beat (HB) reset triggers into account. Since the additional flags are not yet used, it is ensured that they remain at zero. In summary, every detail of the CF is checked during the automatic validation procedure.

The Cluster Processor

The validation of the CP must cover two aspects, firstly the correct calculation of the cluster properties and secondly the correct handling of the input buses. The latter includes the

selection of the next, non-empty input after the current processing has been finished and the recognition of the reset pattern with subsequent masking of the input bus.

The test for the first case is straight forward. Some predefined cluster patterns are written to the input FIFOs of the CP and the result is then compared to pre-calculated expected values. The row number which is part of the cluster pattern but not involved in the actual calculations (it is just passed through the CP) is used in the end to identify the cluster. This set of clusters is then just reused during the whole simulation. It is not an issue that the number of tested clusters is strongly limited because the actual calculations take place in the Digital Signal Processor (DSP) blocks of the FPGA, which are supposed to work. If this assumption would be invalid, then this FPGA would not be suited for our purpose. But the processed clusters are still important to validate the bit-shift operations for the δx_i multiplications (see e.g. figure 5.23a) and the correct accumulation.

To be as close as possible to the expected environment, the input FIFOs are not filled evenly, but randomly with different priorities. This requires an internal mechanism to prefer a FIFO which is almost full for the next reading cycle. With this it is ensured that a single FIFO does not run full while the others are almost empty. In addition, several HB reset cluster patterns are sent in-between to test the correct forwarding of the reset and the generation of the reset cluster. All of this is part of the automatic testing.

The FIFO Merger

The merger is a rather simple module. The automatic validation is done by feeding a predefined pattern to all input ports. The output port is then checked for this pattern. In addition, also the reset cluster detection and forwarding mechanism is tested.

6.1.5 The complete Clusteriser

Before the entire clusteriser is simulated, small subsets are built for a better overview. A system with hundreds of modules involved, thousands of interconnections and even more internal paths is quite challenging to debug. Therefore, smaller systems with less CFs and CPs are built first for a basic validation. The smallest system consists of one CF connected to just one CP. This system is then enlarged in steps to two CFs and five CFs, each with one CP to include the merging capabilities of the CP in the testing procedure. The next step is then to increase the number of CPs to two and finally to five — and with that the number of CFs to 10 and 23, respectively — to cover a full pad row in the biggest system. The latter two need also a FIFO merger in the end for a synchronous readout of the CPs.

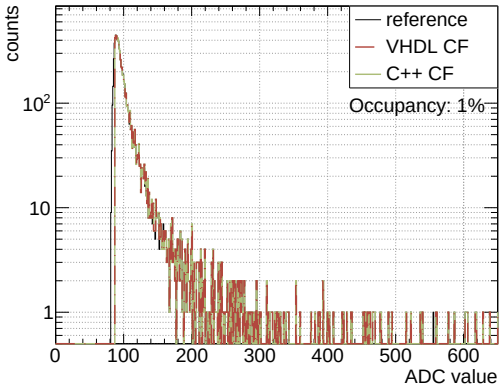
Starting with smaller Systems

So far, only a basic validation of the individual modules is done: it was shown that the CF is able to find all peaks in the central region and writes out the correct bins. It was also shown that the CP calculates the properties of a cluster correctly. But more data is needed to fully verify both modules. Therefore, test datasets with random clusters are generated. These clusters are 2-dimensional Gaussian distributions with uniformly distributed centres in pad and time direction within the valid regions of the individual systems. To have cluster shapes similar to the ones expected in the real system, equation 5.6 and equation 5.7 are

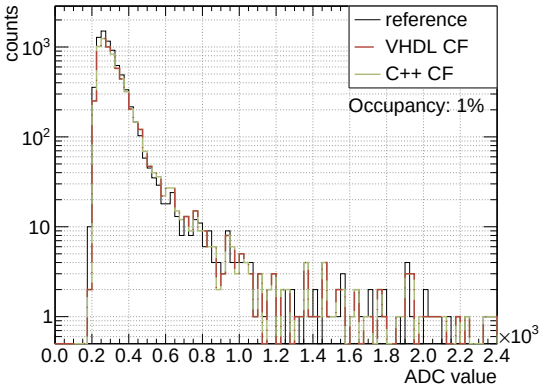
used to calculate the widths in both directions. A random radius within the coverage of the TPC is chosen which is used to select the correct pad sizes and to calculate the inclination angle for σ_t. The additionally needed drift length is randomly chosen as well between 30 and 250 cm. The value for σ_p is expressed in units of pad widths and σ_t in units of time bins with a length of 200 ns, corresponding to the sampling frequency of 5 MHz of the TPC. The normalisation of the Gaussian function, which corresponds to the $q_{\max}$ value, is randomly drawn from a Landau distribution. This parameterised function is sampled a thousand times to fill a histogram to emulate the binning effect of the detector. This histogram is then scaled to the desired $q_{\max}$ value and added to the dataset. For sufficient statistics, 100 000 time bins are filled with six different *occupancy* levels of 1, 5, 10, 20, 30 and 40 %. The occupancy is defined as the ratio of bins with an ADC value above zero divided by the total number of bins of the generated data set. The 100 000 time bins correspond to $\lceil 100\,000/447 \rceil = 224$ HB frames. In this way, also the synchronisation is extensively tested within the simulation. Such datasets are generated individually for each of the clusteriser systems, as the pad area covered is different and ranges from 6 pads for the smallest system to 138 pads for the largest one. They contain between ∼300 clusters in the dataset for one CF and an occupancy of 1 % and ∼300k clusters in the dataset for 23 CFs and an occupancy of 40 %. In total, summing over all systems and occupancies, more than 1.7M clusters were generated of which more than 1.38M clusters could be found and were reconstructed. This difference between the number of generated and found clusters is not an issue because its origin are the overlapping clusters. There are 1.7M clusters generated with a random placement, so it is possible and for the high occupancy case rather likely that they overlap. If then the peak of two clusters is placed at the same pad–time coordinate, or next to each other, only one can be found instead of the two generated ones.

The validation is done in two ways. First the resulting distributions of the cluster properties $q_{\max}$, q_{tot}, pad and time information, as well as the corresponding widths are compared to the ones of the generated clusters. Second, these distributions are compared to the results of a software implementation of the CF, which will be discussed in section 6.3. The same input files are also used for the software version. Those comparisons can be seen in figure 6.5 for the system with 23 CFs and an occupancy of 1 % and in figure 6.6 for an occupancy of 30 %. In both occupancy cases, a cut on the $q_{\max}$ value of the cluster of ADC > 86 and on the contributing bins of ADC > 5 is applied to test this mechanism as well. The 23 CF case was chosen because the statistics is the highest compared to the other systems, simply due to the higher pad coverage.

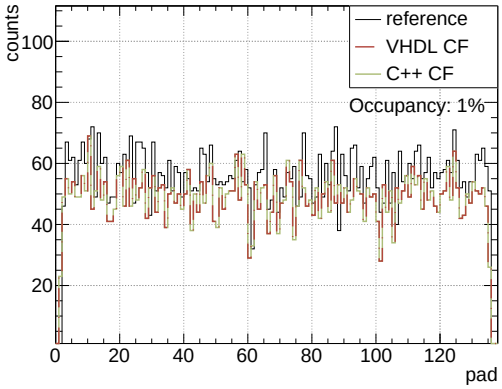
The black line is always the reference distribution of the randomly generated parameters. The distributions obtained in the VHDL simulation is shown in red and the green lines are the corresponding ones from the software implementation of the algorithm. Those two are not only very similar, they are indeed identical. For each individual cluster it was checked that the one found in software is bit-accurate with the one obtained from the VHDL simulation. It is noteworthy that the algorithm, written in two completely different languages — one in VHDL and the other in modern C++ — which are executed in two completely different ways (with ModelSim and with the O^2 framework), without a single deviating bit leads to exactly the same result. This means that for further studies of the CF performance, one can rely on the software implementation which is running substantially faster. For a comparison, those ModelSim simulations run between 30 min for the smallest



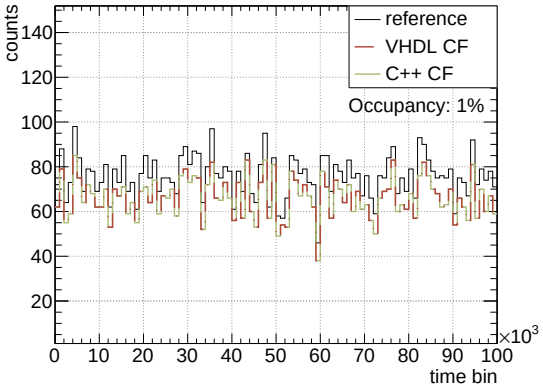
(a) Distribution of $q_{\max}$ of the clusters.



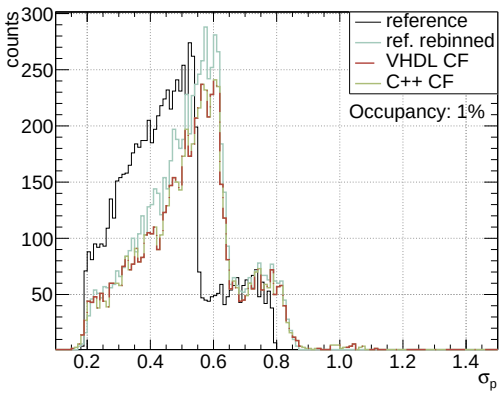
(b) Distribution of q_{tot} of the clusters.



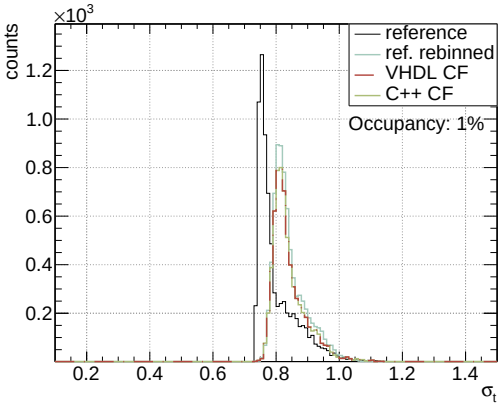
(c) Distribution of the cluster centres in pad direction.



(d) Distribution of the cluster centres in time direction.

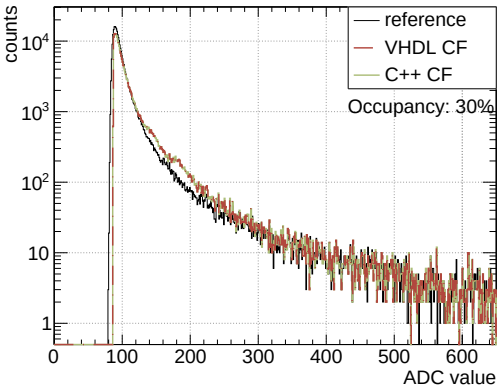


(e) Width of the clusters in pad direction.

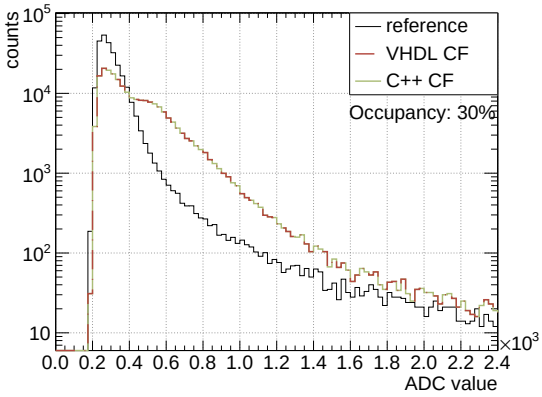


(f) Width of the clusters in time direction.

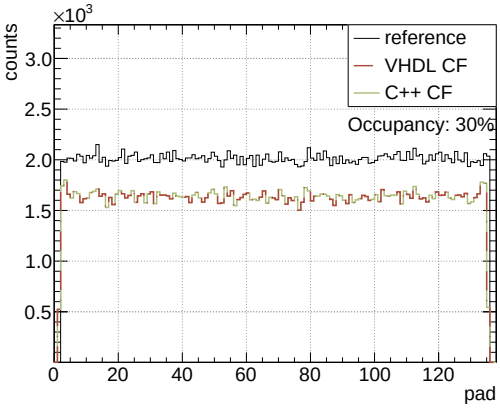
Figure 6.5: Distributions of the cluster properties for the simulation system with 23 CF instances, an occupancy of 1 % and a cut on $q_{\max}$ of ADC > 86 and the contributions threshold of ADC > 5. The reference of the data input is shown, as well as the result of the VHDL implementation in red and the software implementation in green.



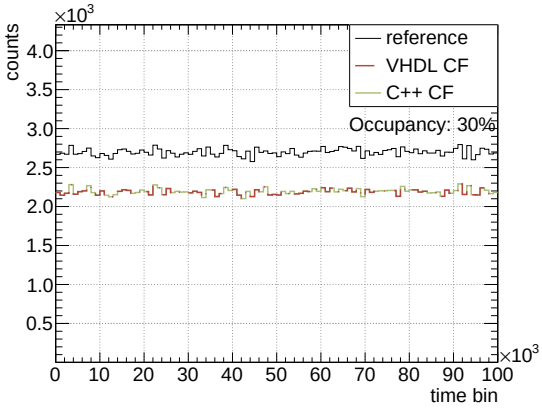
(a) Distribution of $q_{\max}$ of the clusters.



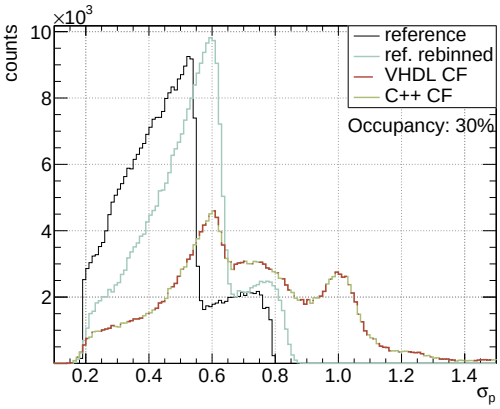
(b) Distribution of q_{tot} of the clusters.



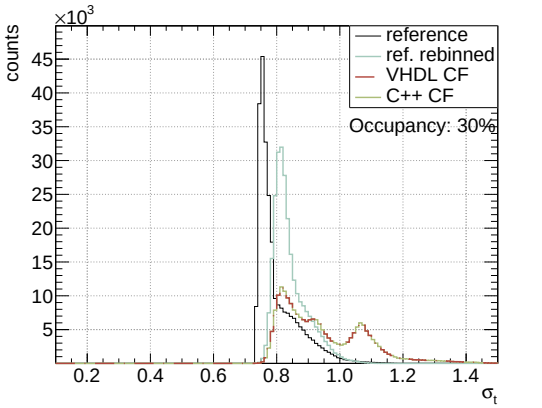
(c) Distribution of the cluster centres in pad direction.



(d) Distribution of the cluster centres in time direction.



(e) Width of the clusters in pad direction.



(f) Width of the clusters in time direction.

Figure 6.6: Same as in figure 6.5 but with an occupancy of 30 % instead of 1 %.

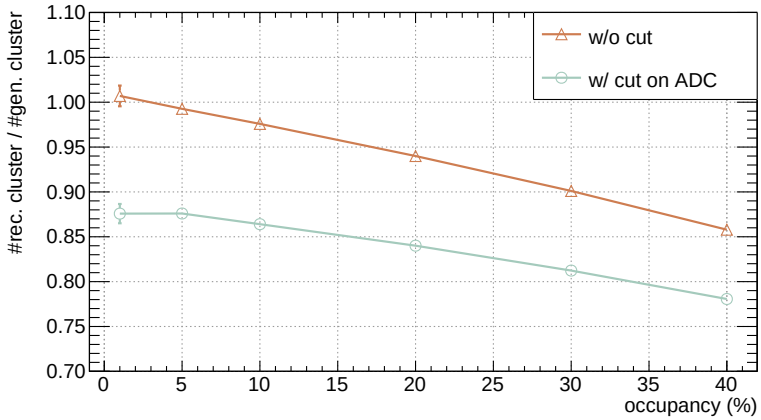


Figure 6.7: The efficiency of the clusteriser as a function of the occupancy. The efficiency is the number of reconstructed clusters divided by the number of generated clusters. The orange curve shows the efficiency without the cut on the ADC value, so the reduction is purely due to the overlapping of two cluster peaks, while the blue one includes a cut on the ADC value of the peak in addition.

system and 2 h for the system with 23 CFs to process the 100 000 time bins of the dataset, while the software version is done within a few seconds.

In general, all distributions of the CF results correspond very well to those of the references. In the two plots with the ADC value, the CF results follows very closely the reference distribution, at least for the low occupancy case. For the high occupancy one, an excess at higher ADC values is clearly visible. This is even more pronounced in the q_{tot} distribution. However, this is an expected effect since at higher occupancies the probability of overlapping clusters is also higher. This will lead to a higher q_{max} value because the tail of a neighbouring cluster contributes also to the maximum value. Since in the current implementation the q_{tot} value is calculated by summing always over all the 25 bins of the cluster (except for the exclusion via the contribution threshold), it is expected that the effect is more visible in this distribution. In the plot of the q_{max} value, the cut at a low ADC value of 86 is clearly visible, which was enabled to validate also this mechanism. Since small clusters are rejected, this contributes also the the reduced number of entries at low ADC values in the plot of the q_{tot} values.

The distributions of the cluster centre in pad and time direction in figures (c) and (d) of figure 6.5 and figure 6.6 are nicely flat as they are supposed to be. The reduction in entries, comparing the red and green lines with the black one, is due to two effects. The first one strongly depends on the occupancy and is simply the overlapping of two or more clusters. When two generated peaks end up in the same or a neighbouring bin of the pad–time plane, then only one cluster can be found although two were generated. With an increasing occupancy, the probability of such a positioning increases as well. The reduction in efficiency due to this effect is shown by the orange curve in figure 6.7. The efficiency is defined as the number of reconstructed clusters divided by the number of generated ones. The number of clusters correspond to the entries of the q_{max} distributions of the respective datasets. It can be seen that for a very low occupancy the efficiency is close to

one, meaning that all generated clusters are also found. The second effect is the additional cut on the $q_{\max}$ value to reject small clusters. By including this cut, the blue curve is obtained which is overall lower than the orange one, due to the additional loss of clusters.

The last set of plots in figure 6.5 and figure 6.6 shows the σ_x distributions in pad and time direction. In the low occupancy case (figure 6.5e and figure 6.5f), the shape is quite well reproduced but a shift towards higher values is visible. This shift can be explained by the binning effect of the data generation. The same effect was observed by calculating the standard deviation of the weighted mean directly from the individual cluster histograms after the binning of the original cluster function, for both pad and time directions. The resulting distributions are shown in blue, which display exactly this shift. So this effect is expected. Comparing the blue curve to the red and green ones, only the reduction in entries is visible, the shape is well reproduced. For the high occupancy case in figure 6.6, there are many more entries for higher σ values, but also this is expected. Again, in a high occupancy environment, there are many clusters close to others and they do even overlap. Since there is no way to perfectly separate the corresponding distributions from each other in the CRU, the resolution must get worse, which is the visible effect.

To summarise, the clusteriser is behaving as expected. The individual distributions are reconstructed very well and no border effects due to the segmentation of the pad row is visible. Also, the calculation of the cluster properties works as expected. Since an automatic classification of those distributions is not straight forward, some manual intervention is still needed to qualify the result. This, together with the very long simulation time prevents the inclusion of this validation into the automatic testing procedure and must be done manually.

Validation of the whole Clusteriser Network

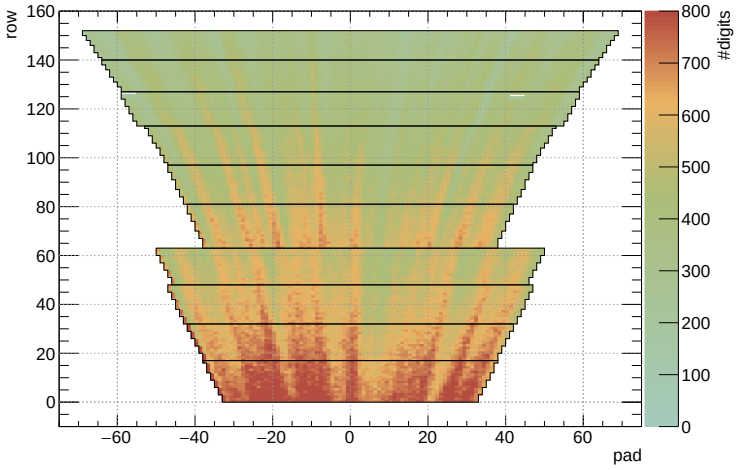
After the basic concept of the individual CF instances is validated and the processing and merging by the CPs and FIFO mergers proven to be working, the full clusteriser with all its 275 CF, the 69 CP and the two merging stages in the end can be simulated. The main purpose of this additional step is to validate the optimisation of the CF grid as it was discussed in subsection 5.3.6. It must be proven that the reusing of the individual CF instances for different positions in different regions does not introduce errors in the mapping. Also, the performance of the merging network needs to be tested to show that at reasonable and worst case data rates, no back pressure is generated on the CF instances. For a proper simulation of the latter one, the clusters need to be correlated. In the real detector, most of the clusters originate from a track, crossing several (in the ideal case all) pad rows. So within a short time interval, there are many clusters in different rows which arrive in the merging network closely in time, leading to spikes in the required bandwidth. Since real data, taken with the detector, is unavailable at the moment because the upgraded TPC does not yet exists, the O^2 framework is used to generate simulated input data for the validation procedure. In this framework, the detector response is simulated. The charges, generated by tracks within the active area of the detector, are transported to the pad plane where they are digitised. This includes the simulation of the amplification of the GEM stack and the electronics response. This software simulation is tuned that the signals are as close as possible to the ones expected from the real detector. The so called *digits* can then be extracted from

the simulation and used to feed the VHDL simulation. In addition, those digits are further processed within the O^2 framework to find the clusters and compute the properties also in software. This is then used for a comparison with the VHDL clusteriser network response.

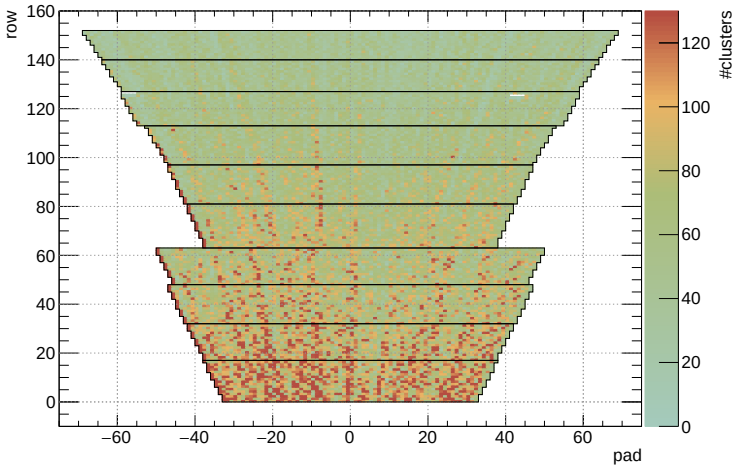
In total, around 12k time bins were generated in which 0.97M clusters were found. To reduce the computational effort, only one sector of the TPC was taken into account, which is the minimum that must be simulated to cover all ten different regions. The regions are simulated separately with ModelSim to test all the ten different configurations of the modules. The test bench instantiates the clusteriser module, takes care of the individual configurations and provides the input data in the correct format. After the simulation is done, the clusters are compared with the result of the software CF. Again, in almost a million clusters, not a single deviating bit was found. Exactly the same clusters were found and exactly the same properties were calculated in both, the software version and the VHDL implementation for the CRU. Even with rather challenging simulation parameters. The occupancy distribution of the dataset is shown in figure 6.8c. The number of occupied pads, meaning the pads with signal above zero, was counted for each row individually in each time bin. Then by dividing by the number of pads in the specific row, the occupancy was calculated and filled into the histogram. As can be seen, the occupancy reaches up to 60 % although a maximum occupancy of only 30 % is expected for the TPC after the upgrade. So this simulation can be considered as a kind of worst case environment. Figure 6.8b shows the number of clusters, found in each row–pad combination, integrated over all simulated time bins. The individual regions are marked with a black box. It must be noted that the pad width changes between the regions. This becomes particularly visible when going from region 3 to 4 where the transition from IROC to Outer Readout Chamber (OROC) takes place. A row then covers a larger area with fewer pads because the pad width is larger. The pad numbers are centred around zero, which means that the pad located in the middle of each row is assigned the pad number zero for a better symmetry visibility. The main message from this plot is that each pad of each region was hit at least once during the simulation. This means that, since all clusters from the ModelSim simulation could uniquely be matched to a cluster in the software simulation — where this reusing of CF instances is not needed and therefore not implemented — and vice versa, no error in the mapping is introduced and therefore the clusteriser is completely validated.

There are just two additional details to be mentioned. One is the high number of clusters on the leftmost pad of each row in the inner regions. There is indeed a huge excess in number of clusters in this leftmost pad. Compared to the other ones almost twice as many clusters are found in the worst case. As can be seen in figure 6.8a, which shows the same plot but for the digits used as input for the two clusterisers, there is an excess in the same pads, too. The other detail to be mentioned is the existence of the two white spots in row 125 and 126. There are indeed four consecutive empty pads in both rows. Here again, the same white spots are visible at exactly the same positions in the input data. The observation was communicated to the corresponding developers, who could not find out the reason in time for the submission of this thesis. However, since the same spots are reproduced by the clusteriser, the confidence in the correct functionality of the CF and CP is even greater.

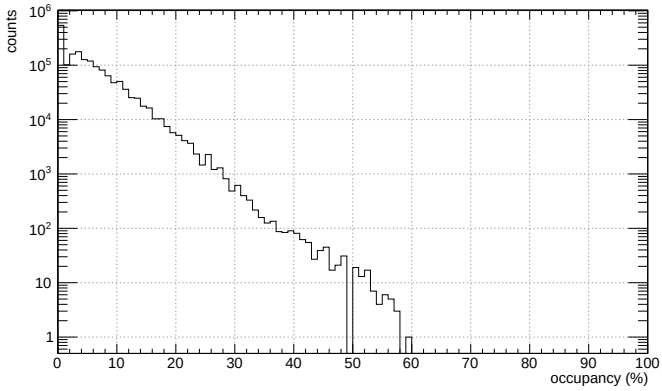
The last item to be checked is the behaviour of the FIFO mergers under this condition. The filling level of the four final FIFOs of region six (the innermost one with 1 600 pads) is



(a) Number of input digits per pad and row.



(b) Number of found clusters per pad and row.



(c) Simulated occupancy distribution of the individual rows.

Figure 6.8: Input and results of the full clusteriser simulation.

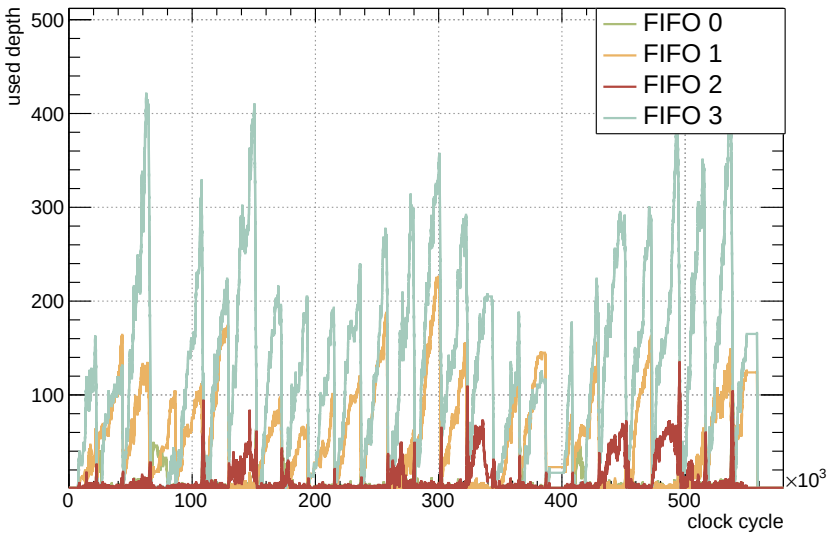


Figure 6.9: Filling level of the four final FIFO mergers as a function of time during the simulation. The FIFOs have a depth of 512 entries.

shown in figure 6.9. It can nicely be seen how all four FIFO levels build up whenever there was a new wave of tracks. Afterwards, they are emptied and filled again. Having a FIFO in such a position raises always the question about the possibility of a buffer overflow. If those final FIFOs would run full, there are more (but smaller) FIFOs in the merging stage 1, which can also buffer part of the load. If those run full as well, there are still the output FIFOs of the CPs and, in addition, the output FIFOs of the CFs. So there is a cascade of FIFOs which are able to buffer a significant amount of clusters even in the worst case scenario. Only in the very rare case when all FIFOs are full, clusters are thrown away in a controlled manner to not disturb the processing chain.

This validation step must also be carried out by hand. First of all, the evaluation of the results is not easily automatised, and second, the simulation time is just too long. Each of the ten regions need up to 10 h to simulate the processing of the 12k time bins (with the software implementation, this is done for all regions combined within a few seconds). So if this would be done after each commit, the build server would be busy the whole time with this simulation and would not be available for its real purpose which is the compilation of the FW.

6.2 Validation during Test Beam Data Taking

The decoding modules of the UL could be tested in a real system during the time of this thesis. There was a test beam campaign in May 2017 at the CERN PS to test one preproduction IROC and the new FECs. The ideal case would be to also use a CRU together with all the O^2 machinery, meaning a First Level Processor (FLP) and the O^2 framework for the reconstruction and processing, to read out the system. Then, the overall test would be as close as possible to the final setting of the experiment.

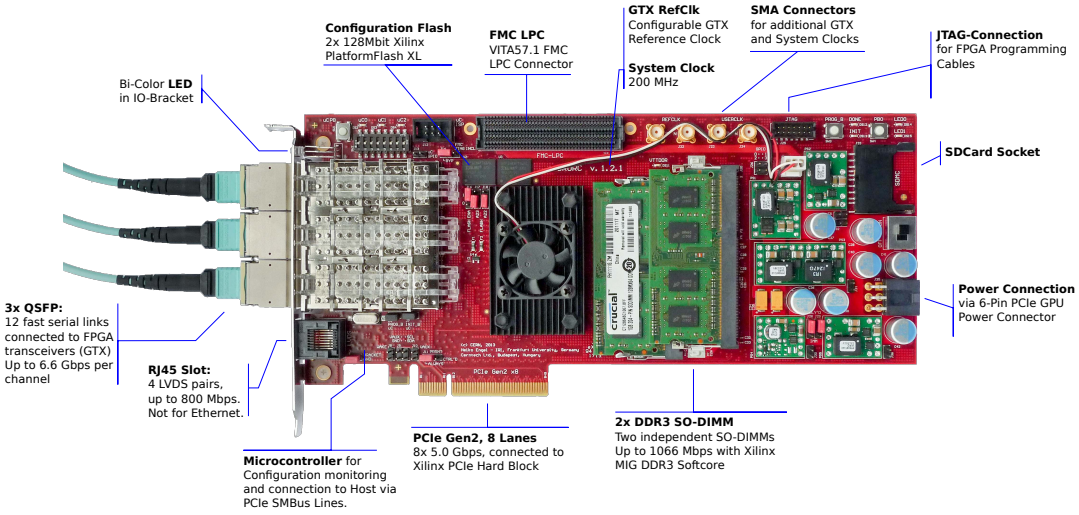


Figure 6.10: Image of the C-RORC with its major components, taken from [61].

But there are some arguments speaking against it. First of all, the main purpose of the campaign was to test the ROC and not the subsequent readout chain. The FECs are mandatory to be able to read out the chamber, but adding there another prototype hardware — the CRU — would introduce another source of error and could lead to an overall failure of the operation. Especially because the CRU hardware was at that time still far from the final version and just a prototype. Even more, the FW was also not ready in time, not the common parts and only a few modules of the TPC specific UL. Therefore, another solution had to be developed which was found by reusing the Common Read-Out Receiver Card (C-RORC), which is an already well established FPGA based readout card, currently in use in the Data Acquisition (DAQ) and the High Level Trigger (HLT) systems of ALICE. This card has some disadvantages compared to the CRU, especially concerning the available bandwidth and number of possible links, but has the huge advantage that the hardware is proven to be working.

6.2.1 The C-RORC as Readout Card

Since the CRU was not available, the readout system for the test beam was built with a C-RORC. This is a PCIe card with a Xilinx Virtex-6 FPGA as the central component. The board offers a PCIe generation 2 connection to the host machine with a bandwidth of up to 3.7 GB/s. The three Quad Small Form-factor Pluggable (QSFP) modules each provide four high speed optical links with up to 6.6 Gbit/s per channel [61]. A photo of the board is shown in figure 6.10. In collaboration with the original developer of this card, a FW for the test beam was built, reusing the existing Direct Memory Access (DMA) engine and the available software tools. The FPGA has even enough resources that twelve GBTx cores could be implemented, which opens the possibility to make use of all available optical connections. With that, six FECs can be controlled and read out by a single C-RORC. Unfortunately, it is not possible to add more than one C-RORC to the readout system (and with that not more

than six FECs). All the FECs need to receive a synchronous clock which must be provided by the C-RORC. Having here more than one in the system would require an external clock source synchronising the readout cards. But the transceivers of the FPGA can only use internally generated clocks and not externaly provided ones. This is why multiple C-RORCs would not be synchronous.

The input data rate of all twelve links sums up to $12 \times 112\,\text{bit} \times 40\,\text{MHz} = 53.760\,\text{Gbit/s}$, which is already almost twice as high as the available PCIe bandwidth. Since no Zero Suppression (ZS) can be applied to the data (again because of the CM effect), a continuous readout is not possible and a triggered one must be implemented in which the data is transferred only when particles crossed the chamber. A triggered readout is anyhow better suited for the environment of the PS test beam since the interaction rate is rather low with only a few hundred Hz.

6.2.2 The Triggered Readout Mode

The future FEC of the TPC is designed for a continuous readout, not a triggered one. Therefore, the FEC will send a continuous data stream also in this case and the triggering must be applied later in the C-RORC. Here, a readout gate is opened upon arrival of the trigger signal, allowing to send a configurable number of GBT frames to the host machine. A header is prepended to each chunk of data, containing a time stamp. With this time stamp and the synchronisation pattern, which must be contained in the very first chunk of data, the GBT frames can be decoded in software even though big portions of the continuous stream are missing. By correlating the time stamp of the first trigger sample with the frame number in which the synchronisation pattern is found, it can be calculated in which subsequent frame numbers the first channel of the SAMPA readout is located. With this it is possible to decode the data also in the triggered readout mode.

6.2.3 Validation of the GBT Decoder

Three different readout modes were implemented for the test beam. In the first one, only raw GBT frames are read out. The second one provides the already decoded ADC values, together with the ID generated by the GBT decoder. In the third mode, the two modes are combined and raw frames together with the decoded values are read out. The last one is used for the validation of the GBT decoder module. During the test beam campaign, eight runs were taken in this mode with more than 120k events. This corresponds to more than 4.4 billion individual ADC values which could be compared. The spectrum of those values is shown in figure 6.11. The green line shows the ADC values which were decoded in software afterwards, while the red one corresponds to the ADC values already decoded in the C-RORC. The two histograms are identical, which can also be seen from the ratio below, that is equal to unity over the entire range of ADC values from 0 to 1023. There are some entries at zero and an increase at an ADC count of $\sim$500 which belongs to dead channels of the SAMPA chips. The origin of the other visible structures are also understood by non-uniformities of the SAMPA ADCs. This leads to the fact that even and odd numbers did not occur with the same probability. This issue was fixed in a later version of the

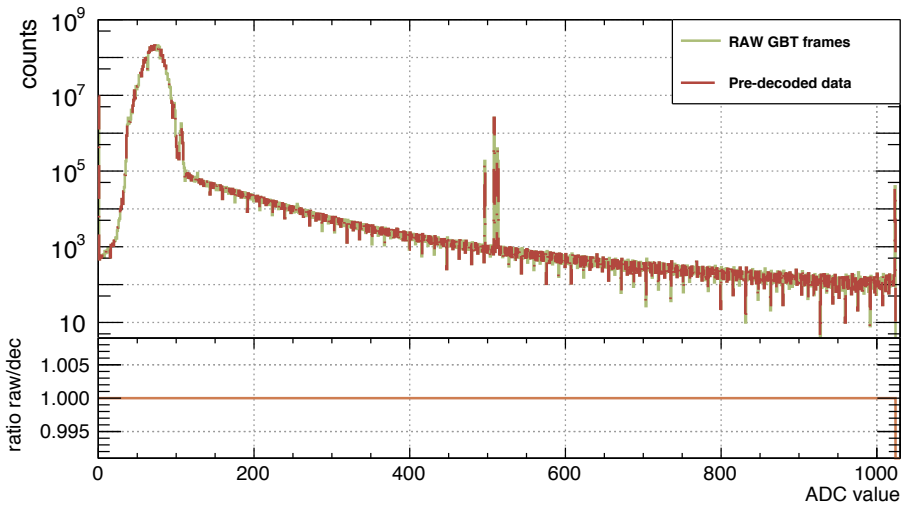


Figure 6.11: Spectrum of the ADC values recorded at the test beam. The green line shows the values from the GBT frames which were decoded in software and the red ones which were decoded already in the C-RORC. The orange line below shows the ratio of the two histograms, which is equal to unity over the entire range.

SAMPA chip. The equality of the two histograms proof again that the GBT decoder is working, also in a real FPGA.

6.3 Cluster Reconstruction in Software

The same cluster reconstruction algorithm, or at least one which is as similar as possible, was also implemented in modern C++ within the ALICE O^2 (Online-Offline) software framework. In this framework, all functionalities needed for the ALICE experiment are combined. This includes software for the readout of the detectors, the event building, the recording of the data, calibration tasks, reconstruction of the data as well as for the physics simulation and analysis [47]. Since the purpose of the simulation is to generate data that is as similar as possible to the real recorded ones, also the same reconstruction steps have to be applied. For the TPC, the cluster reconstruction is done already in the CRU during the normal data taking mode. So there must be a software version of the clusteriser available for the processing of the simulated data which provides exactly the same results. Therefore, also the same two-step approach was implemented, first going through the data and findig the peaks according to the definition in figure 5.15 and in a second step calculating the same properties as written in equation 5.26. In order to guarantee that the calculations deliver the same results, the Fixed-Point (FP) arithmetic is also used in software.

Though, there are two basic differences between the C++ version and the one written in VHDL due to the slightly different applications and due to the characteristics of the respective languages. First, the separation into small CF instances is not implemented in software. This was introduced for the FPGA version to reduce the fan-out of the individual paths. Because of the overlap needed in between in individual instances, this leads to a higher computing effort which can not be justified for the software version. Here, besides

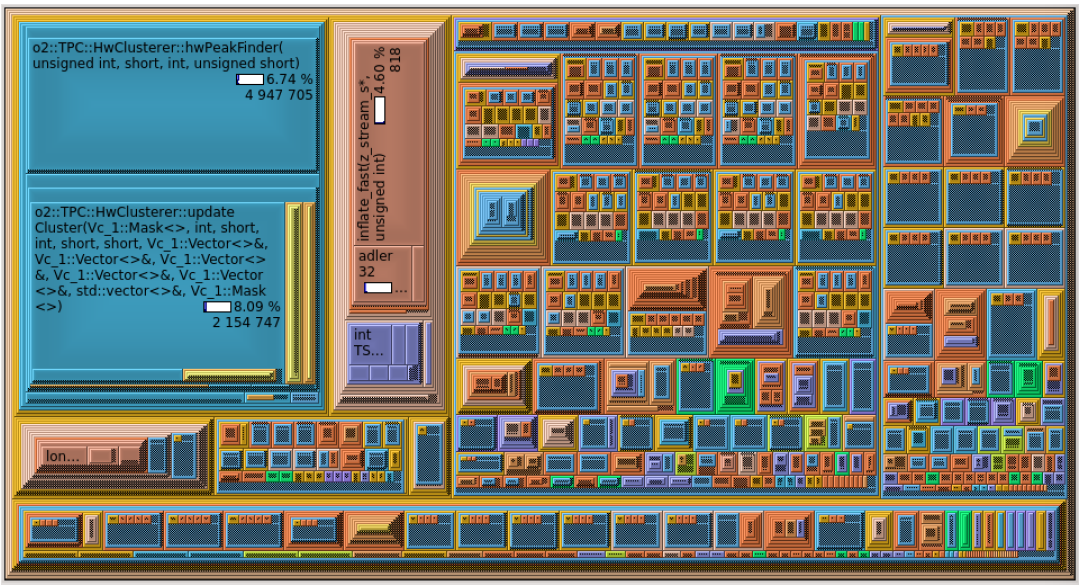


Figure 6.12: The Callgrind [62] output of the software CF, visualised with the tool KCachegrind [63]. The blue rectangles show the share of computing time which was used by the CF. As can be seen, although they are the largest contiguous blocks, they use only a small fraction of about 20 % of the overall time.

the equality of the result, also the needed computing time is an important factor. So there are no borders within a row introduced in contrast to the VHDL version. Second, due to the implemented parallelism scheme of the O^2 framework, each sector of the TPC is treated by a different computing process. So it does make sense to handle a whole sector, meaning all the 152 rows of the ten regions combined, by one clusteriser. This reduces the overhead of the handling of the data and makes the execution more straight forward. Since there are no individual small CF instances which are even reused at other locations in different regions as it is done in the VHDL version, the mapping is much easier to implement and to debug.

The overall correctness of the implementation is crosschecked in subsection 6.1.4 and 6.1.5 with the results of the VHDL implementation. The assumption is that if both implementations give the same result, although they are written in completely different languages and run in completely different ways, then the algorithm must have been implemented in a correct way in both cases. In addition, there are small test cases written with known clusters where the output is automatically checked. Also, a few checks were done with a tracking algorithm working with the clusters from the software implementation. However, extensive tests which proof the physics performance of the clusteriser are not yet completely done. They are planed for the future when the framework is evolved far enough that this can be done in a comprehensive way. But the algorithm itself is well justified and, as can be seen in the corresponding sections, implemented in a correct way.

Since the software clusteriser is used besides the validation of the VHDL modules mainly for the reconstruction of the simulation data, the performance is quite an important factor, together with the memory footprint. To reduce the memory footprint, the locally buffered

data which is needed for the cluster finder is reduced to a minimum. Similar to the VHDL version, only six time bins are needed at the same time in the local buffer, five which contain the full cluster and a sixth one so that the comparison operation of the individual pads for the peak finding is finished also for the last of the five time bins. The eight needed comparisons for each pad are done not at the same time: four are done when the *current* pad is the centre one of the algorithm, the other four relations during the processing of the next time bin, when the corresponding other pads are the centre ones. In this way, the data of only six time bins needs to be expanded. This means, since the input data to the clusteriser is just the signals, noise and the baseline are added within the clusteriser to the signal pads and to the empty pads. In this way, a lot of disk space can be saved by not storing noise but only the real signals. To optimise the execution time, the tool *Callgrind* of the *Valgrind* framework [62] was used throughout the development and debugging process to profile the code. With this tool, the individual function calls together with a counter, how often the functions were called, and the corresponding execution time can be visualised. This allows to find the best place for optimisations and helps to remove unnecessary calls. Such a visualisation of the function calls is shown in figure 6.12. As can be seen, there are many small contributions to the overall execution time. The two big blocks, which are shown in blue, correspond to the clusteriser. The two main functions are the peak finder and a function to calculate the cluster properties, contributing 6.74 % and 8.09 % to the total time. Adding all parts of the clusteriser, one finds that only 19.95 % of the total time is used by the cluster finder and processing algorithm. Everything else is needed for the surrounding framework which takes also care of reading the input data from and storing to the disk.

To achieve this execution time, also Single Instruction, Multiple Data (SIMD) vectorisation was used. This is another kind of parallelism where the same instruction is performed on multiple data points simultaneously. Most of the modern CPUs provide SIMD instructions to improve the performance. This technique is applicable in the case of the clusteriser because always the same instructions need to be applied. For example, the computation of the relation between all neighbouring pads is always the same. So by combining the data of e.g. multiple rows within a SIMD vector, those rows can be processed simultaneously, improving the processing time significantly.

Chapter 7

Conclusion and Outlook

This thesis was accomplished in the scope of the ALICE TPC readout upgrade towards a continuous readout with a total data rate of 3.7 TB/s. The upgrade is necessary in preparation for the LHC Run 3, starting in 2021, where an interaction rate of 50 kHz in PbPb collisions is expected. The possible application of a Huffman encoded differential detector readout was studied. It was shown that with the length-limited Huffman encoding scheme, which is well suited for an implementation in a Front-End Device (FED), a sufficiently high compression factor is achieved. The compression factor is better than 2.5 over a large detector occupancy range of up to 40 % but degrades as soon as the noise contribution in the signal increases. With such a behaviour, no reliable detector operation is possible. This finding contributed to the review and a modification of the readout scheme presented in the original TPC upgrade Technical Design Report (TDR) towards an uncompressed raw data readout.

The main topic of this thesis was the development and implementation of the online Cluster Finder (CF) in VHDL. The same algorithm has also been implemented bit-accurate in modern C++. This additional implementation in the ALICE O^2 framework is required for verification purposes of the algorithm itself since it provides several orders of magnitude faster processing times compared to the VHDL simulation. It will also be used for the general reconstruction of simulated ALICE data. This 2-dimensional hardware CF is an essential preprocessing step in the future readout chain to enable an efficient physics data taking for the TPC. It runs on the FPGAs of the Common Readout Units (CRUs) and will inspect the whole data volume of the TPC in real-time already during the readout. Furthermore, all the necessary data preparation steps which need to be performed beforehand were also designed and implemented. This includes the decoding of the input data, a configurable sorting algorithm to map the up to 1 600 individual input channels of a CRU to the corresponding positions on a 2-dimensional grid in a flexible way and the Baseline Correction (BLC). This BLC includes the pedestal subtraction, the correction of the Common Mode (CM) effect as well as a multiplicative gain correction for each pad separately.

Each single module was simulated in detail to verify the proper functionality. Since the developments for such a central part of the readout chain are continuously being advanced, an automatic verification procedure was implemented for all individual modules in order to guarantee a consistent interface and behaviour also in future developments. Great

emphasis was placed on the validation of the CF. Several differently sized simulations were run to ensure the correct behaviour of all involved modules. The comparisons with the results of the software version did not show a single deviating bit in more than 2 million matched clusters.

In addition, the decoding of the input data and with that the basis of all following processing steps was validated in a real readout system during the test beam campaign in May 2017. It was also shown that the resources available in the FPGA of the CRU are sufficient to realise all components. Thus, the entire preprocessing chain in the Firmware (FW) for the TPC related CRUs was implemented and the detector could successfully be read out. The implementation was achieved well ahead of schedule as the first readout with a trigger on cosmic particles is anticipated for autumn 2019, when the first part of the TPC is upgraded.

The goal of this thesis was, to implement the CF for the TPC in the CRU and proof that the FPGA is sufficiently large for the needed digital logic. This has been achieved, although a few details could still be improved. For example could at some point a charge splitting be implemented in the CF. It was seen that especially in the high occupancy case the q_{tot} distribution deviates from the ideal one due to overlapping or nearby clusters. By separating the charges along the minima between the charge distributions, this deviation can be reduced. The for the splitting necessary flags of the minima are already implemented but not yet used. Another improvement which can be done in the CF is the peak finding itself. The current implementation compares the actual ADC value of neighbouring pads to look for rising and falling edges. This method is susceptible to fluctuations and noise. A different approach would be to compare the difference of these values to a threshold. A rising or falling edge is only found when the difference exceeds the threshold. This could suppress small fluctuations as expected from noise and reduce the number of clusters found incorrectly, improving the overall compression factor.

Appendix A

Acronyms

ACORDE	ALICE COsmic Ray DEtector
ADC	Analog-to-Digital Converter
ALICE	A Large Ion Collider Experiment
ALM	Adaptive Logic Module
ASIC	Application-Specific Integrated Circuit
BC	Bunch Crossing
BLC	Baseline Correction
CC	Clock Cycle
CDC	Clock-Domain Crossing
CDF	Cluster Data Format
CERN	Conseil Européen pour la Recherche Nucléaire
CF	Cluster Finder
CM	Common Mode
CP	Cluster Processor
CR	Counting Room
CRU	Common Readout Unit
CSA	Charge Sensitive Amplifier
CTP	Central Trigger Processor
DAQ	Data Acquisition
DAS	Direct ADC Serialisation
DCAL	Di-jet CALorimeter
DCS	Detector Control System
DMA	Direct Memory Access
DSP	Digital Signal Processor
EMCAL	ElectroMagnetic CALorimeter
EOP	End-Of-Packet
EPN	Event Processing Node
FEC	Front-End Card
FED	Front-End Device
FEE	Front-End Electronics
FIFO	First In, First Out
FLP	First Level Processor
FMD	Forward Multiplicity Detector
FP	Fixed-Point
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
FW	Firmware
GBT	GigaBit Transceiver
GEM	Gas Electron Multiplier

HB	Heart Beat
HLT	High Level Trigger
HMPID	High Momentum Particle IDentification
HW	Half-Word
IBF	Ion Back-Flow
IP	Intellectual Property
IROC	Inner Readout Chamber
ITS	Inner Tracking System
LE	Logic Element
LEIR	Low Energy Ion Ring
LEP	Large Electron Positron
LHC	Large Hadron Collider
LS	Long Shutdown
LSB	Least Significant Bit
LTU	Local Trigger Unit
LUT	Lookup-Table
M20K	20 kbit Memory Block
MAPS	Monolithic Active Pixel Sensors
MFT	Muon Forward Tracker
MLAB	Memory Logic Array Block
MRPC	Multi-gap Resistive-Plate Chamber
MSB	Most Significant Bit
MUX	Multiplexer
MWPC	Multi-Wire Proportional Chamber
OROC	Outer Readout Chamber
PCB	Printed Circuit Board
PHOS	PHOton Spectrometer
PID	Particle Identification
PLL	Phase-Locked Loop
PMD	Photon Multiplicity Detector
PON	Passive Optical Network
PS	Proton Synchrotron
PSB	Proton Synchrotron Booster
QA	Quality Assurance
QGP	Quark–Gluon Plasma
RAM	Random-Access Memory
RCC	Ring Cathode Chamber
RDH	Raw Data Header
ROC	Readout Chamber
SC	Slow-Control
GBT-SCA	GBT-Slow Control Adapter ASIC
SDD	Silicon Drift Detector
SIMD	Single Instruction, Multiple Data
SM	Super-Module
SOP	Start-Of-Packet
SPD	Silicon Pixel Detector
SPS	Super Proton Synchrotron
SSD	Silicon Strip Detector
TDR	Technical Design Report
TOF	Time-Of-Flight
TPC	Time-Projection Chamber
TR	Transition Radiation
TRD	Transition Radiation Detector
TTC	Timing, Trigger and Control

TTS	Trigger, Timing and clock distribution System	VHDL	Very high speed integrated circuit Hardware Description Language
UL	User Logic	ZDC	Zero Degree Craorimeter
UUT	Unit Under Test	ZS	Zero Suppression

Appendix B

Pad Plane Mapping

The mapping of the individual pads to the SAMPA chip within a Front-End Cards (FECs) is shown on the following pages for all ROCs. One TPC sector is composed of an Inner Readout Chamber (IROC) and the three Outer Readout Chambers (OROCs). Figure B.1 shows the mapping for the IROC, figure B.2 for the OROC 1, figure B.3 for the OROC 2 and figure B.4 for the OROC 3. Each pad displays the ID of the SAMPA within a FEC to which it is connected to.

Please notice the horizontal straight blue lines in the mappings. Although the blue boxes indicate in general just which pads are grouped together within a single *connector* between the pad plane and the FEC (each FEC is connected via 4 connectors to the pad plane), the straight ones also mark the different *readout regions*. Each of the 10 readout region is transmitted to and processed by a single Common Readout Unit (CRU). With that, it is ensured, that always a complete pad rows is transmitted to the same CRU which is important for the cluster finding. The straight lines can be grouped into two categories:

1.
   - between rows* 31/32 for figure B.1
   - between rows 62/63 for figure B.1 and figure B.2
   - between 69/70 for figure B.2 and figure B.3
   - between 126/127 for figure B.3 and figure B.4.
2.
   - between rows 16/17 and 47/48 for figure B.1
   - between rows 80/81 for figure B.2
   - between rows 112/113 for figure B.3
   - between rows 139/140 for figure B.4

The first category has a change in the SAMPA number when crossing the line. This indicates a transition to a new FEC and is of no further importance. The second category has the SAMPA ID 2 on both sides of the line and marks the transition to a different CRU (region) within a single SAMPA chip. Here, the pads are connected to SAMPA 2 in such a way that the pads below are always SAMPA channels 0–15 and above are always SAMPA channels

*The row numbers are written always on the right side, next to the pads.

region	partition	FECs	rows	max. pads
0	0	15	17	76
1	0	15	15	84
2	1	18	16	94
3	1	18	15	100
4	2	18	18	84
5	2	18	16	94
6	3	20	16	106
7	3	20	14	118
8	4	20	13	128
9	4	20	12	138

Table B.1: Key parameters of the pad plane regions.

16–31. In this way, even if half of the data of SAMPA 2 is shipped to one CRU und the other half to a different one, complete pad rows can always be restored.

The number of FECs in the individual readout regions can be read off and varies from 15 to 20. It is summarised together with more key parameters of the individual regions in table B.1. The partition is the subdivision of a whole sector in radial direction into units of FECs. Therefore always two regions belong to one partition. The number of FECs is in pad direction and means the number of neighbouring FECs belonging to the same partition. There is also the number of rows within each region given, which is important for the number of individual Cluster Finder (CF) instances since the cluster finding is done in each row individually. This number ranges from 12 to 18 in the lowest number in the outermost region and the lowermost number in region four. The last column shows the highest number of pads which can be found within a single row within this region.

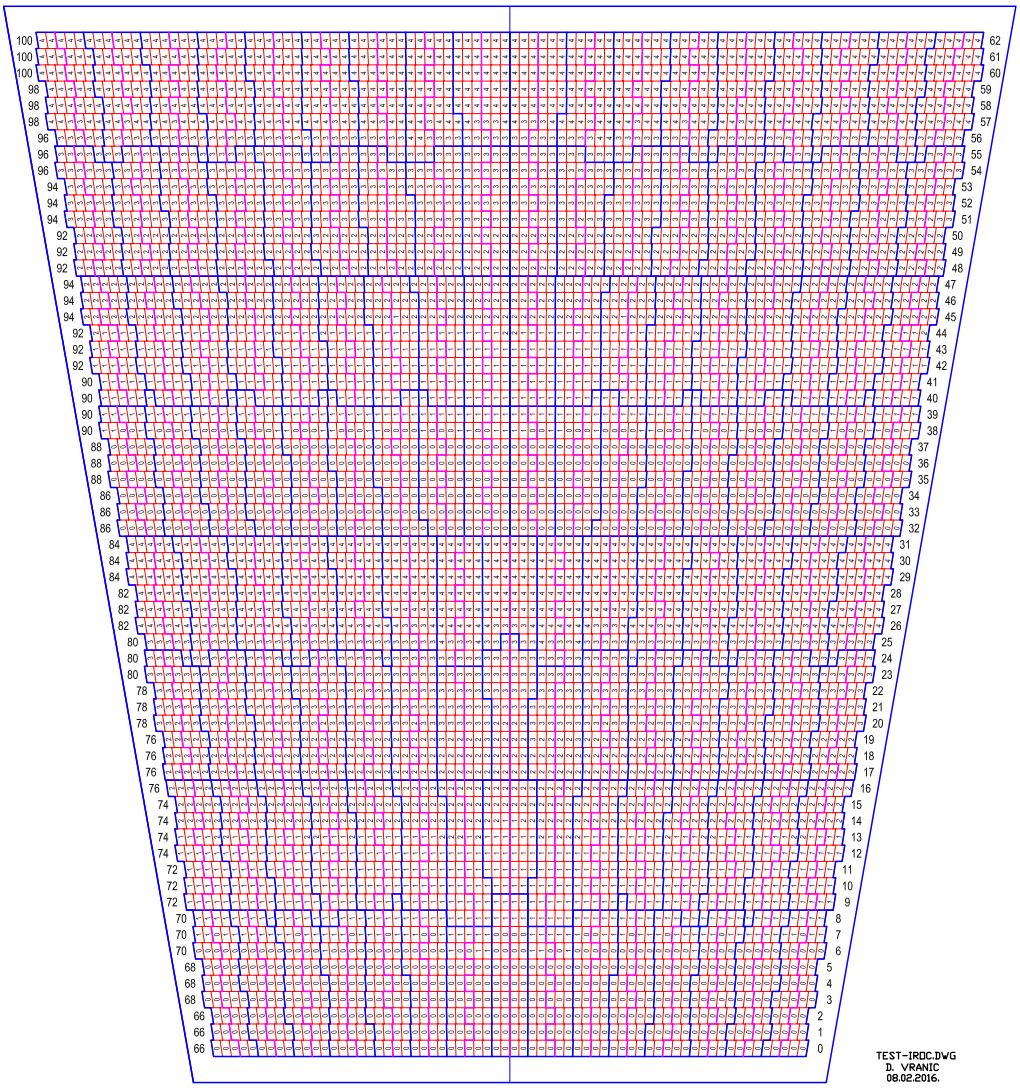


Figure B.1: The pad plane of the IROC. Each pad shows the ID of the SAMPA within a FEC to which it is connected to, taken from [34].

Figure B.2: The pad plane of the OROC 1. Each pad shows the ID of the SAMPA within a FEC to which it is connected to, taken from [34].

Figure B.3: The pad plane of the OROC 2. Each pad shows the ID of the SAMPA within a FEC to which it is connected to, taken from [34].

Figure B.4: The pad plane of the OROC 3. Each pad shows the ID of the SAMPA within a FEC to which it is connected to, taken from [34].

Appendix C

The Raw Data Header

Every data package needs to be preceded by a Raw Data Header (RDH), to identify and efficiently process the data in the First Level Processor (FLP). The RDH consists in total of 512 bit, containing all relevant information. Depending on where the header is generated, either in the Front-End Electronics (FEE) and sent via the GBT system, in the FEE and sent via the DDL system or in the User Logic (UL) of the CRU, the individual words of the RDH are differently defined in length and numbering. Once the header it is stored in the memory of the FLP they have the same size and format. The version which is important for the TPC readout is the one generated in the UL of the CRU. It is shown in figure C.1. The RDH consists of four 128 bit words. The red fields need to be filled by the UL while the remaining ones are filled by the interface module to the Direct Memory Access (DMA) engine and must be set to zero by the UL. Since they are not of interest for the TPC, only the definitions of the red fields are given in the following. This is taken from [64].

Priority bit
: Field to adjust the priority of the following data. If this is set to 0x1, the packet is propagated with a higher priority than the others. This can be used e.g. to report a Heart Beat (HB) frame when the buffers are full.

FEE ID
: A unique ID, assigned to the FEE.

Block length
: The size of the payload (without the RDH) in bytes.

Header size
: The size of the RDH in bytes ($4 \times 128\,\text{bit} = 64\,\text{B}$).

Header version
: Version number of the header.

HB orbit
: Heart Beat orbit.

TRG orbit
Trigger orbit.

TRG type
The current trigger type, set by the Central Trigger Processor (CTP).

HB BC
Heart Beat Bunch Crossing.

TRG BC
Trigger Bunch Crossing.

Pages counter
If the data volume of one trigger (or HB) exceeds the maximum of 8 kB, this counter is used to keep track of the pages belonging to the same trigger.

Stop bit
Bit to identify the last page (set to 0x1, otherwise stays at 0x0) of a series of pages belonging to the same trigger.

PAR
The Pause-and-Reconfigure field is used by the detector to trigger a synchronous reconfiguration of the FEE.

Detector field
A field for detector specific content.

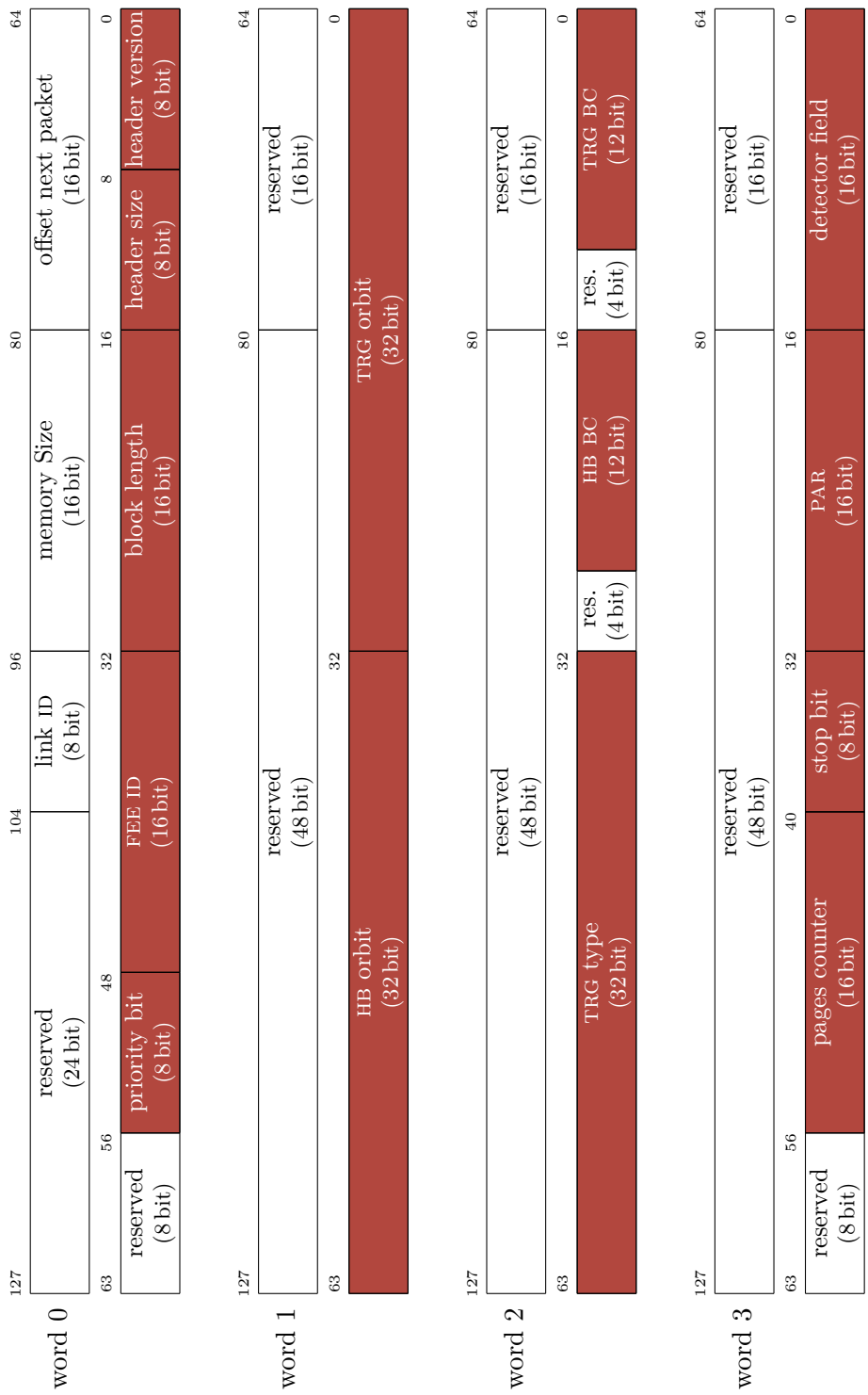


Figure C.1: The four 128 bit words of the RDH version 3. The red fields need to be filled by the UL, based on [64].

Appendix D

Bibliography

[1] D. Boyanovsky et al. "Phase transitions in the early and the present universe". In: *Ann. Rev. Nucl. Part. Sci.* 56 (2006), pp. 441–500. DOI: 10.1146/annurev.nucl.56.080805.140539. arXiv: hep-ph/0602002 [hep-ph] (cit. on p. 1).

[2] Edward V. Shuryak. "Theory of Hadronic Plasma". In: *Sov. Phys. JETP* 47 (1978). [Zh. Eksp. Teor. Fiz.74,408(1978)], pp. 212–219 (cit. on p. 1).

[3] The ALICE Collaboration. *Upgrade of the ALICE Experiment: Letter of Intent.* Tech. rep. CERN-LHCC-2012-012. LHCC-I-022. ALICE-UG-002. Geneva: CERN, Aug. 2012. URL: http://cds.cern.ch/record/1475243 (cit. on pp. 1, 2, 8, 15).

[4] *LS2 preparatory meeting.* Jan. 2019. URL: https://indico.cern.ch/event/776676/ (visited on 01/25/2019) (cit. on pp. 1, 16).

[5] The ALICE Collaboration. *Upgrade of the ALICE Time Projection Chamber.* Tech. rep. CERN-LHCC-2013-020. ALICE-TDR-016. Oct. 2013. URL: https://cds.cern.ch/record/1622286 (cit. on pp. 2, 10, 12, 14, 15, 17, 18, 24, 36, 66–68, 72).

[6] Lyndon Evans and Philip Bryant. "LHC Machine". In: *Journal of Instrumentation* 3.08 (2008), S08001. URL: http://stacks.iop.org/1748-0221/3/i=08/a=S08001 (cit. on p. 3).

[7] Esma Mobs. "The CERN accelerator complex - August 2018. Complexe des accélérateurs du CERN - Août 2018". In: (Aug. 2018). General Photo. URL: http://cds.cern.ch/record/2636343 (cit. on p. 3).

[8] *The accelerator complex.* URL: https://home.cern/science/accelerators/accelerator-complex (visited on 01/21/2019) (cit. on p. 4).

[9] The ATLAS Collaboration. "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC". In: *Phys. Lett.* B716 (2012), pp. 1–29. DOI: 10.1016/j.physletb.2012.08.020. arXiv: 1207.7214 [hep-ex] (cit. on p. 4).

[10] The CMS Collaboration. "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC". In: *Phys. Lett.* B716 (2012), pp. 30–61. DOI: 10.1016/j.physletb.2012.08.021. arXiv: 1207.7235 [hep-ex] (cit. on p. 4).

[11] The ATLAS Collaboration. "The ATLAS Experiment at the CERN Large Hadron Collider". In: *Journal of Instrumentation* 3.08 (2008), S08003. URL: http://stacks.iop.org/1748-0221/3/i=08/a=S08003 (cit. on p. 4).

[12] The ALICE Collaboration. "The ALICE experiment at the CERN LHC". In: *Journal of Instrumentation* 3.08 (2008), S08002. URL: http://stacks.iop.org/1748-0221/3/i=08/a=S08002 (cit. on pp. 4, 6, 7).

[13] The CMS Collaboration et al. "The CMS experiment at the CERN LHC". In: *Journal of Instrumentation* 3.08 (2008), S08004. URL: http://stacks.iop.org/1748-0221/3/i=08/a=S08004 (cit. on p. 4).

[14] The LHCb Collaboration. "The LHCb Detector at the LHC". In: *Journal of Instrumentation* 3.08 (2008), S08005. URL: http://stacks.iop.org/1748-0221/3/i=08/a=S08005 (cit. on p. 4).

[15] The LHCf Collaboration. "The LHCf detector at the CERN Large Hadron Collider". In: *Journal of Instrumentation* 3.08 (2008), S08006. URL: http://stacks.iop.org/1748-0221/3/i=08/a=S08006 (cit. on p. 4).

[16] The TOTEM Collaboration. "The TOTEM Experiment at the CERN Large Hadron Collider". In: *Journal of Instrumentation* 3.08 (2008), S08007. URL: http://stacks.iop.org/1748-0221/3/i=08/a=S08007 (cit. on p. 4).

[17] The MoEDAL Collaboration. *Technical Design Report of the MoEDAL Experiment*. Tech. rep. CERN-LHCC-2009-006. MoEDAL-TDR-001. June 2009. URL: https://cds.cern.ch/record/1181486 (cit. on p. 4).

[18] The ALICE Collaboration. "Performance of the ALICE Experiment at the CERN LHC". In: (2014). DOI: 10.1142/S0217751X14300440. eprint: arXiv:1402.4476 (cit. on p. 4).

[19] Arturo Tauro. *3D ALICE Schematic RUN2 - with Description*. May 2011 (cit. on p. 5).

[20] J. Alme et al. "The ALICE TPC, a large 3-dimensional tracking device with fast readout for ultra-high multiplicity events". In: (2010). DOI: 10.1016/j.nima.2010.04.042. eprint: arXiv:1001.1950 (cit. on pp. 6, 11, 12).

[21] The ALICE Collaboration. "The ALICE Transition Radiation Detector: construction, operation, and performance". In: (2017). DOI: 10.1016/j.nima.2017.09.028. eprint: arXiv:1709.02743 (cit. on p. 7).

[22] Andrea Alici. "The MRPC-based ALICE Time-Of-Flight detector: status and performance". In: (2012). DOI: 10.1016/j.nima.2012.05.004. eprint: arXiv:1203.5976 (cit. on p. 7).

[23] The ALICE Collaboration. *Addendum of the Letter of Intent for the upgrade of the ALICE experiment : The Muon Forward Tracker*. Tech. rep. CERN-LHCC-2013-014. LHCC-I-022-ADD-1. Final submission of the presetn LoI addendum is scheduled for September 7th. Geneva: CERN, Aug. 2013. URL: http://cds.cern.ch/record/1592659 (cit. on p. 8).

[24] Sabyasachi Siddhanta. “The upgrade of the Inner Tracking System of ALICE”. In: *Nuclear Physics A* 931 (2014). QUARK MATTER 2014, pp. 1147–1151. ISSN: 0375-9474. DOI: https://doi.org/10.1016/j.nuclphysa.2014.09.041. URL: http://www.sciencedirect.com/science/article/pii/S0375947414004230 (cit. on p. 8).

[25] The ALICE Collaboration. *Technical Design Report for the Upgrade of the ALICE Inner Tracking System*. Tech. rep. CERN-LHCC-2013-024. ALICE-TDR-017. Nov. 2013. URL: https://cds.cern.ch/record/1625842 (cit. on p. 8).

[26] The ALICE Collaboration. *Upgrade of the ALICE Readout & Trigger System*. Tech. rep. CERN-LHCC-2013-019. ALICE-TDR-015. Sept. 2013. URL: http://cds.cern.ch/record/1603472 (cit. on pp. 8, 17, 26, 27).

[27] M. Tanabashi et al. “Review of Particle Physics”. In: *Phys. Rev. D* 98 (3 Aug. 2018), p. 030001. DOI: 10.1103/PhysRevD.98.030001. URL: https://link.aps.org/doi/10.1103/PhysRevD.98.030001 (cit. on pp. 9, 80).

[28] Hermann Kolanoski and Norbert Wermes. *Teilchendetektoren. Grundlagen und Anwendungen*. ger. 1. Aufl. 2016. SpringerLink : Bücher. Berlin, Heidelberg: Springer Spektrum, 2016. ISBN: 978-3-662-45350-6. DOI: 10.1007/978-3-662-45350-6. URL: http://dx.doi.org/10.1007/978-3-662-45350-6 (cit. on p. 9).

[29] The ALICE Collaboration. *ALICE Technical Design Report of the Time Projection Chamber*. Technical Design Report ALICE. Geneva: CERN, 2000. URL: http://cds.cern.ch/record/451098 (cit. on pp. 11, 26, 37, 65, 66).

[30] F.V. Böhmer et al. “Simulation of space-charge effects in an ungated GEM-based TPC”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 719 (2013), pp. 101–108. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2013.04.020. URL: http://www.sciencedirect.com/science/article/pii/S0168900213004166 (cit. on pp. 13, 14).

[31] F. Sauli. “GEM: A new concept for electron amplification in gas detectors”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 386.2 (1997), pp. 531–534. ISSN: 0168-9002. DOI: https://doi.org/10.1016/S0168-9002(96)01172-2. URL: http://www.sciencedirect.com/science/article/pii/S0168900296011722 (cit. on p. 12).

[32] *TPC wiki*. URL: https://twiki.cern.ch/twiki/bin/view/ALICE/TPC_Installation (visited on 12/27/2018) (cit. on pp. 16, 93).

[33] Harald Appelshaeuser et al. *Readout scheme of the upgraded ALICE TPC*. Nov. 2016. URL: https://cds.cern.ch/record/2231785 (cit. on pp. 17, 20, 21, 26, 49).

[34] *ALICE TPC Read-Out Upgrade Wiki*. URL: https://espace.cern.ch/alice-tpc-cru/_layouts/15/start.aspx#/ALICE%20TPC%20CRU%20Wiki/Mapping.aspx (cit. on pp. 18, 56, 66, 68, 123–126).

[35] David A. Huffman. “A method for the construction of minimum-redundancy codes”. In: *Proceedings of the IRE* 40.9 (1952), pp. 1098–1101 (cit. on p. 21).

[36] Khalid Sayood. *Introduction to data compression*. eng. 4th ed. Morgan Kaufmann series in multimedia information and systems. Waltham, Mass.: Morgan Kaufmann, 2012, Online–Ressource (1 v. p.) ISBN: 978-0-12-416000-2. URL: http://proquest.tech.safaribooksonline.de/9780124157965 (cit. on p. 21).

[37] Lawrence L. Larmore and Daniel S. Hirschberg. "A Fast Algorithm for Optimal Length-limited Huffman Codes". In: *J. ACM* 37.3 (July 1990), pp. 464–473. ISSN: 0004-5411. DOI: 10.1145/79147.79150. URL: http://doi.acm.org/10.1145/79147.79150 (cit. on p. 23).

[38] Arild Velure and Bruno Sanches. *SAMPA V3 Specification*. Revision 0.2. Sept. 2018 (cit. on p. 29).

[39] P. Moreira et al. "The GBT Project". In: *Proceedings, Topical Workshop on Electronics for Particle Physics (TWEPP09)* (2009), pp. 342–346. URL: https://cds.cern.ch/record/1235836 (cit. on p. 29).

[40] K. Wyllie P. Moreira J. Christiansen. *GBTx Manual*. Version 0.15. Oct. 2016. URL: https://espace.cern.ch/GBT-Project/GBTX/Manuals/gbtxManual.pdf (cit. on pp. 29, 31).

[41] Albert X. Widmer and Peter A. Franaszek. "A DC-balanced, partitioned-block, 8B/10B transmission code". In: *IBM Journal of research and development* 27.5 (1983), pp. 440–451 (cit. on p. 30).

[42] Alex Kluge and Pierre Vande Vyvre. *The detector read-out in ALICE during Run 3 and 4*. version 1.5. CERN, EP Department. June 2016. URL: https://twiki.cern.ch/twiki/pub/ALICE/CruHwFwSwDev/ALICErun34_readout.pdf (cit. on pp. 33, 36).

[43] J.P. Cachemiche et al. "The PCIe-based readout system for the LHCb experiment". In: *Journal of Instrumentation* 11.02 (2016), P02013. URL: http://stacks.iop.org/1748-0221/11/i=02/a=P02013 (cit. on p. 33).

[44] E. David et al. *CRU Specification*. Version 0.7. CERN. June 2016. URL: https://twiki.cern.ch/twiki/pub/ALICE/CruHwFwSwDev/CRU_Specification_v0.7.pdf (cit. on p. 33).

[45] *Intel Arria10 Device Overview*. Intel. Apr. 2018. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/arria-10/a10_overview.pdf (cit. on pp. 34, 58).

[46] Jean-Pierre Cachemiche. *Photo of the CRU v1*. Centre de Physique des Particules de Marseille, Apr. 2015 (cit. on p. 35).

[47] P. Buncic et al. *Technical Design Report for the Upgrade of the Online-Offline Computing System*. Tech. rep. CERN-LHCC-2015-006. ALICE-TDR-019. Apr. 2015. URL: https://cds.cern.ch/record/2011297 (cit. on pp. 36, 111).

[48] *Understanding Metastability in FPGAs*. Version 1.2. Altera. July 2009. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01082-quartus-ii-metastability.pdf (cit. on p. 41).

[49] *GitLab Repository, CRU firmware core*. Feb. 2019. URL: https://gitlab.cern.ch/alice-cru/cru-fw (visited on 02/01/2019) (cit. on pp. 43–45).

[50] O. Bourrion et al. *Interface between CTS-CRU and CTS-Detector Front Ends Trigger Notes for Developers*. Oct. 2018 (cit. on p. 45).

[51] *Recommended Physical RAM for Intel Devices 17.0*. URL: http://fpgasoftware.intel.com/requirements/17.1/ (visited on 11/02/2018) (cit. on p. 49).

[52] *Intel Quartus Prime Pro Edition User Guide*. Intel. Oct. 2018. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug-qpp-compiler.pdf (cit. on p. 49).

[53] Dr. Mesut Arslandok. private communication (cit. on p. 49).

[54] Donald Ervin Knuth. *The art of computer programming, Volume 3, Sorting and searching*. eng. Second edition (Online edition). Upper Saddle River, NJ: Addison-Wesley, 1998, Online–Ressource (1 volume). ISBN: 978-0-201-89685-5. URL: http://proquest.tech.safaribooksonline.de/9780321635792 (cit. on p. 57).

[55] R. C. Bose and R. J. Nelson. "A Sorting Problem". In: *J. ACM* 9.2 (Apr. 1962), pp. 282–296. ISSN: 0004-5411. DOI: 10.1145/321119.321126. URL: http://doi.acm.org/10.1145/321119.321126 (cit. on p. 57).

[56] *Intel Arria10 Native Fixed Point DSP IP Core User Guide*. Intel. Mar. 2017. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_nfp_dsp.pdf (cit. on pp. 61, 62).

[57] *Intel FPGA Integer Arithmetic IP Cores User Guide*. Intel. Nov. 2017. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_lpm_alt_mfug.pdf (cit. on p. 63).

[58] Thomas H. Cormen et al. *Introduction to algorithms*. MIT press, 2009 (cit. on p. 71).

[59] Ansgar Steland. *Basiswissen Statistik. Kompaktkurs für Anwender aus Wirtschaft, Informatik und Technik*. ger. 4. Aufl. 2016. SpringerLink : Bücher. Berlin, Heidelberg: Springer Spektrum, 2016. ISBN: 978-3-662-49948-1. DOI: 10.1007/978-3-662-49948-1. URL: http://dx.doi.org/10.1007/978-3-662-49948-1 (cit. on p. 80).

[60] *ModelSim*. URL: https://www.mentor.com/products/fpga/verification-simulation/modelsim/?sfm=free_form (visited on 01/25/2019) (cit. on p. 93).

[61] H. Engel et al. *The C-RORC PCIe Card and its Application in the ALICE and ATLAS Experiments*. Tech. rep. ATL-DAQ-PROC-2014-039. AIDA-PUB-2015-022. Geneva: CERN, Oct. 2014. URL: http://cds.cern.ch/record/1958271 (cit. on p. 109).

[62] *Valgrind*. URL: http://www.valgrind.org (visited on 01/15/2019) (cit. on pp. 112, 113).

[63] *KCachegrind*. URL: http://kcachegrind.sourceforge.net/html/Home.html (visited on 01/25/2019) (cit. on p. 112).

[64] *ALICE RUN 3 Raw Data Header (RDH V3)*. Sept. 2018. URL: https://docs.google.com/document/d/1otkSDYasqpVBDnxplBI7dWNxaZohctA-bvhyrzvtLoQ/ (visited on 10/26/2018) (cit. on pp. 127, 129).

[65] *DeepL Translator*. Feb. 2019. URL: https://www.deepl.com/translator (visited on 02/01/2019).

Appendix E

Acknowledgments

I would like to take the opportunity to thank my supervisor Prof. Dr. Johanna Stachel for the support during my doctoral studies, for the great opportunity to work in such an impressive and challenging field, and for the trust she has placed in me and my abilities. Actually, I owe her much more. Also because of the internship that I was able to do in her group back in 2005, I decided to study physics, which had a decisive influence on my life and why I still have a very special connection to the TRD today.

Next I thank Dr. Jorge Mercado for his foresight. Without him, I would never have taken the step towards digital design which opened up a whole new world for me in understanding complex electrical systems and detector electronics. Unfortunately he did not stay in academia long enough to see this thesis completed, but during the time he was still around he supported me wherever he could.

Further, I would like to thank my colleagues, PD Dr. Kai Schweda for the very interesting (and always challenging) discussions on various physical and other topics, for the sometimes more and sometimes less useful advices also in private life and for the detailed proofreading of this work. I thank Dr. Alexander Schmah for the extensive review of this thesis and many conversations about the working principles of 521 different gaseous detectors. They were really fun and advanced also my understanding of the many contributing details. Many thanks to PD Dr. Yvonne Pachmayer for the help with one or the other reference as well as the correction of parts of this document. I would also like to thank her for the many, many chats during the coffee breaks throughout the last years. Thanks also to my office neighbour Ole Schmidt that he patiently listened to everything I had to get rid of, for his hints and advice on all the small and big problems I was stuck with.

I would also like to thank my parents with all my heart. They were always there when they where needed so I could concentrate on this work. And of course I would like to thank my wife Regina. She took the burden off me wherever and whenever she could, so this work is just as well her merit.

But above all, I want to thank my son Erik:
Your laughter has brightened up even the hardest times
and you have shown me what is most important in life.
Thank you very much!

Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Heidelberg, den 3. Februar 2019 .....